

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A164 044



DTIC
 ELECTE
 FEB 13 1986
 S D

DEPARTMENT OF AERONAUTICS AND SPACE
 RESEARCH AND DEVELOPMENT
 DIVISION
 AIR FORCE INSTITUTE OF TECHNOLOGY
 WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-3900

DTIC FILE COPY

DISTRIBUTION STATEMENT A
 Approved for public release
 Distribution Unlimited

DEPARTMENT OF THE AIR FORCE
 AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

1

AFIT/GE/ENG/85D-46

DTIC
ELECTE
FEB 13 1986
S D D

DEVELOPMENT OF A COMPUTER AIDED DESIGN
PACKAGE FOR CONTROL SYSTEM DESIGN
AND ANALYSIS FOR A PERSONAL COMPUTER
(ICECAP-PC)
THESIS

Volume II of II

Susan K. Mashiko
Captain, USAF

Gary C. Tarczynski
Captain, USAF

AFIT/GE/ENG/85D-46

Approved for public release; distribution unlimited

Appendix D: Data Dictionary

This appendix contains the data dictionary. The name, description, make-up, source, destination and the use of each of the variables, constants, type definitions and procedures are contained in this section. The entries are self-explanatory.

Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification:	
By	
Distribution	
Availability Codes	
Dist	Avail a. d/ or Special
A-1	



DATA DICTIONARY FOR ICECAP-PC

```
*****
Name:      abbrev_code
Aliases:   None
Type Of Entry: Constant
Description: This character is attached to the beginning of an
              abbreviation of a command word. It indicates to
              the code that it is the abbreviation for some other
              word.
Make_up:   Char
Source:
Destination:
Used In:   val_n_dec
*****
```

```
*****
Name:      abort
Aliases:   None
Type Of Entry: Label
Description: This is a label for a goto statement.
Make_up:   Char
Source:
Destination:
Used In:   form
*****
```

```
*****
Name:      abort2
Aliases:   None
Type Of Entry: Label
Description: This is a label for a goto statement.
Make_up:   Char
Source:
Destination:
Used In:   form
*****
```

```
*****
Name:      abort_command
Aliases:   None
Type Of Entry: Global variable and data flow
Description: This variable is the flag that indicates whether
              the command input by the user is the one to abort.
Make_up:   boolean
Source:    declared in icecappc
Destination:
Used In:   get_int,      get_strng,      readcom,
```

get_cmd,	recover,	update
get_real,	get_r_num,	get_fact,
get_unfact,	poly,	getff,
matrxmanip1,	matrxmanip2,	get_matrx_entries,
matrxadd,	mmatrxmlt,	get_mat,
matrxsub,	inroot,	delroot,
matrxinv,	form,	ppoly,
mmatrix,	get_poly,	get_poly_name

Name: abort_str
 Aliases: None
 Type Of Entry: Global constant
 Description: Literal that indicates abort_command
 Make_up: Character literal (\$)
 Source: Declared in icecappc
 Destination:
 Used In: input by the user

Name: again
 Aliases: None
 Type Of Entry: Label
 Description: Label for a goto statement.
 Make_up: Character
 Source:
 Destination:
 Used In: get_poly_name

Name: amat
 Aliases: None
 Type Of Entry: Variable
 Description: This is a matrix.
 Make_up: matrix
 Source:
 Destination:
 Used In: matrxadd, matrxsub, mmatrxmlt,
 smatrxmlt, matrxtran

Name: apoly
 Aliases: None
 Type Of Entry: Variable
 Description: This is a polynomial.

Make_up: polynomial

Source:

Destination:

Used In: spolymlt, polymlt, polysub,
polyadd

Name: as_assigned

Aliases: None

Type Of Entry: Global constant

Description: Literal that indicates input is from the source as
indicated by the status of the in_terminal flag

Make_up: Character literal

Source: Declared in icecappc

Destination:

Used In: disp_line, prompt_cmd, get_int,
gettf, get_real, recover,
make_pretty, update, get_fact,
ccopyy, get_r_num, get_poly_name,
ppoly, make_pretty_small_matrix, get_mat,
disppoly, make_pretty_large_matrix_one,
inroot, delroot, disp_matrx,
mmatrix

Name: b

Aliases: None

Type Of Entry: Variable

Description: This variable is used as a temporary storage area
the polynomial.

Make_up: b1

Source:

Destination:

Used In: roots

Name: b1

Aliases: None

Type Of Entry: Variable

Description: This variable is used as a temporary storage area
the polynomial.

Make_up: array[1..30] of real

Source:

Destination:

Used In: roots

```

*****
Name:      backspace
Aliases:   None
Type Of Entry: Global constant
Description: Decimal ASCII value for the backspace character.
Make_up:   Integer
Source:    Declared in icecappc
Destination: N/A
Used In:   del_lst_ch,      ck_chr,      get_strng
*****

```

```

*****
Name:      blanks
Aliases:   None
Type Of Entry: Global variable
Description: A string of blank characters used to 'erase' a line
            on the CRT. Can also be used as a source of blank
            characters to clear strings throughout the program.

Make_up:   string[ screen_size ]
Source:    Declared in icecappc
Destination: N/A
Used In:   pause,          get_data,      clear_msg,
            prompt_cmd,    proces_error,  recover,
            update,        get_r_num,     get_poly_name,
            ppoly,         delroot
*****

```

```

*****
Name:      bld_stat_line
Type:      Procedure
Description: This procedure builds the status line from initial-
            ization data from disk staorage. The data is in param
            group 1.

Global Variables Used:  stat_line,  help_level,  temp,
                       printer,    trans

Global Variables Changed: stat_line

Global Constants Used:  None

Passed Variables:  help_level,  temp,          printer,
                  trans

Returned:          None
Files Read:        None
Files Written:     None
Aliases:           None
Procedures Called: None
Called By:         get_data

Version:           1.2
Date:              18 Oct 83
Author:            Vincent M. Parisi II, Capt, USAF

```

Contained In File: GETDAT.PAS

Name: bmat
Aliases: None
Type Of Entry: Variable
Description: This is a matrix.
Make_up: matrix
Source:
Destination:
Used In: matrxadd, matrxsub, mmatrixmlt,
smatrixmlt, matrxtran

Name: bpoly
Aliases: None
Type Of Entry: Variable
Description: This is a polynomial.
Make_up: polynomial
Source:
Destination:
Used In: spolymlt, polymlt, polysub,
polyadd

Name: buffer
Aliases: None
Type Of Entry: Global type definition
Description: This is the type definition of the structure that holds the user input commands once it has been separated into individual words, one word per storage location.
Make_up: array[1..buffersize] of string[wordsize]
Source: Declared in icecappc
Destination: N/A
Used In: icecappc

Name: bufferpointer
Aliases: None
Type Of Entry: Variable and data flow
Description: This variable points to the next location in the cmdbuffer that will receive the next command word.
Make_up: Integer

Source: get_cmd, readcom
Destination: N/A
Used In: get_cmd, readcom, proces_error

Name: buffersize
Aliases: None
Type Of Entry: Global constant
Description: Maximum number of individual words allowed in a user
input command.
Make_up: Integer (6)
Source: Declared in icecappc
Destination: N/A
Used In: icecappc, get_line, readcom,
val_n_dec, displa_commandword, help,
select_routine

Name: c
Aliases: None
Type Of Entry: Variable
Description: This variable is used as a temporary storage area
the polynomial.
Make_up: c1
Source:
Destination:
Used In: roots

Name: c
Aliases: None
Type Of Entry: Variable
Description: This variable is used in the IBM unique procedure
stdout
Make_up: char
Source:
Destination:
Used In: stdout

Name: c1
Aliases: None
Type Of Entry: Variable
Description: This variable is used as a temporary storage area
the polynomial.

Make_up: array[1..30] of real
Source:
Destination:
Used In: roots

Name: C1
Aliases: None
Type Of Entry: Variable
Description: Col + width - 1
Make_up: Integer
Source:
Destination:
Used In: rectangle

Name: call_routine
Aliases: None
Type Of Entry: Global variable and data flow
Description: This variable is the name of the routine to call
to accomplish the desired action from the user
input.

Make_up: cmdword
Source: Declared in icecappc
Destination:
Used In: icecappc, get_data, get_cmd,
val_n_dec, select_routine

Name: ccopy
Type: Procedure
Description: This procedure manages the copy function. It gets the
source and destination location then performs the
copy operation. (called ccopy because of the
compiler function called copy)

Global Variables Used: cmdbuffer
Global Variables Changed: None
Global Constants Used: as_assigned
Passed Variables: cmdbuffer

Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: clear, gotoxy, trim, out_string,
get_location, move_matrix, move_poly,
move_tf, highlight, nohighlight, pause,

Called By: disp_msg
select_routine

Version: 2.0
Date: 22 Sep 85
Author: Vincent M. Parisi II, Capt, USAF
Modified by: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: COPY.PAS

Name: cdpol
Aliases: None
Type Of Entry: Variable
Description: This is a polynomial, which is used as a temporary storage area for the CLTF denominator polynomial.
Make_up: polynomial
Source:
Destination:
Used In: form

Name: ch
Aliases: None
Type Of Entry: Variable
Description: The input from the user in the get_strng procedure. Appended onto the string if a valid ASCII character, ignored if a control character.
Make_up: Char
Source:
Destination:
Used In: get_chr, get_int, ck_chr, get_real

Name: change_msg
Aliases: None
Type Of Entry: Constant
Description: The constant is the message number for help.
Make_up: Integer
Source:
Destination:
Used In: help

Name: check

Aliases: None
Type Of Entry: Variable
Description: This is a temporary storage area that is used to compare the passed command word against certain keywords.
Make_up: cmdword
Source:
Destination:
Used In: define

Name: check_word
Type: Function
Description: This procedure checks the word to see if it matches the dictionary entries.
Global Variables Used: None
Global Variables Changed: None
Global Constants Used: None
Passed Variables: decode, command
Returned: command, check_word
Files Read: None
Files Written: None
Aliases: None
Procedures Called: trim
Called By: val_n_dec

Version: 1.1
Date: 29 Oct 84
Author: Paul A. Moore, Capt, USAF
Contained In File: VALNDEC.PAS

Name: chg_col
Aliases: None
Type Of Entry: Variable
Description: This is a column location of the entry to be changed in the selected matrix.
Make_up: Integer
Source:
Destination:
Used In: chgmat

Name: chg_row
Aliases: None
Type Of Entry: Variable

Description: This is a row location of the entry to be changed
in the selected matrix.

Make_up: Integer

Source:

Destination:

Used In: chgmat

Name: chgmat

Type: Procedure

Description: This procedure will display the requested matrix
on the screen and ask the user which row and col
location should be modified and will store the result
in the original location.

Global Variables Used: abort_command, cmdbuffer

Global Variables Changed: None

Global Constants Used: as_assigned, crt_only

Passed Variables: cmdbuffer, wordnumber

Returned: None

Files Read: MATRIX.DAT

Files Written: MATRIX.DAT

Aliases: None

Procedures Called: trim, gotoxy, disp_msg, out_string,
clear_msg, out_real, clear, pause,
get_r_num, get_strng, ucase, disp_matrx,
get_int, make_pretty_small_matrix,
make_pretty_large_matrix_one,
make_pretty_large_matrix_two

Called By: modify

Version: 1.0

Date: 22 Sep 85

Author: Susan K. Mashiko, Capt, USAF

Gary C. Tarczynski, Capt, USAF

Contained In File: MODIFY.PAS

Name: choice

Aliases: None

Type Of Entry: Variable

Description: This is a object that is passed to the procedure.

Make_up: cmdword

Source:

Destination:

Used In: chgmat, inroot, delroot,
poly_into_storage, form, disp_matrx,
poly_from_storage, disppoly

Name: chr1
Aliases: None
Type Of Entry: Data flow
Description: This variable indicates to get_strng the lower bound of ASCII characters that can be accepted on input.

Make_up: Char
Source: get_strng
Destination: N/A
Used In: get_strng

Name: chr2
Aliases: None
Type Of Entry: Data flow
Description: This variable indicates to get_strng the upper bound of ASCII characters that can be accepted on input.

Make_up: Char
Source: get_strng
Destination: N/A
Used In: get_strng

Name: ck_chr
Type: Procedure
Description: This procedure checks each character input to see if it is a delete or a backspace. If it is the screen is updated appropriately and the destination string is changed.

Global Variables Used: strng
Global Variables Changed: strng
Global Constants Used: del, backspace
Passed Variables: ch, strng
Returned: strng
Files Read: None
Files Written: None
Aliases: None
Procedures Called: del_lst_ch
Called By:

Version: 1.2
Date: 18 Oct 83
Author: Vincent M. Parisi II, Capt, USAF

Contained In File: GETINT.PAS

Name: clear

Type: Procedure

Description: This procedure clears the screen and homes the cursor
If the status line is on the status line is displayed.

Global Variables Used: term, stat_on, stat_line

Global Variables Changed: None

Global Constants Used: None

Passed Variables: None

Returned: None

Files Read: None

Files Written: None

Aliases: None

Procedures Called: gotoxy

Called By: title_slide, get_data, disp_msg,
clear_msg, get_cmd, recover,
update, ccopyy, help,
gettf, get_matrix_entries,
matrxadd, get_mat, disp,
disptf, modify, chgmat,
matrxinv, inroot, delroot,
form, disppoly, disp_matrx,
ppoly, mmatrix, polymlt,
define, get_poly, mmatrixmlt

Version: 2.0

Date: 21 Oct 83

Author: Vincent M. Parisi II, Capt, USAF

Contained In File: TERMINAL.PAS

Name: clear_msg

Type: Procedure

Description: The procedure clears the message indicated by the
msg_num from the screen. It is the programmer's
responsibility to position the cursor prior to calling
this routine. The cursor should be placed at the
beginning of the line where you wish the message
erased.

Global Variables Used: msg_dir, blanks

Global Variables Changed: None

Global Constants Used: crt_only

Passed Variables: msg_num

Returned: None

Files Read: None

Files Written: None
 Aliases: None
 Procedures Called: clear, out_string
 Called By: disp_msg, proces_error, recover,
 update, get_real, get_fact,
 roots, gettf, get_mat,
 mmatrxmlt, chgmat, delroot,
 form, get_poly_name, get_matrx_name

Version: 1.3
 Date: 18 Oct 83
 Author: Vincent M. Parisi II, Capt, USAF
 Contained In File: MSG.PAS

 Name: clearscreen
 Type: Procedure
 Description: This procedure clears the screen and homes the cursor
 If the status line is on the status line is displayed.
 Global Variables Used: term, stat_on, stat_line
 Global Variables Changed: None
 Global Constants Used: term_length, stat_line_width
 Passed Variables: None
 Returned: None
 Files Read: None
 Files Written: None
 Aliases: None
 Procedures Called: None
 Called By: icecappc

Version: 1.0
 Date: 12 Dec 84
 Author: Paul A. Moore, Capt, USAF
 Contained In File: TERMINAL.PAS

 Name: cmat
 Aliases: None
 Type Of Entry: Variable
 Description: This is a matrix.
 Make_up: matrix
 Source:
 Destination:
 Used In: matrxadd, matrxsub, mmatrxmlt

Name: cmd
 Aliases: None
 Type Of Entry: Variable
 Description: This is a string of char.
 Make_up: cmdword
 Source:
 Destination:
 Used In: readcom, val_n_dec

 Name: cmdbuffer
 Aliases: None
 Type Of Entry: Global variable and data flow
 Description: This is a buffer of individual commandwords input by the user.
 Make_up: buffer
 Source: Declared in icecappc
 Destination: N/A
 Used In: icecappc, get_cmd, displa_commandword,
 val_n_dec, readcom, proces_error,
 ccopy, help, disp,
 inroot, modify, delroot,
 select_routine, chgmat, ppoly,
 define, get_poly_name

 Name: cmd_col
 Aliases: None
 Type Of Entry: Constant and data flow
 Description: This is the column that is used as the beginning of user entered commands.
 Make_up: Integer
 Source: get_cmd
 Destination: N/A
 Used In: get_cmd

 Name: cmd_len
 Aliases: None
 Type Of Entry: Variable
 Description: This variable is the length of the command.
 Make_up: Integer
 Source:
 Destination:
 Used In: check_word

```

*****
Name:          cmd_row
Aliases:      None
Type Of Entry: Constant
Description:   This constant is the row that the cursor is
               positioned at for user entered commands.
Make_up:     Integer
Source:      get_cmd
Destination: N/A
Used In:     get_cmd
*****

```

```

*****
Name:          cmdword
Aliases:      None
Type Of Entry: Global type definition
Description:   This is the type definition of a short string
               which can then be used as a parameter for passing
               between procedures.
Make_up:     string[ wordsize ]
Source:      Declared in icecappc
Destination: N/A
Used In:     icecappc,      check_word,      val_n_dec,
               trim,      displa_commandword, get_cmd,
               get_location, ccopyy,      poly,
               help,      gettf,      polymanip,
               polymanip2, get_mat,      select_routine,
               disppoly,  modify,      disp_matrx,
               poly_into_storage, matrxmanip2, get_poly,
               poly_from_storage, define
*****

```

```

*****
Name:          cmd_word
Aliases:      None
Type Of Entry: Variable
Description:   This variable is equated to the cmdbuffer[word_num]
Make_up:     cmdword
Source:      displa_commandword
Destination: N/A
Used In:     displa_commandword
*****

```

```

*****
Name:          cnpol
Aliases:      None
Type Of Entry: Variable
Description:   This is a polynomial, which is used as a temporary
               storage area for the CLTF numerator polynomial.
*****

```

Make_up: polynomial
Source:
Destination:
Used In: form

Name: coeff_poly
Aliases: None
Type Of Entry: variable
Description: The variable is a temporary storage area for the polynomial.

Make_up: array[1..maxdegl] of real

Source:

Destination:

Used In: roots

Name: col
Aliases: None
Type Of Entry: Variable
Description: This variable is the screen column portion of the cursor position information.

make_up: Integer

Source:

Destination:

Used In: prompt_cmd, get_r_num, make_pretty_large_matrix_one,
disp_matrx, chgmat, get_poly_name,
make_pretty_small_matrix, get_matrx_entries,
gotoxy

Name: col_element
Aliases: None
Type Of Entry: Variable
Description: This variable is a counter for the display and modification of a matrix.

Make_up: Integer

Source:

Destination:

Used In: disp_matrx, chgmat, get_matrx_entries

Name: column
Aliases: None
Type Of Entry: Variable

Description: This variable is the column location for the left side of the rectangle.
Make_up: Integer
Source:
Destination:
Used In: rectangle

Name: column_length
Aliases: None
Type Of Entry: Variable
Description: This variable is a length of the vertical column of the matrix bracket.
Make_up: Integer
Source:
Destination:
Used In: left_bracket, right_bracket

Name: column_location
Aliases: None
Type Of Entry: Variable
Description: This variable is a column location of the vertical column of the right matrix bracket.
Make_up: Integer
Source:
Destination:
Used In: right_bracket

Name: command
Aliases: None
Type Of Entry: Variable
Description: This variable is the command word entered by the user.
Make_up: cmdword
Source: check_word
Destination: N/A
Used In: check_word

Name: complex
Aliases: None
Type Of Entry: Type definition
Description: Record type that contains each element of the

factored form of a polynomial.
Make_up: complex = record
 realpart : real;
 imagpart : real;
 end;
Source: declared in concons
Destination:
Used In: polynomial

Name: copy
Aliases: None
Type Of Entry: Variable
Description: This variable is a storage area.
Make_up: File
Source:
Destination:
Used In: select_routine

Name: copy_msg
Aliases: None
Type Of Entry: Constant
Description: This constant is the message number of the main
 menu copy message.
Make_up: Integer
Source:
Destination:
Used In: help

Name: COPY.PAS
Type: File
Description: This file manages the copy function. It gets the
 source and destination location then performs the
 copy operation for transfer functions, polynomials,
 and matrices.
Procedures Contained: get_location, move_tf, move_poly,
 move_matrix, gettf
Version: 3.0
Date: 22 Sep 85
Author: Vincent M. Parisi II, Capt, USAF
Modified by: Susan K. Mashiko, Capt, USAF
 Gary C. Tarczynski, Capt, USAF

```

*****
Name:      counter
Aliases:   None
Type Of Entry: Variable
Description: This is a the counter that is set up internal to
              the spolymlt procedure.
Make_up:   integer
Source:
Destination:
Used In:    spolymlt
*****

```

```

*****
Name:      cpoly
Aliases:   None
Type Of Entry: Variable
Description: This is a polynomial.
Make_up:   polynomial
Source:
Destination:
Used In:    polymlt,          polysub,          polyadd
*****

```

```

*****
Name:      crt
Aliases:   None
Type Of Entry: Global variable
Description: This variable is the flag that indicates whether
              the output should goto the CRT, CRT=true goes to
              CRT.
Make_up:   Boolean
Source:    Declared in icecappc
Destination:
Used In:    icecappc,          out_int,          get_data,
              out_real
*****

```

```

*****
Name:      crt_only
Aliases:   None
Type Of Entry: Global constant
Description: Literal that indicates that the input is to come only
              from the terminal ( usually a CRT/ keyboard )
Make_up:   Character literal
Source:    Declared in icecappc
Destination: N/A
Used In:    pause,          clear_msg,          prompt_help,
              prompt_cmd, get_r_num,          recover,
              update,      make_pretty,        gettf,

```


get_poly, get_poly_name, form,
ppoly, get_mat, inroot,
make_pretty_large_matrix_one, delroot,
make_pretty_small_matrix

Name: data
Aliases: None
Type Of Entry: Global type definition
Description: Type definition of the file structure that contains
the program parameters, and initialization values,
terminal and printer control codes, and command syntax
data structure as well as the message directory.

Make_up: data = record
param : array[1..num_param_group] of param_group;
term : array[1..term_length] of byte;
printr : array[1..printer_length] of byte;
msg_dir : array[1..num_msg_dir] of msg;
decode_dict : dict_buffer;
end;

Source: Declared in msdwtype

Destination:

Used In:

Name: data_file
Aliases: None
Type Of Entry: Variable
Description: Type file of type data that contains program
parameters.

Make_up: File of type data

Source:

Destination:

Used In: get_data

Name: dataptr
Aliases: None
Type Of Entry: Variable
Description: This is a pointer.

Make_up: ^datarecord

Source:

Destination:

Used In: get_data

```

*****
Name:          datarecord
Aliases:       None
Type Of Entry: Type definition
Description:   A record definition comprised of data.
Make_up:      datarecord = record
                tdata    :  data;
                end;

```

```

Source:
Destination:
Used In:      get_data
*****

```

```

*****
Name:          data_recs
Aliases:       None
Type Of Entry: Pointer
Description:   A pointer that points to a record containing a
                record so the initialization data can be disposed of
                after it is transferred to global storage areas.

```

```

Make_up:      dataptr
Source:
Destination:
Used In:      get_data
*****

```

```

*****
Name:          decode
Aliases:       None
Type Of Entry: Data flow
Description:   This data flow is one line of decoding information
                which is used to validate and decode a command
                line input by the user. It is constructed in the
                process called get_line.

```

```

Make_up:      Type dictionary
Source:       Declared in icecappc
Destination:
Used In:      get_line,      prompt_help,      check_word
*****

```

```

*****
Name:          decode_dict
Aliases:       None
Type Of Entry: Global file variable and data flow
Description:   The command syntax data structure which contains all
                information for decoding a command and determine if
                it is valid. Record element of type data.

```

```

Make_up:      dict_buffer
Source:       Declared in icecappc

```

Destination:
Used In: icecappc, get_data

Name: define
Type: Procedure
Description: This procedure will input a polynomial in either
factored or polynomial form.
Global Variables Used: strng, abort_command
Global Variables Changed: strng
Global Constants Used: None
Passed Variables: cmdbuffer, wordnumber
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: clear, pause, disp_msg, gettf,
getmat, get_poly, trim
Called By: select_routine

Version: 2.0
Date: 22 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: DEFINE.PAS

Name: DEFINE.PAS
Type: File
Description: This file contains the procedures that handle the
logic for the definition of various inputs: transfer
functions and polynomials.
Procedures Contained: get_poly, define
Version: 3.0
Date: 22 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

Name: define_msg
Aliases: None
Type Of Entry: Constant
Description: This constant is the number of the message for
the define option.
Make_up: Integer
Source:

Destination:
Used In: help

Name: def_obj
Aliases: None
Type Of Entry: Variable
Description: This variable is used to indicate the object that the called routine is to operate with.
Make_up: cmdword
Source:
Destination:
Used In: gettf, get_mat, define, get_poly

Name: degree
Aliases: None
Type Of Entry: Global Variable
Description: This variable is used to indicate the degree of a polynomial.
Make_up: Integer
Source: Declared in concons
Destination:
Used In: make_pretty, roots

Name: degreel
Aliases: None
Type Of Entry: Global Variable
Description: This variable is used to indicate the degree of a polynomial plus one.
Make_up: Integer
Source: Declared in concons
Destination:
Used In:

Name: del
Aliases: None
Type Of Entry: Global constant
Description: The decimal number that represents the ASCII value for the delete character.
Make_up: Integer
Source: Declared in icecappc
Destination: N/A

Used In: ck_chr, get_string

Name: del_lst_ch
Type: Procedure
Description: This procedure deletes the last character from the CRT

Global Variables Used: None
Global Variables Changed: None
Global Constants Used: backspace
Passed Variables: None
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called:
Called By: ck_chr

Version: 1.0
Date: 18 Oct 83
Author: Vincent M Parisi II, Capt, USAF
Contained In File: GETINT.PAS

Name: delroot
Type: Procedure
Description: This procedure will delete a root from a polynomial. If the root is complex the conjugate will also be removed.

Global Variables Used: blanks, cmdbuffer, abort_command
Global Variables Changed: None
Global Constants Used: crt_only, as_assigned
Passed Variables: cmdbuffer, wordnumber
Returned: None
Files Read: TF&POLS.DAT
Files Written: TF&POLS.DAT
Aliases: None
Procedures Called: clear, gotoxy, disppoly, trim,
highlight, out_string, nohighlight, get_int,
disp_msg, pause, clear_msg, form_poly
Called By: modify

Version: 2.0
Date: 22 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: DELROOT.PAS

Name: DELROOT.PAS
Type: File
Description: This file will delete a root from a polynomial.
If the root is complex the conjugate will also be removed.

Procedures Contained: delroot
Version: 2.0
Date: 22 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

Name: delu
Aliases: None
Type Of Entry: Variable
Description: This variable is used as the delta value of the u variable.

Make_up: Real
Source:
Destination:
Used In: roots

Name: delv
Aliases: None
Type Of Entry: Variable
Description: This variable is used as the delta value of the v variable.

Make_up: Real
Source:
Destination:
Used In: roots

Name: denom
Aliases: None
Type Of Entry: Variable
Description: This variable is used when roots determines if the denominator of the polynomial is zero.

Make_up: Real
Source:
Destination:
Used In: roots

Name: denom_deg
Aliases: None
Type Of Entry: Variable
Description: This variable is the degree of the transfer func-
tions denominator.
Make_up: Integer
Source:
Destination:
Used In: gettf

Name: denominator
Aliases: None
Type Of Entry: Variable
Description: This variable is the polynomial that is the denom-
inator of the transfer function.
Make_up: Polynomial
Source:
Destination:
Used In: gettf, disptf

Name: dest
Aliases: None
Type Of Entry: variable
Description: This variable determines where the output will go.
c - crt
p - printer
b - list_dev
a - as_assigned
Make_up: Char
Source: Declared in out_string
Destination: N/A
Used In: out_string, out_int, out_real

Name: destination
Aliases: None
Type Of Entry: Variable
Description: This variable is the stor_loc number for the
destination.
Make_up: Integer
Source:

Destination:
Used In: ccopy

Name: dest_loc
Aliases: None
Type Of Entry: Variable
Description: This variable is the stor_loc number for the destination.
Make_up: Integer
Source:
Destination:
Used In: move_tf, move_poly, move_matrix, ccopy

Name: dict_buffer
Aliases: None
Type Of Entry: Global type definition
Description: Type definition of the structure that contains the command syntax data that is read from disk as part of the structure data.
Make_up: dict_buffer = record
ptrs : array[1..num_ptrs] of ptr_recs;
words : array[1..num_words] of string[wordlength];
abbrev : array[1..num_words] of integer;
end;
Source: Declared in msdwtype
Destination:
Used In: icecappc, msdwtype, get_line, get_data

Name: dictionary
Aliases: None
Type Of Entry: Global type definition
Description: This is the type definition of an abstract record which makes up a data flow.
Make_up: dictionary = record
dictword : string[wordsize];
matchp : integer;
nomatchp : integer;
end;
Source: Declared in icecappc
Destination: N/A
Used In: icecappc, check_word, get_line

Name: dimension
Aliases: None
Type Of Entry: Variable
Description: This variable is the dimension of the matrix.
Make_up: Integer
Source:
Destination:
Used In: matrxinv

Name: disp
Type: Procedure
Description: This procedure contains the logic to display the
selected DISPLAY option on the screen.
Global Variables Used: cmdbuffer
Global Variables Changed: None
Global Constants Used: None
Passed Variables: cmdbuffer, wordnumber
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: disptf, trim, ppoly, clear,
mmatrix, disp_msg, pause
Called By: select_routine

Version: 1.0
Date: 4 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: DISP.PAS

Name: displa_commandword
Type: Procedure
Description: This procedure displays the commandword pointed to
by word_num.
Global Variables Used: cmdbuffer
Global Variables Changed: None
Global Constants Used: wordsize, buffersize
Passed Variables: cmdbuffer, word_num
Returned: None
Files Read: None
Files Written: None

Aliases: None
Procedures Called: out_string, trim
Called By: get_cmd, proces_error

Version: 1.1
Date: 30 Jun 84
Author: Vincent M Parisi II, Capt, USAF
Modifier: Paul A Moore, Capt, USAF
Contained In File: DISPLAYC.PAS

Name: DISPLAYC.PAS
Type: File
Description: This file contains the procedure to display the
commandword pointed to by word_num.
Procedures Contained: displa_commandword

Version: 1.1
Date: 30 Jun 84
Author: Vincent M Parisi II, Capt, USAF

Name: display_msg
Aliases: None
Type Of Entry: Variable
Description: This variable is the number of the help message for
the display function.

Make_up: Integer

Source:

Destination:

Used In: help

Name: display_word
Aliases: None
Type Of Entry: Variable
Description: This variable is the command word that is displayed
to the user.

Make_up: msg_line

Source:

Destination:

Used In: prompt_help

Name: disp_line
Type: Procedure

Description: This procedure reads one line of text from the file
MSG.DAT and displays it on the assigned device.

Global Variables Used: msg_txt
Global Variables Changed: None
Global Constants Used: as_assigned, screenwidth
Passed Variables: rec_num
Returned: None
Files Read: HELP.SYS
Files Written: None
Aliases: None
Procedures Called: out_string
Called By: disp_msg

Version: 1.2
Date: 18 Oct 83
Author: Vincent M Parisi II, Capt, USAF
Contained In File: MSG.PAS

Name: disp_matrx
Type: Procedure
Description: This procedure displays the matrix from a record in
'matrix.dat' The user should place a pause in his
code to keep the display on the screen.

Global Variables Used: matrix
Global Variables Changed: None
Global Constants Used: as_assigned
Passed Variables: choice
Returned: choice
Files Read: MATRIX.DAT
Files Written: None
Aliases: None
Procedures Called: clear, gotoxy, out_string, disp_msg,
make_pretty_small_matrix, out_real, trim,
make_pretty_large_matrix_one, pause,
make_pretty_large_matrix_two
Called By: matrxmanip1, matrxmanip2, mmatrix

Version: 2.0
Date: 20 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: MATRIX.PAS

Name: disp_msg
Type: Procedure

Description: This procedure displays the message pointed to by the parameter passed in, msg_num. The message is displayed at the current cursor position. If the message length is longer than 23 lines the display stops after showing 22 lines and waits for the user to enter a <CR>. If a '\$' is entered the procedure is exited and returns to calling procedure.

Global Variables Used: msg_dir

Global Variables Changed: None

Global Constants Used: None

Passed Variables: msg_num

Returned: None

Files Read: None

Files Written: None

Aliases: None

Procedures Called: disp_line, gotoxy, clear,

Called By: proc_err, recover, update,
make_pretty, ccopy, help,
get_real, get_fact, roots,
gettf, define, mmatrix,
mmatrixmlt, getmat, matrixadd,
modify, matrixinv, disptf,
delroot, form, disppoly,
ppoly, get_poly, disp_matrix,
disp, get_poly_name, get_matrix_name

Version: 3.1

Date: 23 Aug 85

Author: Vincent M Parisi II, Capt, USAF

Modified by: Susan K. Mashiko, Capt, USAF

Gary C. Tarczynski, Capt, USAF

Contained In File: MSG.PAS

Name: disp_obj

Aliases: None

Type Of Entry: Variable

Description: This variable is used to indicate the object that the called routine is to operate with.

Make_up: cmdword

Source:

Destination:

Used In: gettf

Name: DISP.PAS

Type: File
Description: This file contains the logic to display the selected DISPLAY option on the screen.
Procedures Contained: disp, disptf
Version: 1.0
Date: 4 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

Name: disppoly
Type: Procedure
Description: This procedure displays the polynomial form a record in 'tf&pols.dat' The user should place a pause in his code to keep the display on the screen.
Global Variables Used: polynomial
Global Variables Changed: None
Global Constants Used: as_assigned
Passed Variables: choice
Returned: None
Files Read: TF&POLS.DAT
Files Written: None
Aliases: None
Procedures Called: clear, gotoxy, out_string, disp_msg, make_pretty, out_real, trim
Called By: polymanip, polymanip2, ppoly, inroot, delroot, get_poly

Version: 1.0
Date: 6 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: POLY.PAS

Name: disptf
Type: Procedure
Description: This procedure displays the transfer function from a record in 'tf&pols.dat'
Global Variables Used: None
Global Variables Changed: None
Global Constants Used: as_assigned
Passed Variables: disp_obj
Returned: None
Files Read: TF&POLS.DAT
Files Written: None
Aliases: None

Procedures Called: clear, gotoxy, out_string, disp_msg,
make_pretty, out_real, trim, pause
Called By: disp

Version: 2.0
Date: 25 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: DISP.PAS

Name: disp_row
Aliases: None
Type Of Entry: Variable
Description: This variable is the row offset that the display
of the numerator, denominator, or polynomial should
be started on.

Make_up: Integer
Source:
Destination:
Used In: gettf, poly, get_poly

Name: divisor
Aliases: None
Type Of Entry: Variable
Description: This variable is the divisor used in the calculation
of the inverse of a matrix.

Make_up: Real
Source:
Destination:
Used In: matrxinv

Name: dlen
Aliases: None
Type Of Entry: Variable
Description: This variable is the length of the dictionary.
Make_up: Integer
Source: get_line
Destination:
Used In: get_line

Name: d_len

Aliases: None
Type Of Entry: Variable
Description: This variable is the length of dword
Make_up: Integer
Source:
Destination:
Used In: check_word

Name: DONEWORD
Aliases: None
Type Of Entry: Global constant
Description:
Make_up: Character
Source: Declared in msdwcons
Destination:
Used In: prompt_help, val_n_dec

Name: dword
Aliases: None
Type Of Entry: Variable
Description: This variable is the dictionary word the user input
is compared with.
Make_up: cmdword
Source:
Destination:
Used In: check_word

Name: ENDCODE
Aliases: None
Type Of Entry: Global constant
Description: This unique number indicates that the end has been
reached in the trace through the command syntax
structure. Indicates to prompt_help that no more
words are to be displayed as the end of valid objects
for the previously entered command words have been
reached.

Make_up: Integer
Source: Declared in msdwcons
Destination:
Used In: prompt_help, val_n_dec

Name: epsi
Aliases: None
Type Of Entry: Variable
Description: This number is used for comparison in roots. Once the delta value is less than epsi then the converged root is considered valid.

Make_up: Real
Source:
Destination:
Used In: roots

Name: error_code
Aliases: None
Type Of Entry: Data flow
Description: This indicates which problem occurred in the command decoding process of val_n_dec. The user never sees these codes per se, only the error messages generated in response to them.

Make_up: Char
A or a - no error, entry value used as control for the the while statement.
N or n - exit value when successful validation has been accomplished.
B or b - word does not exist in dictionary.
C or c - Indicates too many words in command.
D or d - Indicates the command is incomplete.

Source: get_cmd, val_n_dec
Destination:
Used In: get_cmd, val_n_dec, proces_error

Name: ff
Aliases: None
Type Of Entry: Global constant
Description: This is the decimal value for the form feed character.

Make_up: Integer
Source: Declared in msdwcons
Destination:
Used In:

Name: field
Aliases: None
Type Of Entry: Data flow

Description: This parameter specifies the width for numerical output. For example: if it is a 5 and the number that is output is a 12 then the number 12 will be right justified in five space field.

Make_up: Integer

Source:

Destination:

Used In: out_int

Name: fieldwidth

Aliases: None

Type Of Entry: Variable

Description: This number is passed to outstring to indicate how long the number to be output is.

Make_up: Integer

Source:

Destination:

Used In: out_real

Name: first

Aliases: None

Type Of Entry: Variable

Description: This variable is used to pass a the name of a polynomial or transfer function.

Make_up: cmdword

Source:

Destination:

Used In: mmatrix, ppoly, polmanip, polmanip2, matrxmanip2

Name: form

Type: Procedure

Description: This procedure will form OLTF's and CLTF's

Global Variables Used: abort_command

Global Variables Changed: None

Global Constants Used: crt_only

Passed Variables: None

Returned: None

Files Read: None

Files Written: None

Aliases: None

Procedures Called: clear, gotoxy, disp_msg, out_string, get_int, polymt, pause, clear_msg,

poly_from_storage, poly_into_storage,
highlight, nohighlight, spolymlt,
polyadd, disptf
Called By: select_routine

Version: 1.0
Date: 7 Aug 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: FORM.PAS

Name: form_msg
Aliases: None
Type Of Entry: Variable
Description: This number is number of the message for help
for the form command.
Make_up: Integer
Source:
Destination:
Used In: help

Name: form_poly
Type: Procedure
Description: This procedure will form a polynomial from the
factored form.
Global Variables Used: None
Global Variables Changed: None
Global Constants Used: maxdegl
Passed Variables: poly
Returned: poly
Files Read: None
Files Written: None
Aliases: None
Procedures Called: None
Called By: poly, inroot, delroot

Version: 1.0
Date: 26 Aug 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: GETTF.PAS

Name: FORM.PAS

Type: File
Description: This file will form OLTF's and CLTF's
Procedure contained: poly_from_storage, poly_into_storage,
form

Version: 1.0
Date: 7 Aug 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

Name: gain
Aliases: None
Type Of Entry: Variable
Description: This variable is the storage area for the gain of
the transfer function.

Make_up: Real

Source:

Destination:

Used In: form

Name: gdpoly
Aliases: None
Type Of Entry: Variable
Description: This variable is a polynomial, the denominator of
the feedforward transfer function.

Make_up: polynomial

Source:

Destination:

Used In: form

Name: getchi
Type: Function
Description: This function gets one character from a string and
returns it to the read function for conversion.

Global Variables Used: strng

Global Variables Changed: strng

Global Constants Used: None

Passed Variables: None

Returned: char

Files Read: None

Files Written: None

Aliases: None

Procedures Called: None

Called By:

Version: 1.0
Date: 18 Oct 83
Author: Vincent M Parisi II, Capt, USAF
Contained In File: GETINT.PAS

Name: get_cmd
Type: Procedure
Description: This procedure handles all processing associated with getting a valid command from the user. It is called by the program and operation is maintained here until a decoded and validated command is entered.
Global Variables Used: help_level, cmdbuffer, call_routine, abort_command
Global Variables Changed: abort_command
Global Constants Used: yes
Passed Variables: cmdbuffer, call_routine, num_of_commands
Returned: num_of_commands
Files Read: None
Files Written: None
Aliases: None
Procedures Called: gotoxy, readcom, get_line, val_n_dec, prompt_help, displa_commandword, prompt_cmd, instruction, proces_error, clear
Called By: icecappc

Version: 3.1
Date: 16 Aug 83
Author: Vincent M Parisi II, Capt, USAF
Contained In File: GETCOM.PAS

Name: GETCOM.PAS
Type: File
Description: This file contains the procedures which handle all processing associated with getting a valid command from the user. It is called by the program and operation is maintained here until a decoded and validated command is entered.

Procedures Contained: get_cmd
Version: 3.1
Date: 16 Aug 83
Author: Vincent M Parisi II, Capt, USAF

Name: get_data
 Type: Procedure
 Description: This procedure reads the data.dat file and initializes the program variables passed to it.
 Global Variables Used: blanks, call_routine, status_line, msg_dir, decode_dict, printer, trans, temp, crt, show_abbreviation, in_terminal, stat_on, macro_error, help_level, list_dev_name, trans_file_name, macro_file_name
 Global Variables Changed: same as global variables
 Global Constants Used: term_length, screen_width, num_words, printer_length, num_msg_dir, num_ptrs
 Passed Variables: term_dat, print_dat, msg_dir, printer, decode_dict, trans, temp, crt, stat_on, show_abbreviation, in_terminal, macro_error, help_level, list_dev_name, trans_file_name, macro_file_name
 Returned: all passed variables are changed except term_dat and print_dat
 Files Written: printer.out, transaction, macro.inp, temp.out
 Files Read: help.sys, microwsdw.sys
 Aliases: None
 Procedures Called: clear, title_slide, bld_stat_line,
 Called By: icecappc

Version: 4.0
 Date: 22 Jul 85
 Author: Vincent M Parisi II, Capt, USAF
 Modified by: Susan K. Mashiko, Capt, USAF
 Gary C. Tarczynski, Capt, USAF

Contained In File: GETDAT.PAS

Name: GETDAT.PAS
 Type: File
 Description: This file reads the data file from the disk and inserts it into the appropriate tables in the main program. It also initializes the flags and help_level and displays the titleslide.

Procedures Contained: title_slide, bld_stat_line, get_data

Version: 4.0
 Date: 22 Jul 85
 Author: Vincent M Parisi II, Capt, USAF
 Modified by: Susan K. Mashiko, Capt, USAF
 Gary C. Tarczynski, Capt, USAF

Name: get_fact
Type: Procedure
Description: This procedure gets the factored form of the polynomial.
Global Variables Used: abort_command
Global Variables Changed: None
Global Constants Used: as_assigned
Passed Variables: poly, row, abort_command
Returned: poly
Files Read: None
Files Written: None
Aliases: None
Procedures Called: make_pretty, get_r_num, gotoxy,
out_real, disp_msg, pause,
clear_msg
Called By: poly

Version: 2.2
Date: 8 Sep 85
Author: Vincent M Parisi II, Capt, USAF
Modified by: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: GETTF.PAS

Name: GETINT.PAS
Type: File
Description: This file handles the input of integers. Normal program integer input does not have edit capabilities to exclude inputs such as letters. This procedure only accepts valid input.
Procedures Contained: del_lst_ch, ck_chr, out_int,
get_int, get_chi

Version: 1.3
Date: 18 Oct 83
Author: Vincent M Parisi II, Capt, USAF

Name: get_int
Type: Procedure
Description: This procedure handles the input of integers. Normal program integer input does not have edit capabilities to exclude inputs such as letters. This procedure only accepts valid input.

Global Variables Used: strng, abort_command
Global Variables Changed: strng, abort_command
Global Constants Used: as_assigned
Passed Variables: number, abort_command
Returned: number, abort_command
Files Read: None
Files Written: None
Aliases: None
Procedures Called: get_strng
Called By: gettf, getmat, chgmat, delroot, form,
get_poly

Version: 1.3
Date: 18 Oct 83
Author: Vincent M Parisi II, Capt, USAF
Contained In File: GETINT.PAS

Name: GETLINE.PAS
Type: File

Description: This procedure builds a decoded entry from the record pointed to on entry. The pointers come from the ptrs part of dict_buffer and the word comes from the words part of the dict_buffer.

Procedures Contained: get_line
Version: 3.0
Date: 17 Jul 83
Author: Vincent M Parisi II, Capt, USAF

Name: get_line
Type: Procedure

Description: This procedure builds a decoded entry from the record pointed to on entry. The pointers come from the ptrs part of dict_buffer and the word comes from the words part of the dict_buffer.

Global Variables Used: blanks
Global Variables Changed: None
Global Constants Used: wordsize, word_length
Passed Variables: decode, rec_num
Returned: None
Files Read: dict_file
Files Written: None
Aliases: None
Procedures Called: None
Called By: get_cmd, prompt_help, val_n_dec

Version: 3.0
Date: 17 Jul 83
Author: Vincent M Parisi II, Capt, USAF
Contained In File: GETLINE.PAS

Name: get_location
Type: Procedure
Description: This procedure determines the record location of the source and destination objects for the copy function.
Global Variables Used: None
Global Variables Changed: None
Global Constants Used: None
Passed Variables: location, rec_loc, type_move
Returned: rec_loc, type_move
Files Read: None
Files Written: None
Aliases: None
Procedures Called: None
Called By: ccopy

Version: 2.0
Date: 4 Sep 85
Author: Vincent M. Parisi II, Capt, USAF
Modified by: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: COPY.PAS

Name: getmat
Type: Procedure
Description: This procedure will get a matrix and store it.
Global Variables Used: matrix, abort_command
Global Variables Changed: matrix
Global Constants Used: as_assigned, max_cols, max_rows, crt_only
Passed Variables: def_obj
Returned: None
Files Read: MATRIX.DAT
Files Written: None
Aliases: None
Procedures Called: out_string, pause, clear, gotoxy, get_int, get_matrix_entries, disp_msg, clear_msg, make_pretty_large_matrix_one, make_pretty_small_matrix
Called By: define

Version: 1.0
Date: 11 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: GETMAT.PAS

Name: get_matrix_entries
Type: Procedure
Description: This procedure will get a matrix entry.
Global Variables Used: abort_command
Global Variables Changed: None
Global Constants Used: None
Passed Variables: matrix, abort_command
Returned: matrix
Files Read: None
Files Written: None
Aliases: None
Procedures Called: get_r_num, pause, clear,
make_pretty_large_matrix_two
Called By: getmat

Version: 1.0
Date: 11 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: GETMAT.PAS

Name: get_matrx_name
Type: Procedure
Description: This procedure will get the name of a matrix from
the screen.
Global Variables Used: blanks, abort_command
Global Variables Changed: blanks
Global Constants Used: as_assigned, crt_only
Passed Variables: mat_name, row, col, abort_command
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: highlight, nohighlight, gotoxy,
out_string, ucase, clear_msg,
get_strng, disp_msg, pause
Called By: mmatrix

Version: 1.0

Date: 20 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: MMATRIX.PAS

Name: GETMAT.PAS
Type: File

Description: This file contains the procedure that will get a matrix and store it.

Procedures Contained: left_bracket, right_bracket, get_mat,
make_pretty_large_matrix_one,
make_pretty_large_matrix_two,
make_pretty_small_matrix, get_matrx_entries,

Version: 1.0

Date: 12 Sep 85

Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

Name: get_poly
Type: Procedure

Description: This procedure will get a polynomial in either the factored or the poly form.

Global Variables Used: abort_command

Global Variables Changed: None

Global Constants Used: as_assigned, crt_only

Passed Variables: def_obj, method

Returned: None

Files Read: None

Files Written: TF&POLS.DAT

Aliases: None

Procedures Called: clear, gotoxy, disp_msg, out_string,
get_int, disppoly, trim, pause

Called By: define

Version: 1.0

Date: 28 Sep 85

Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

Contained In File: DEFINE.PAS

Name: get_poly_name
Type: Procedure

Description: This procedure will get the name of a polynomial form

the screen.

Global Variables Used: blanks, abort_command
 Global Variables Changed: None
 Global Constants Used: as_assigned, crt_only
 Passed Variables: poly_name, row, col, abort_command
 Returned: poly_name
 Files Read: None
 Files Written: None
 Aliases: None
 Procedures Called: highlight, nohighlight, gotoxy, out_string,
 ucase, trim, pause,
 get_string, disp_msg, clear_msg
 Called By: ppoly

Version: 1.0
 Date: 6 Sep 85
 Author: Susan K. Mashiko, Capt, USAF
 Gary C. Tarczynski, Capt, USAF
 Contained In File: POLY.PAS

Name: get_real
 Type: Procedure
 Description: This procedure gets a real number from the user. The
 procedure checks the validity of the input number.
 This prevents in inadvertant exit to the operating
 system.

Global Variables Used: abort_command, string
 Global Variables Changed: abort_command, string
 Global Constants Used: as_assigned
 Passed Variables: number, abort_command
 Returned: number
 Files Read: None
 Files Written: None
 Aliases: None
 Procedures Called: get_string, gotoxy, pause,
 disp_msg, clear_msg, highlight,
 out_string, nohighlight
 Called By: get_r_num

Version: 1.0
 Date: 19 Aug 85
 Author: Susan K. Mashiko, Capt, USAF
 Gary C. Tarczynski, Capt, USAF
 Contained In File: REALS.PAS

Name: get_r_num
Type: Procedure
Description: This procedure gets a real number from the user. The procedure checks the validity of the input number. This prevents an inadvertent exit to the operating system. Additionally this procedure will provide a prompt on row 20, and will provide an error message if there is something wrong with the input. The user must pass the row and the col of the desired location of the real number.

Global Variables Used: blanks, abort_command
Global Variables Changed: blanks
Global Constants Used: as_assigned, crt_only
Passed Variables: number, abort_command, row, col
Returned: number
Files Read: None
Files Written: None
Aliases: None
Procedures Called: out_real, gotoxy,
highlight, get_real,
out_string, nohighlight,
Called By: get_fact, get_unfact, mmatrix,
get_matrx_entries, chgmat, inroot

Version: 1.2
Date: 20 Aug 85
Author: Vincent M. Parisi II, Capt, USAF
Modified by: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: GETTF.PAS

Name: GETSTRIN.PAS
Type: File
Description: This file gets the ASCII input from terminal keyboard or the macro command file as specified in the input parameter. Collects characters until a <CR>.

Procedures Contained: get_strng
Version: 2.0
Date: 28 Aug 83
Author: Vincent M Parisi II, Capt, USAF

Name: get_strng
Type: Procedure
Description: This procedure gets the ASCII input from terminal

keyboard or the macro command file as specified in the input parameter. Collects characters until a <CR>.

Global Variables Used: in_terminal, macro_file, strng, abort_command
Global Variables Changed: strng, abort_command
Global Constants Used: screen_width, abort_str
Passed Variables: strng, abort_command, in_dev, chr1, chr2
Returned: strng, abort_command
Files Read: None
Files Written: None
Aliases: None
Procedures Called: None
Called By: get_int, readcom, recover, update, get_real, chgmat, get_poly_name, get_matrx_name

Version: 2.0
Date: 28 Aug 83
Author: Vincent M Parisi II, Capt, USAF
Contained In File: GETSTRIN.PAS

Name: gettf
Type: Procedure
Description: This procedure gets the transfer function in either polynomial or factored form.

Global Variables Used: abort_command
Global Variables Changed: abort_command
Global Constants Used: crt_only, as_assigned, max_deg
Passed Variables: def_obj, method
Returned: None
Files Read: TF&POLS.DAT
Files Written: None
Aliases: None
Procedures Called: clear, gotoxy, disp_msg, out_string, get_int, clear_msg, trim, poly, disptf, pause
Called By: define

Version: 3.0
Date: 25 Sep 85
Author: Vincent M. Parisi II, Capt, USAF
Modified by: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: GETTF.PAS

Name: GETTF.PAS
Type: File
Description: This file gets the transfer function in either polynomial or factored form.
Procedures Contained: get_r_num, make_pretty, get_fact,
form_poly, roots, get_unfact,
poly, gettf

Version: 3.0
Date: 25 Sep 85
Author: Vincent M. Parisi II, Capt, USAF
Modified by: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

Name: get_unfact
Type: Procedure
Description: This procedure gets the unfactored form of the polynomial.
Global Variables Used: abort_command
Global Variables Changed: None
Global Constants Used: as_assigned
Passed Variables: poly, row, abort_command
Returned: poly
Files Read: None
Files Written: None
Aliases: None
Procedures Called: make_pretty, get_r_num, gotoxy,
out_real, disp_msg, pause,
clear_msg
Called By: poly

Version: 2.0
Date: 25 Sep 85
Author: Vincent M Parisi II, Capt, USAF
Modified by: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

Contained In File: GETTF.PAS

Name: gnpol
Aliases: None
Type Of Entry: Variable
Description: Variable is a polynomial, numerator of the feed

forward transfer function.
Make_up: Polynomial
Source:
Destination:
Used In: form

Name: gotoxy
Type: Procedure

Description: This procedure places the cursor at the x and y coordinates passed to it. It is capable of sending an initial string of characters either the row or column then an intermediate string of char, the other address and finally a trailing string of char if required. Offsets if any are added prior to sending the row/col

Global Variables Used: term
Global Variables Changed: None
Global Constants Used: term_length
Passed Variables: row : integer; col : integer
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: ttype
Called By:

clear,	rectangle,	pause,
title_slide,	disp_msg,	prompt_help,
prompt_cmd,	proc_error,	get_cmd,
get_real	get_r_num,	recover,
update,	ccopyy,	make_pretty,
roots,	gettf,	get_fact,
get_unfact,	left_bracket,	right_bracket,
getmat,	matrxadd,	mmatrxmlt,
matrxinv,	disptf,	chgmat,
disppoly,	inroot,	delroot,
form,	disp_matrx,	mmatrix,
polymt,	get_poly_name,	ppoly,
get_poly,	make_pretty_small_matrix,	
make_pretty_large_matrix_one,		
get_matrx_name		

Version: 2.0
Date: 21 Oct 83
Author: Vincent M Parisi II, Capt, USAF
Modifier: Paul A. Moore, Capt, USAF
Contained In File: TERMINAL.PAS

Name: graphics
Type: Procedure
Description: This procedure places the terminal in graphics mode.
Global Variables Used: term
Global Variables Changed: None
Global Constants Used: term_length
Passed Variables: None
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called:
Called By: rectangle, make_pretty, left_bracket,
right_bracket

Version: 2.0
Date: 21 Oct 83
Author: Vincent M Parisi II, Capt, USAF
Contained In File: TERMINAL.PAS

Name: hdpol
Aliases: None
Type Of Entry: Variable
Description: Variable is a polynomial, denominator of the feedback
transfer function.
Make_up: Polynomial
Source:
Destination:
Used In: form

Name: height
Aliases: None
Type Of Entry: Variable
Description: Variable is the desired height in rows of the
rectangle.
Make_up: Integer
Source:
Destination:
Used In: rectangle

Name: help
Type: Procedure

Description: This procedure handles the logic for providing on line help. The valid command is scanned to determine what help is requested. The display message routine is then called with the correct number of the message desired.

Global Variables Used: cmdbuffer
Global Variables Changed: None
Global Constants Used: wordsize, buffersize
Passed Variables: cmdbuffer, wordnumber
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: pause, clear, disp_msg, trim,
Called By: select_routine

Version: 2.0
Date: 18 Sep 85
Author: Vincent M Parisi II, Capt, USAF
Modified by: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: HELP.PAS

Name: help
Aliases: None
Type Of Entry: Char
Description: Variable that is used to insert in the status line, it indicates the help level of the system.
Make_up: Char
Source: bld_stat_line
Destination:
Used In: bld_stat_line

Name: help_level
Aliases: None
Type Of Entry: Global file variable and data flow
Description: Variable indicates the amount of help the user wants in the operation of the program.
Make_up: Integer
Source: Declared in icecappc
Destination:
Used In: icecappc, bld_stat_line, get_cmd, proces_error, get_data

```
*****
Name:      help_obj
Aliases:   None
Type Of Entry: Variable
Description: This variable is the object of the command help. It
            is passed to other procedures.
Make_up:   cmdword
Source:
Destination:
Used In:    help
*****
```

```
*****
Name:      HELP.PAS
Type:      File
Description: This file handles the logic for providing on
            line help. The valid command is scanned to determine
            what help is requested. The display message routine is
            then called with the correct number of the message
            desired.
Procedure Contained: help
Version:    2.0
Date:      18 Sep 85
Author:     Vincent M Parisi II, Capt, USAF
Modified by: Susan K. Mashiko, Capt, USAF
            Gary C. Tarczynski, Capt, USAF
*****
```

```
*****
Name:      HELP.SYS
Type :     Data
Description: This file is created by the program builddat, and is
            in turn used by the procedure get_data for initial-
            ization.
Used In:   get_data
*****
```

```
*****
Name:      highlight
Type:      Procedure
Description: This procedure places the terminal in reverse video
            mode.
Global Variables Used:   term
Global Variables Changed: None
Global Constants Used:  term_length
Passed Variables:      None
Returned:               None
Files Read:             None
Files Written:          None
*****
```

Aliases: None
 Procedures Called:
 Called By: highlight, title_slide, prompt_cmd
 proces_error, get_r_num, ccopyy,
 get_real, roots, delroot,
 polymlt, form, get_poly_name,
 get_matrx_name

Version: 2.0
 Date: 21 Oct 83
 Author: Vincent M Parisi II, Capt, USAF
 Contained In File: TERMINAL.PAS

Name: hnpol
 Aliases: None
 Type Of Entry: Variable
 Description: Variable is a polynomial, the numerator of the feed
 back transfer function.
 Make_up: Polynomial
 Source:
 Destination:
 Used In: form

Name: i
 Aliases: None
 Type Of Entry: Variable
 Description: This variable is used as a counter.
 Make_up: Integer
 Source: Same as the procedure that uses it
 Destination:
 Used In: ucase, gotoxy, graphics,
 clear, nographics, clearscreen,
 highlight, nohighlight, videolow,
 svideolow, videobold, svideobold,
 rectangle, disp_msg, clear_msg,
 trim, readcom, check_word,
 get_cmd, get_data, recover,
 ccopyy, make_pretty, update,
 move_tf, move_poly, move_matrix,
 get_real, gettf, get_fact,
 get_unfact, roots, left_bracket,
 matrxsub, right_bracket, get_matrx_entries,
 getmat, matrxadd, mmatrxmlt,
 polymanip, polymanip2, dispmatrx,
 matrxmanipl, polysub, smatrxmlt,

matrxmanip2, matrxtran, matrxinv,
disppoly, get_poly, disptf,
chgmt, inroot, delroot,
spolymt, polymt, polyadd,
make_pretty_small_matrix, select_routine,
make_pretty_large_matrix_one, proces_error

Name: icecappc
Type: Main program
Description: This file contains the main program for the MICROSDW men
system and the icecappc subroutines. There are four
different versions: IBM-PC, IBM with hard drive, Z-100,
Z-100 with hard drive.

Global Variables Used: None
Global Variables Changed: None
Global Constants Used: None
Passed Variables: None
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: get_cmd, get_data, select_routine
(IBM version - standard_output)
Called By: N/A

Version:
Date:
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: ICECAPPC.PAS

Name: imag_root
Aliases: None
Type Of Entry: Variable
Description: This variable is used as the temporary storage area
for the imaginary root.
Make_up: Real
Source:
Destination:
Used In: roots

Name: in_dev
Aliases: None

Type Of Entry: Variable and data flow
Description: This parameter indicates to get_strng which device
is to be used for input.
Make_up: char
Source: get_strng
Destination:
Used In: get_strng

Name: inroot
Type: Procedure
Description: This procedure will insert a root into a polynomial.
If the root is complex the conjugate will also be
inserted.

Global Variables Used: cmdbuffer, abort_command

Global Variables Changed: None

Global Constants Used: crt_only, as_assigned

Passed Variables: cmdbuffer, wordnumber

Returned: None

Files Read: TF&POLS.DAT

Files Written: TF&POLS.DAT

Aliases: None

Procedures Called: clear, gotoxy, disppoly, trim,
out_real, out_string, get_r_num, out_int,
form_poly, pause

Called By: modify

Version: 2.0

Date: 19 Sep 85

Author: Susan K. Mashiko, Capt, USAF

Gary C. Tarczynski, Capt, USAF

Contained In File: INROOT.PAS

Name: INROOT.PAS
Type: File
Description: This file will insert a root into a polynomial.
If the root is complex the conjugate will also be
inserted.

Procedures Contained: inroot

Version: 2.0

Date: 19 Sep 85

Author: Susan K. Mashiko, Capt, USAF

Gary C. Tarczynski, Capt, USAF

Name: instr_col
Aliases: None
Type Of Entry: Constant
Description: This constant is the column location where instructions for command entry begins.
Make_up: Integer
Source: get_cmd
Destination: instruction
Used In: get_cmd, instruction

Name: instr_row
Aliases: None
Type Of Entry: Constant
Description: This constant is the row location where instructions for command entry begins.
Make_up: Integer
Source: get_cmd
Destination: instruction
Used In: get_cmd, instruction

Name: instrstring
Aliases: None
Type Of Entry: Variable
Description: This variable is used as a string.
Make_up: msg_line
Source: ucase
Destination:
Used In: ucase, svideobold, svideolow

Name: instruction
Type: Procedure
Description: This procedure issues the appropriate instructions for entering a command based on the number of command words already entered.
Global Variables Used: None
Global Variables Changed: None
Global Constants Used: None
Passed Variables: level, instr_row, instr_col
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: out_string

Called By: get_cmd

Version: 1.1
Date: 16 Aug 83
Author: Vincent M Parisi II, Capt, USAF
Contained In File: INSTRUC.PAS

Name: INSTRUC.PAS
Type: File
Description: This file issues the appropriate instructions
for entering a command based on the number of
command words already entered.

Procedures Contained: instruction
Version: 1.1
Date: 16 Aug 83
Author: Vincent M Parisi II, Capt, USAF

Name: in_terminal
Aliases: None
Type Of Entry: Global variable and data flow
Description: This variable indicates whether the input should come
from the terminal keyboard or the macrofile.

Make_up: Boolean
Source: Declared in icecappc
Destination:
Used In: icecappc, get_strng, getdata

Name: iteration
Aliases: None
Type Of Entry: Variable
Description: This variable is the counter for the number of
iterations the procedure roots goes thru before the
root is found.

Make_up: Integer
Source:
Destination:
Used In: roots

Name: j
Aliases: None
Type Of Entry: Variable

Description: Generally set up as a counter.
Make_up: integer
Source:
Destination:
Used In: prompt_help, readcom, matrxadd, matrxsub,
get_matrix_entries, smatrxmlt, mmatrixmlt,
matrxinv, matrxinv, matrxtran, chgmat,
polymlt

Name: k
Aliases: None
Type Of Entry: Variable
Description: Generally set up as a counter.
Make_up: integer
Source:
Destination:
Used In: roots

Name: l
Aliases: None
Type Of Entry: Variable
Description: Used as a counter.
Make_up: Integer
Source:
Destination:
Used In: mmatrixmlt

Name: L1
Aliases: None
Type Of Entry: Variable
Description: line + height - 1
Make_up: integer
Source: rectangle
Destination:
Used In: rectangle

Name: last_rec_num
Aliases: None
Type Of Entry: Variable
Description: This variable is the number of the last record in
the file.

Make_up: Integer
Source:
Destination:
Used In: val n dec

Name: left_bracket
Type: Procedure
Description: This procedure draws the left bracket around a matrix
displayed on the terminal.
Global Variables Used: term
Global Variables Changed: None
Global Constants Used: None
Passed Variables: num_rows
Returned: None
Files Read: None
Files Written: None
Aliases: None
Porcedures called: graphics, gotoxy, nographics
Called by: make_pretty_small_matrix, make_pretty_large_matrix_one,
make_pretty_large_matrix_two

Version: 1.0
Date: 11 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: GETMAT.PAS

Name: lencmd
Aliases: None
Type Of Entry: Variable
Description: This variable is the length of the command line that
was input by the user.
Make_up: Integer
Source: readcom
Destination:
Used In: readcom

Name: length
Aliases: None
Type Of Entry: Variable
Description: This variable is the length of the message to be
displayed or erased.
Make_up: Integer

Source:
Destination:
Used In: disp_msg, clear_msg

Name: level
Aliases: None
Type Of Entry: Variable and data flow
Description: This variable indicates which iteration through the recursive val_n_dec procedure a fault occurred. Thus it indicates which word in the command buffer is causing problems and allows prompts based on that word/position.
Make_up: Integer
Source: get_cmd, val_n_dec
Destination:
Used In: get_cmd, instruction, val_n_dec
proc_error

Name: line
Aliases: None
Type of Entry: Variable
Description: This variable indicates which row should be the top of the rectangle.
Make_up: Integer
Source:
Destination:
Used In: rectangle

Name: list_dev
Aliases: None
Type Of Entry: Global file variable
Description: Logical file - when output is to this 'file' the output is written to a file 'PRINTER.OUT'
Make_up: TEXT
Source: Declared in icecappc
Destination:
Used In: get_data, out_string, out_int

Name: list_dev_name
Aliases: None
Type Of Entry: Global file variable

Description: This variable is not used in icecappc
Make_up:
Source: Declared in icecappc
Destination:
Used In: icecappc, get_data

Name: location
Aliases: None
Type Of Entry: Variable
Description: This variable is passed internal to the file Copy.
It is the object that is to be copied.
Make_up: cmdword
Source:
Destination:
Used In: get_location

Name: logo
Aliases: None
Type Of Entry: Constant
Description: This constant is word or phrase printed on the CRT
to prompt the user for a command.
Make_up: Char
Source:
Destination:
Used In: prompt_cmd

Name: m
Aliases: None
Type Of Entry: Variable
Description: Generally set up as a counter.
Make_up: integer
Source:
Destination:
Used In: roots

Name: macro_error
Aliases: None
Type Of Entry: Global variable
Description: This variable is the flag that indicates whether
an error has occurred while getting input from the
macro command file.

Make_up: Boolean
Source: Declared in icecappc
Destination:
Used In: icecappc, readcom, get_data

Name: macro_file
Aliases: None
Type Of Entry: Global variable
Description: File of text that contains the commands and data
input as macro commands for non-interactive program
use.

Make_up: Text
Source: Declared in icecappc
Destination:
Used In: get_data, get_strng

Name: macro_file_name
Aliases: None
Type Of Entry: Global variable
Description: The name of the text file that contains the commands
and data input as macro commands for non-interactive
program use.

Make_up: Paramstring
Source: Declared in icecappc
Destination:
Used In: icecappc, get_data

Name: MACRO.INP
Type: Text
Description: This file contains the command and the data input for
the non_interactive mode of ICECAP-PC.
Used In: get_data

Name: make_pretty
Type: Procedure
Description: This procedure pretties up the screen for transfer
function input.
Global Variables Used: term, degree
Global Variables Changed: None
Global Constants Used: crt_only, screen_width, as_assigned
Passed Variables: row, degree

Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: gotoxy, disp_msg, graphics, nographics,
put_string, out_int
Called By: get_fact, get_unfact, disptf, disppoly

Version: 2.3
Date: 26 Aug 85
Author: Vincent M Parisi II, Capt, USAF
Modified by: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: GETTF.PAS

Name: make_pretty_large_matrix_one
Type: Procedure
Description: This procedure will draw the left bracket and place
row and col numbers on the first display screen of
a matrix with more than 5 cols.

Global Variables Used: None
Global Variables Changed: None
Global Constants Used: crt_only, as_assigned
Passed Variables: num_row, num_col
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: gotoxy, out_string, out_int,
left_bracket
Called By: getmat, disp_matrx, chgmat, get_matrix_entries

Version: 1.0
Date: 11 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: GETMAT.PAS

Name: make_pretty_large_matrix_two
Type: Procedure
Description: This procedure will draw the right bracket of a
matrix with more than 5 columns. It will also write
the col and row identifiers on the screen.

Global Variables Used: None
Global Variables Changed: None

Global Constants Used: crt_only, as_assigned
Passed Variables: num_row, num_col
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: gotoxy, out_string, out_int,
left_bracket
Called By: getmat, disp_matrx, chgmat, get_matrix_entries

Version: 1.0
Date: 11 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: GETMAT.PAS

Name: make_pretty_small_matrix
Type: Procedure
Description: This procedure will draw the brackets and label the
cols and rows for a matrix with 5 cols or less.

Global Variables Used: None
Global Variables Changed: None
Global Constants Used: crt_only, as_assigned
Passed Variables: num_row, num_col
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: gotoxy, out_string, out_int,
left_bracket, right_bracket
Called By: getmat, disp_matrx, chgmat

Version: 1.0
Date: 11 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: GETMAT.PAS

Name: mat
Aliases: None
Type Of Entry: Variable
Description: This is a matrix.
Make_up: matrix
Source:
Destination:

Used In: recover, update, chgmat, disp_matrx

Name: mat1
Aliases: None
Type Of Entry: Variable
Description: This is a temporary matrix that is used to pass
matrices between procedures.
Make_up: matrix
Source:
Destination:
Used In: matrxmanip1, matrxmanip2

Name: mat2
Aliases: None
Type Of Entry: Variable
Description: This is a temporary matrix that is used to pass
matrices between procedures.
Make_up: matrix
Source:
Destination:
Used In: matrxmanip1, matrxmanip2

Name: mat3
Aliases: None
Type Of Entry: Variable
Description: This is a temporary matrix that is used to pass
matrices between procedures.
Make_up: matrix
Source:
Destination:
Used In: matrxmanip1

Name: mata
Aliases: None
Type Of Entry: Variable
Description: This is a file of matrix.
Make_up: file of matrix
Source:
Destination:
Used In: recover, update

```
*****
Name:      mat_file
Aliases:   None
Type Of Entry: Variable
Description: This is a file of matrix.
Make_up:   file of matrix
Source:
Destination:
Used In:    move_matrix
*****
```

```
*****
Name:      mat_name
Aliases:   None
Type Of Entry: Variable
Description: This is the matrix name.
Make_up:   msg_line
Source:
Destination:
Used In:    mmatrix
*****
```

```
*****
Name:      mat_obj
Aliases:   None
Type Of Entry: Variable
Description: This variable is used to indicate the object that
             the called routine is to operate with.
Make_up:   cmdword
Source:
Destination:
Used In:    matrxmanip2, mmatrix
*****
```

```
*****
Name:      matrices
Aliases:   None
Type Of Entry: Variable
Description: This is a file of matrix.
Make_up:   msg_line
Source:
Destination:
Used In:    getmat
*****
```

```
*****
Name:      matrix
Aliases:   None
Type Of Entry: Type definition
*****
```


Description: This is the record definition for a file of matrix.
Make_up: matrix = record
 num_rows : integer;
 num_cols : integer;
 element : array[1..max_rows, 1..max_cols]
 of real;
 end;
Source: Declared in concons

Destination:
Used In: recover, update, move_matrix,
 matrxmanip1, matrxmanip2, disp_matrx
 matrxtran, matrxinv, chqmat,
 get_matrx_entries, matrxsub, smatrxmlt,
 getmat, matrxadd, mmatrxmlt

Name: MATRIX.PAS
Type: File
Description: This file will decode the command string from
 define.pas and call the appropriate procedures.
Procedures Contained: disp_matrx, matrx_manip1, matrx_manip2,
 get_matrx_name, mmatrix
Version: 1.0
Date: 22 Sep 85
Author: Susan K. Mashiko, Capt, USAF
 Gary C. Tarczynski, Capt, USAF

Name: matrix
Aliases: None
Type Of Entry: Variable
Description: This is a matrix.
Make_up: matrix
Source:
Destination:
Used In: move_matrix

Name: MATRXMAN.PAS
Type: File
Description: This file contains the matrix manipulation procedures.
Procedures Contained: matrxadd, mmatrxmlt, matrxsub,
 smatrxsub, matrxtran, matrxinv
Version: 1.0
Date: 22 Sep 85
Author: Susan K. Mashiko, Capt, USAF

Gary C. Tarczynski, Capt, USAF

Name: matrxadd
Type: Procedure
Description: This procedure will add two matrices together and store the result in the third matrix passed to the procedure.

Global Variables Used: abort_command
Global Variables Changed: abort_command
Global Constants Used: None
Passed Variables: amat, bmat, cmat, abort_command
Returned: cmat, abort_command
Files Read: None
Files Written: None
Aliases: None
Procedures Called: clear, gotoxy, pause, disp_msg
Called By: matrxsub, matrxmanipl

Version: 1.0
Date: 18 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: MATRXMAN.PAS

Name: matrxinv
Type: Procedure
Description: This procedure will invert a matrix and place the result in the second matrix passed to it.

Global Variables Used: abort_command
Global Variables Changed: None
Global Constants Used: None
Passed Variables: amat, bmat, abort_command
Returned: amat, bmat, abort_command
Files Read: None
Files Written: None
Aliases: None
Procedures Called: clear, gotoxy, pause, disp_msg
Called By: matrxmanipl2

Version: 1.0
Date: 18 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: MATRXMAN.PAS

```

*****
Name:   matr_x_manip1
Type:   Procedure
Description: This procedure add, subtract and multiply two matrices.
Global Variables Used:   abort_command
Global Variables Changed: None
Global Constants Used:   None
Passed Variables: first, second, third, mat_obj
Returned:   third
Files Read:  MATRIX.DAT
Files Written: MATRIX.DAT
Aliases:     None
Procedures Called: trim, disp_matrx, matr_xadd, mmatr_xmlt,
                  matr_xsub
Called By:   mmatrix

Version:     1.0
Date:        21 Sep 85
Author:      Susan K. Mashiko, Capt, USAF
              Gary C. Tarczynski, Capt, USAF
Contained In File: MATRIX.PAS
*****

```

```

*****
Name:   matr_x_manip2
Type:   Procedure
Description: This procedure invert, transpose, or multiply a
            matrix by a scalar.
Global Variables Used:   abort_command
Global Variables Changed: None
Global Constants Used:   None
Passed Variables: first, number, result, mat_obj
Returned:   result
Files Read:  MATRIX.DAT
Files Written: MATRIX.DAT
Aliases:     None
Procedures Called: trim, disp_matrx, matr_xinv, smatr_xmlt,
                  matr_xtran
Called By:   mmatrix

Version:     1.0
Date:        21 Sep 85
Author:      Susan K. Mashiko, Capt, USAF
              Gary C. Tarczynski, Capt, USAF
Contained In File: MATRIX.PAS
*****

```

```

*****
Name:   matr_xsub

```

Type: Procedure
Description: This procedure will subtract the second matrix from
the the first and store it in the third.
Global Variables Used: abort_command
Global Variables Changed: abort_command
Global Constants Used: None
Passed Variables: amat, bmat, cmat, abort_command
Returned: cmat, abort_command
Files Read: None
Files Written: None
Aliases: None
Procedures Called: matrxdadd
Called By: matrxdmanipl

Version: 1.0
Date: 20 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: MATRXMAN.PAS

Name: matrxdtran
Type: Procedure
Description: This procedure will transpose a matrix and pass it
in the second matrix passed to it.
Global Variables Used: None
Global Variables Changed: None
Global Constants Used: None
Passed Variables: amat, bmat
Returned: amat, bmat
Files Read: None
Files Written: None
Aliases: None
Procedures Called: None
Called By: matrxdmanip2

Version: 1.0
Date: 18 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: MATRXMAN.PAS

Name: mats
Aliases: None
Type Of Entry: Variable
Description: This is a file of matrix.

Make_up: file of matrix
Source:
Destination:
Used In: recover, update, matrxmanip1,
matrxmanip2, disp_matrx, getmat,
chgmat

Name: max_cols
Aliases: None
Type Of Entry: Global constant
Description: This is the maximum number of columns a matrix may
have.
Make_up: Integer
Source: Declared in concons
Destination:
Used In: matrix, getmat

Name: max_deg
Aliases: None
Type Of Entry: Global constant
Description: This is the maximum degree of the polynomials.
Make_up: Integer
Source: Declared in concons
Destination:
Used In: polynomial, gettf, get_poly

Name: max_deg1
Aliases: None
Type Of Entry: Global constant
Description: This is the maximum degree of the polynomials plus
one.
Make_up: Integer
Source: Declared in concons
Destination:
Used In: polynomial

Name: max_rows
Aliases: None
Type Of Entry: Global constant
Description: This is the maximum number of rows allowed in a
matrix.

Make_up: Integer
Source: Declared in concons
Destination:
Used In: matrix, getmat

Name: method
Aliases: None
Type Of Entry: Variable
Description: This is the method by which the polynomial will be entered, either poly or factored form.

Make_up: cmdword
Source:
Destination:
Used In: poly, gettf, get_poly

Name: mmatrix
Type: Procedure
Description: This procedure will decode the command string from define.pas and all the appropriate procedures.
Global Variables Used: abort_command, cmdbuffer
Global Variables Changed: abort_command
Global Constants Used: as_assigned
Passed Variables: cmdbuffer, wordnumber
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: pause, clear, matrxmanip2, get_matrx_name, get_r_num, gotoxy, out_string, disp_matrx, trim, disp_msg, matrxmani1
Called By: disp

Version: 1.0
Date: 19 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: MATRIX.PAS

Name: mmatrxmlt
Type: Procedure
Description: This procedure will multiply the second matrix to the the first and store it in the third.
Global Variables Used: abort_command

Global Variables Changed: abort_command
Global Constants Used: None
Passed Variables: amat, bmat, cmat, abort_command
Returned: cmat, abort_command
Files Read: None
Files Written: None
Aliases: None
Procedures Called: clear, gotoxy, pause, disp_msg
Called By: matrxmanipl

Version: 1.0
Date: 18 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: MATRXMAN.PAS

Name: modify
Type: Procedure
Description: This procedure contains the logic to decide which
modification procedure should be called and calls
it.

Global Variables Used: cmdbuffer
Global Variables Changed: None
Global Constants Used: None
Passed Variables: cmdbuffer, wordnumber
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: inroot, delroot, chgmat, clear, trim,
disp_msg, pause
Called By: select_routine

Version: 1.0
Date: 22 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: MODIFY.PAS

Name: MODIFY.PAS
Type: File
Description: This file contains the logic to decide which
modification procedure should be called and calls
it.

Procedures Contained: chgmat, modify

Version: 1.0
Date: 22 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

Name: mod_msg
Aliases: None
Type Of Entry: Variable
Description: This is the number of the message for the modify command.
Make_up: Integer
Source:
Destination:
Used In: help

Name: mod_obj
Aliases: None
Type Of Entry: Variable
Description: This variable is used to indicate the object that the called routine is to operate with.
Make_up: cmdword
Source:
Destination:
Used In:

Name: move_matrix
Type: Procedure
Description: This procedure receives the source and destination matrix locations, reads the source matrix location and copies it to the destination.
Global Variables Used: None
Global Variables Changed: None
Global Constants Used: None
Passed Variables: source, dest_loc
Returned: None
Files Read: TF&POLS.DAT
Files Written: TF&POLS.DAT
Aliases: None
Procedures Called: None
Called By: ccopyy

Version: 2.0
Date: 4 Sep 85

Author: Vincent M Parisi II, Capt, USAF
Modifeid by: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: COPY.PAS

Name: move_poly
Type: Procedure
Description: This procedure receives the source and destination
poly locations, reads the source poly location
and copies it to the destination.

Global Variables Used: None
Global Variables Changed: None
Global Constants Used: None
Passed Variables: sorce, dest_loc
Returned: None
Files Read: TF&POLS.DAT
Files Written: TF&POLS.DAT
Aliases: None
Procedures Called: None
Called By: ccopyy

Version: 2.0
Date: 4 Sep 85
Author: Vincent M Parisi II, Capt, USAF
Modifeid by: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: COPY.PAS

Name: move_tf
Type: Procedure
Description: This procedure receives the source and destination
tf locations, reads the source tf location
and copies it to the destination.

Global Variables Used: None
Global Variables Changed: None
Global Constants Used: None
Passed Variables: sorce, dest_loc
Returned: None
Files Read: TF&POLS.DAT
Files Written: TF&POLS.DAT
Aliases: None
Procedures Called: None
Called By: ccopyy

Version: 2.0

Date: 4 Sep 85
Author: Vincent M Parisi II, Capt, USAF
Modifeid by: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: COPY.PAS

Name: MSDWCONS.PAS
Type: File of constants
Description: This file contains the constant definitions for the
MICROSDW ans ICECAPPC routines

Global Variables Used:
Global Variables Changed:
Global Constants Used:
Passed Variables:
Returned:
Files Read:
Files Written:
Aliases:
Procedures Called:
Called By:

Version: 4.0
Date: 19 September 85
Author: Paul A Moore, Capt, USAF
Modifiers: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: MSDWCONS.PAS

Name: MSDWTYPE.PAS
Type: File of type definitions
Description: This file contains the type definitions for the
MICROSDW and ICECAPPC routines

Global Variables Used:
Global Variables Changed:
Global Constants Used:
Passed Variables:
Returned:
Files Read:
Files Written:
Aliases:
Procedures Called:
Called By:

Version: 4.0
Date: 19 September 85

Author: Paul A Moore, Capt, USAF
Modifiers: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: MSDWTYPE.PAS

Name: msg
Aliases: None
Type Of Entry: Global type definition
Description: Type definition of the structure that contains
the message record number and length in number of
records.

Make_up: msg = record
loc_rec : integer;
length : byte;
end;

Source: Declared in msdwtype

Destination:

Used In: msdwtype

Name: msg_array
Aliases: None
Type Of Entry: Global type definition
Description: This is the type definition of the structure that
contains the message directory.

Make_up: array[1..num_msg_dir] of msg

Source: Declared in icecappc

Destination:

Used In: icecappc, get_data

Name: msg_dat
Aliases: None
Type Of Entry: Global type definition
Description: Type definition of one line from the file of the
message text.

Make_up: msg_dat = array[1..num_msg_line] of string[screen
width]

Source: Declared in msdwtype

Destination:

Used In:

Name: msg_dir

Aliases: None
 Type Of Entry: Global variable and data flow
 Description: This is the message directory that contains the record number of the messages in the HELP.TXT file. It also is a counter for the total number of messages in HELP.TXT
 Make_up: msg_array
 Source: Declared in icecappc
 Destination:
 Used In: icecappc, disp_msg, clear_msg, get_data

Name: msg_line
 Aliases: None
 Type Of Entry: Global type definition
 Description: This is the type definition of a long string which can then be used as a parameter for passing between procedures.
 Make_up: string[screen_width]
 Source: Declared in icecappc
 Destination: N/A
 Used In: icecappc, ucase, out_string, get_strng, recover, update, ck_chr, get_poly_name, svideobold, svideolow

Name: msg_num
 Aliases: None
 Type Of Entry: Variable
 Description: This variable is the number of the message from the message file that will either be displayed or cleared.
 Make_up: Integer
 Source:
 Destination:
 Used In: disp_msg, clear_msg

Name: MSG.PAS
 Type: File
 Description: This file contains the procedures to display and to clear a message.
 Procedures Contained: disp_line, disp_msg, clear_msg
 Version: 3.1

Date: 23 Aug 85
Author: Vincent M. Parisi II, Capt, USAF
Modified by: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

Name: msg_txt
Aliases: None
Type Of Entry: Global accessed file
Description: This is the file that contains the message text
for display throughout the program. System warnings,
instructions, error messages and help information is
contained in this file.
Make_up: file of msg_dat
Source: Declared in icecappc
Destination:
Used In: disp_line

Name: n
Aliases: None
Type Of Entry: Variable
Description: Used as a counter.
Make_up: Integer
Source:
Destination:
Used In: polyadd

Name: naa
Aliases: None
Type Of Entry: Variable
Description: Used as a counter.
Make_up: Integer
Source:
Destination:
Used In: polymlt

Name: nbb
Aliases: None
Type Of Entry: Variable
Description: Used as a counter.
Make_up: Integer
Source:

Destination:
Used In: polymlt, polysub

Name: nbmat
Aliases: None
Type Of Entry: Variable
Description: File of matrix.
Make_up: matrix
Source:
Destination:
Used In: matrxsub

Name: nbpoly
Aliases: None
Type Of Entry: Variable
Description: This is a storage location for the polynomial.
Make_up: polynomial
Source:
Destination:
Used In: polysub

Name: nc
Aliases: None
Type Of Entry: Variable
Description: Used as a counter.
Make_up: Integer
Source:
Destination:
Used In: polyadd

Name: ncc
Aliases: None
Type Of Entry: Variable
Description: Used as a counter.
Make_up: Integer
Source:
Destination:
Used In: polyadd

Name: newpoly
Aliases: None
Type Of Entry: Variable
Description: This is a storage location for the polynomial after
it has been modified.
Make_up: polynomial
Source:
Destination:
Used In: inroot, delroot

Name: nn
Aliases: None
Type Of Entry: Variable
Description: Used as a counter.
Make_up: Integer
Source:
Destination:
Used In: polyadd

Name: no
Aliases: None
Type Of Entry: Global constant
Description: Boolean Variable
Make_up: Boolean
Source: Declared in icecappc
Destination: N/A
Used In:

Name: nographics
Type: Procedure
Description: This procedure removes the terminal from the graphics
mode.
Global Variables Used: term
Global Variables Changed: None
Global Constants Used: term_length
Passed Variables: None
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called:
Called By: rectangle, left_bracket, right_bracket

Version: 2.0
Date: 21 Oct 83
Author: Vincent M Parisi II, Capt, USAF
Contained In File: TERMINAL.PAS

Name: nohighlight
Type: Procedure
Description: This procedure removes the terminal from the reverse video mode.

Global Variables Used: term
Global Variables Changed: None
Global Constants Used: term_length
Passed Variables: None
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called:
Called By: pause, title_slide, prompt_cmd,
proces_error, ccopy, get_r_num,
get_real, make_pretty, roots,
delroot, form, get_poly_name,
polymt, get_matrx_name

Version: 2.0
Date: 21 Oct 83
Author: Vincent M Parisi II, Capt, USAF
Contained In File: TERMINAL.PAS

Name: number
Aliases: None
Type Of Entry: Variable
Description: This is a variable
Make_up: Real
Source: Declared in icecappc
Destination: N/A
Used In: out_int, get_int, out_real,
get_real, get_r_num, get_fact,
get_unfact, spolymlt, matrxmanip2,
ppoly, dispmatrx, polymanip2,
smatrxmlt, get_matrix_entries, disptf,
mmatrix, getmat, chgmat,
disppoly, inroot

```

*****
Name:          number_of_commands
Aliases:       None
Type Of Entry: Data flow
Description:    The number of command words entered by the user,
                calculated by: bufferpointer - 1.
Make_up:       Integer
Source:        Declared in icecappc
Destination:
Used In:       icecappc,    select_routine
*****

```

```

*****
Name:          num_bools
Aliases:       None
Type Of Entry: Global constant
Description:    The number of boolean variables within each
                parameter group.
Make_up:       Integer (10)
Source:        Declared in msdwcons
Destination:
Used In:       msdwtype
*****

```

```

*****
Name:          num_col
Aliases:       None
Type Of Entry: Variable
Description:    This variable is the number of cols in a matrix.
Make_up:       Integer
Source:
Destination:
Used In:       disp_matrx,    chgmat,    getmat,
                make_pretty_large_matrix_one,    get_matrx_entries
*****

```

```

*****
Name:          num_cols
Aliases:       None
Type Of Entry: Variable
Description:    This variable is the number of columns in a matrix.
Make_up:       Integer
Source:
Destination:
Used In:       left_bracket,    right_bracket
*****

```

```

*****
Name:          num_deg

```

Aliases: None
Type Of Entry: Variable
Description: This variable is the degree of the numerator polynomial.
Make_up: Integer
Source:
Destination:
Used In: gettf

Name: numerator
Aliases: None
Type Of Entry: Variable
Description: This variable is the numerator polynomial.
Make_up: Polynomial
Source:
Destination:
Used In: gettf, disptf

Name: num_ints
Aliases: None
Type Of Entry: Global constant
Description: The number of integer variables within each parameter group.
Make_up: Integer (10)
Source: Declared in msdwcons
Destination:
Used In: msdwtype

Name: num_msg_dir
Aliases: None
Type Of Entry: Global constant
Description: The length of the message directory--i.e. the HELP.TXT file is limited to this number of messages.
Make_up: Integer
Source: Declared in msdwcons
Destination:
Used In: msdwtype, icecappc, get_data

Name: num_msg_line
Aliases: None

Type Of Entry: Global constant
Description: The number of lines of message text available in the file.

Make_up: Integer
Source: Declared in msdwcons
Destination:
Used In: msdwtype

Name: num_of_commands
Aliases: None
Type Of Entry: Variable
Description: The number of commands.

Make_up: Integer
Source: val_n_dec, get_cmd
Destination:
Used In: val_n_dec, get_cmd

Name: num_param_group
Aliases: None
Type Of Entry: Global constant
Description: The number of parameter groups in the file MICROWSDW.SYS

Make_up: Integer
Source: Declared in msdwcons
Destination:
Used In: msdwtype

Name: num_ptr_recs
Aliases: None
Type Of Entry: Global constant
Description: The number of pointers within each record of the command syntax data structure.

Make_up: Integer
Source: Declared in msdwcons
Destination:
Used In: msdwtype

Name: num_ptrs
Aliases: None
Type Of Entry: Global constant
Description: The number of records with decoding information in

the command syntax structure. If the displayed menu contains strange characters there may be insufficient num_ptrs.

Make_up: Integer
Source: Declared in msdwcons
Destination:
Used In: msdwtype, get_data

Name: num_reals
Aliases: None
Type Of Entry: Global constant
Description: The number of real variables within each parameter group.

Make_up: Integer
Source: Declared in msdwcons
Destination:
Used In: msdwtype

Name: num_row
Aliases: None
Type Of Entry: Variable
Description: This variable is the number of rows in a matrix.
Make_up: Integer
Source:
Destination:
Used In: disp_matrx, chgmat, getmat,
make_pretty_large_matrix_one, get_matrx_entries

Name: num_rows
Aliases: None
Type Of Entry: Variable
Description: This variable is the number of rows in a matrix.
Make_up: Integer
Source:
Destination:
Used In: left_bracket, right_bracket

Name: num_strings
Aliases: None
Type Of Entry: Global constant
Description: The number of strings within each parameter

group.
Make_up: Integer
Source: Declared in msdwcons
Destination:
Used In: msdwtype

Name: num_words
Aliases: None
Type Of Entry: Global constant
Description: The number of dictionary words in the command
syntax data structure. If words are missing from
your menu increasing this number should solve the
problem.

Make_up: Integer
Source: Declared in msdwcons
Destination:
Used In: msdwtype, get_data

Name: oldpoly
Aliases: None
Type Of Entry: Variable
Description: This variable is the original polynomial before
it is modified.

Make_up: polynomial
Source:
Destination:
Used In: inroot, delroot

Name: odpol
Aliases: None
Type Of Entry: Variable
Description: This variable is the denominator polynomial of the
OLTF.

Make_up: polynomial
Source:
Destination:
Used In: form

Name: onpol
Aliases: None
Type Of Entry: Variable

Description: This variable is the numerator polynomial of the
OLTF.

Make_up: polynomial

Source:

Destination:

Used In: form

Name: on_off

Aliases: None

Type Of Entry: Variable

Description: This variable is displayed in the status line
at the bottom of the screen.

Make_up: Char

Source: bld_stat_line

Destination:

Used In: bld_stat_line

Name: ostring

Aliases: None

Type Of Entry: Variable

Description: The string the user wishes to output.

Make_up: msg_line

Source: out_string

Destination:

Used In: out_string

Name: out_int

Type: Procedure

Description: This procedure directs the output of integers.

Global Variables Used: crt, trans, printer,
temp, trans_file, temp_file,
list_dev

Global Variables Changed: temp_file, trans_file, list_dev

Global Constants Used: None

Passed Variables: number, field, dest

Returned: None

Files Read: None

Files Written: temp_file, trans_file, list_dev,
temp.out, transaction, printer.out

Aliases: None

Procedures Called:

Called By: make_pretty, inroot,
make_pretty_large_matrix_one,

make_pretty_small_matrix

Version: 1.2
Date: 18 Aug 83
Author: Vincent M Parisi II, Capt, USAF
Contained In File: GETINT.PAS

Name: OUTPUT.PAS
Type: File
Description: This file contains the procedure that handles all of
output.

Procedure Contained: out_string
Version: 1.0
Date: 1 Aug 83
Author: Vincent M Parisi II, Capt, USAF

Name: out_real
Type: Procedure
Description: This procedure outputs real numbers to the crt or
any of the required files.

Global Variables Used: crt, trans, printer, temp
list_dev

Global Variables Changed: None
Global Constants Used: None

Passed Variables: number, fieldwidth, dest
Returned: None
Files Read: trans_file, temp_file, list_dev
Files Written: printer.out, temp.out, transaction
Aliases: None

Procedures Called:
Called By: get_fact, get_unfact, get_r_num,
disptf, chgmat, inroot,
disppoly, disp_matrx

Version: 1.4
Date: 2 Sep 83
Author: Vincent M Parisi II, Capt, USAF
Contained In File: REALS.PAS

Name: out_string
Type: Procedure
Description: This procedure handles all string output for the pro-
gram. Whenever system output is required, this module

```

is called, the output is directed to the appropriate
device. ( crt, printer, transaction, temporary )
Global Variables Used:   trans, printer, temp, crt, list_dev,
                        trans_file, temp_file
Global Variables Changed: None
Global Constants Used:  None
Passed Variables:      ostring, dest
Returned:              None
Files Read:            trans_file, temp_file, list_dev
Files Written:         printer.out, transaction, temp.out
Aliases:               None
Procedures Called:
Called By:             pause,          clear_msg,          disp_line,
                       instruction,  prompt_help,       prompt_cmd,
                       displa_commandword, recover,          update,
                       ccopyy,      get_real,          get_r_num,
                       make_pretty,  gettf,            get_poly,
                       getmat,       get_poly_name,    disptf,
                       chgmat,      ppoly,            inroot,
                       delroot,     mmatrix,         form,
                       disppoly,    make_pretty_large_matrix_one,
                       disp_matrx,  make_pretty_small_matrix,
                       disp,       get_matrx_name

```

```

Version:      1.0
Date:         1 Aug 83
Author:       Vincent M Parisi II, Capt, USAF
Contained In File: OUTPUT.PAS

```

```

Name:         param_group
Aliases:      None
Type Of Entry: Global type definition
Description:   Type definition of the structure that contains the
              program's parameters which are used throughout the
              program.
Make_up:      param_group = record
              bools   : array[ 1..num_bools ] of boolean;
              ints    : array[ 1..num_ints ] of integer;
              reals   : array[ 1..num_reals ] of reals;
              strings : array[ 1..num_strings ] of paramstring;
              end;
Source:       Declared in msdwtype
Destination:
Used In:      icecappc,      msdwtype

```

Name: paramstring
 Aliases: None
 Type Of Entry: Global type definition
 Description: Type definition of a fourteen char string to be used thruout the program.
 Make_up: paramstring = string[14]
 Source: Declared in msdwtype
 Destination:
 Used In: get_data

Name: pause
 Type: Procedure
 Description: This procedure waits for user response to continue anytime there is a stop in the program.
 Global Variables Used: blanks, status_line, stat_on
 Global Variables Changed: None
 Global Constants Used: screen_width, crt_only, stat_line_width
 Passed Variables: None
 Returned: None
 Files Read: None
 Files Written: None
 Aliases: None
 Procedures Called: gotoxy, highlight, nohighlight,
 out_string
 Called By: proces_error, recover, update,
 ccopy, help, get_real,
 get_fact, matrxadd, get_matrx_entries,
 mmatrxmlt, disptf, getmat,
 matrxinv, define, modify,
 chgmat, dispmatrx, inroot,
 delroot, mmatrix, form
 polymanip, polymanip2, get_poly_name,
 get_matrx_name, get_poly, disp,
 select_routine

Version: 1.1
 Date: 29 Oct 84
 Author: Vincent M Parisi II, Capt, USAF
 Modified by: Paul A Moore, Capt, USAF
 Contained In File: PAUSE.PAS

Name: PAUSE.PAS
 Type: File
 Description: This file waits for user response to continue anytime there is a stop in the program.

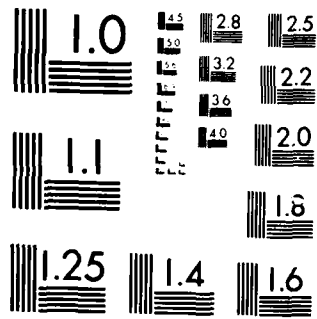
Procedures Contained: pause
Version: 1.1
Date: 29 Oct 84
Author: Vincent M Parisi II, Capt, USAF
Modified by: Paul A Moore, Capt, USAF

Name: pi
Aliases: None
Type Of Entry: Constant
Description: This is the value of pi.
Make_up: real
Source:
Destination:
Used In:

Name: pol
Aliases: None
Type Of Entry: Variable
Description: This is a polynomial.
Make_up: polynomial.
Source:
Destination:
Used In: recover, update, disppoly, get_poly,
poly_into_storage, poly_from_storage

Name: poll
Aliases: None
Type Of Entry: Variable
Description: This is a polynomial.
Make_up: polynomial.
Source:
Destination:
Used In: polymanip, polymanip2

Name: pol2
Aliases: None
Type Of Entry: Variable
Description: This is a polynomial.
Make_up: polynomial.
Source:
Destination:



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

Used In: polymanip, polymanip2

Name: pol3
Aliases: None
Type Of Entry: Variable
Description: This is a polynomial.
Make_up: polynomial.
Source:
Destination:
Used In: polymanip

Name: pola
Aliases: None
Type Of Entry: Variable
Description: This is a file of polynomial.
Make_up: file of polynomial.
Source:
Destination:
Used In: recover, update

Name: pol_deg
Aliases: None
Type Of Entry: Variable
Description: This is a the degree of the polynomial.
Make_up: Integer
Source:
Destination:
Used In: get_poly

Name: pols
Aliases: None
Type Of Entry: Variable
Description: This is a file of polynomial.
Make_up: file of polynomial.
Source:
Destination:
Used In: recover, update

Name: poly

Aliases: None
Type Of Entry: Variable
Description: This is a polynomial.
Make_up: polynomial.
Source:
Destination:
Used In: poly, roots, get_unfact,
form_poly, get_fact, move_tf,
move_poly

Name: poly
Type: Procedure
Description: This procedure gets a polynomial in either factored
or poly form.
Global Variables Used: abort_command
Global Variables Changed: None
Global Constants Used: None
Passed Variables: method, poly, disp_row,
abort_command
Returned: poly
Files Read: None
Files Written: None
Aliases: None
Procedures Called: get_unfact, get_fact,
roots, form_poly
Called By: gettf

Version: 1.2
Date: 25 Sep 85
Author: Vincent M Parisi II, Capt, USAF
Modified by: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: GETTF.PAS

Name: polyadd
Type: Procedure
Description: This procedure will add the second polynomial
to the first and store it in the third.
Global Variables Used: polynomial
Global Variables Changed: polynomial
Global Constants Used: None
Passed Variables: apoly, bpoly, cpoly
Returned: apoly, bpoly, cpoly
Files Read: None
Files Written: None

Aliases: None
Procedures Called: roots
Called By: polymanip, polysub

Version: 2.0
Date: 18 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: POLYMAN.PAS

Name: poly_file
Aliases: None
Type Of Entry: Variable
Description: This a file of polynomial.
Make_up: polynomial
Source:
Destination:
Used In: move_poly

Name: poly_from_storage
Type: Procedure
Description: This procedure will remove a polynomial into storage.
Global Variables Used: None
Global Variables Changed: None
Global Constants Used: None
Passed Variables: choice, pol
Returned: None
Files Read: TF&POLS.DAT
Files Written: None
Aliases: None
Procedures Called: trim
Called By: form

Version: 1.0
Date: 7 Oct 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: FORM.PAS

Name: poly_into_storage
Type: Procedure
Description: This procedure will place a polynomial into storage.
Global Variables Used: None

Global Variables Changed: None
Global Constants Used: None
Passed Variables: choice, pol
Returned: None
Files Read: None
Files Written: TF&POLS.DAT
Aliases: None
Procedures Called: trim
Called By: form

Version: 1.0
Date: 7 Oct 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: FORM.PAS

Name: polymanip
Type: Procedure
Description: This procedure will add, subtract, or multiply two
polynomials.

Global Variables Used: polynomial
Global Variables Changed: None
Global Constants Used: None
Passed Variables: first, second, result
Returned: result
Files Read: TF&POLS.DAT
Files Written: TF&POLS.DAT
Aliases: None
Procedures Called: trim, disppoly, polyadd, polylt,
polysub, pause
Called By: ppoly

Version: 1.0
Date: 6 Sept 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: POLY.PAS

Name: polymanip2
Type: Procedure
Description: This procedure will multiply a polynomial and a scalar.
Global Variables Used: polynomial
Global Variables Changed: None
Global Constants Used: None
Passed Variables: first, number, result, poly_obj

Returned: result
Files Read: None
Files Written: None
Aliases: None
Procedures Called: trim, disppoly, spolymlt
Called By: ppoly

Version: 1.0
Date: 8 Oct 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: POLY.PAS

Name: POLYMAN.PAS
Type: File
Description: This file will add, subtract, or multiply two
polynomials. Additionally, it will multiply a
polynomial by a scalar.
Procedures Contained: polyadd, polymlt, polysub, spolymlt
Version: 3.0
Date: 8 Oct 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

Name: polymlt
Type: Procedure
Description: This procedure will multiply the second polynomial
to the first and store it in the third.
Global Variables Used: polynomial
Global Variables Changed: polynomial
Global Constants Used: None
Passed Variables: apoly, bpoly, cpoly
Returned: cpoly
Files Read: None
Files Written: None
Aliases: None
Procedures Called: roots, clear, gotoxy, highlight,
nohighlight
Called By: polymanip, form

Version: 1.0
Date: 4 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: POLYMAN.PAS

Name: poly_name
 Aliases: None
 Type Of Entry: Variable
 Description: This variable is the name of polynomial obtained by this procedure.
 Make_up: msg_line
 Source:
 Destination:
 Used In: ppoly, get_poly_name

Name: polynomial
 Aliases: None
 Type Of Entry: Global type definition
 Description: Record that contains the components of a polynomial.
 Make_up: polynomial = record
 change : boolean;
 coefficient : real;
 polydeg : integer;
 polyfact : array[1..max_deg] of complex;
 polypoly : array[1..max_deg1] of real;
 end;

Source: Declared in concons
 Destination:
 Used In: poly, roots, get_unfact, get_poly,
 form_poly, get_fact, move_tf, polymanip,
 move_poly, gettf, recover, polymanip2,
 update, polyadd, polysub, spolymlt,
 disptf, disppoly, inroot, delroot,
 ppoly, get_poly_name, poly_from_storage,
 poly_into_storage, form, polymlt

Name: poly_obj
 Aliases: None
 Type Of Entry: Variable
 Description: This variable is the name of polynomial passed to this procedure.
 Make_up: cmdword
 Source:
 Destination:
 Used In: ppoly, polymanip

```

*****
Name:      POLY.PAS
Type:      File
Description: This file contains the procedures to display and
             manipulate polynomials.
Procedures Contained: disppoly, polymanip, ppoly,
                       get_poly_name, polymanip2
Version:    3.0
Date:       8 Oct 85
Author:     Susan K. Mashiko, Capt, USAF
            Gary C. Tarczynski, Capt, USAF
*****

```

```

*****
Name:      polys
Aliases:   None
Type Of Entry: Variable
Description: This is a file of polynomial.
Make up:   file of polynomial.
Source:
Destination:
Used In:   gettf, disptf, poly_into_storage,
           inroot, delroot, poly_from_storage,
           polymanip, polymanip2, disppoly, get_poly
*****

```

```

*****
Name:      polysub
Type:      Procedure
Description: This procedure will subtract the second polynomial
             form the first and store it in the third.
Global Variables Used: polynomial
Global Variables Changed: polynomial
Global Constants Used: None
Passed Variables: apoly, bpoly, cpoly
Returned:   cpoly
Files Read: None
Files Written: None
Aliases:    None
Procedures Called: polyadd
Called By:  polymanip

Version:    1.0
Date:       4 Sep 85
Author:     Susan K. Mashiko, Capt, USAF
            Gary C. Tarczynski, Capt, USAF
Contained In File: POLYMAN.PAS
*****

```

```

*****
Name:      pos
Aliases:   None
Type Of Entry: Variable
Description: This a the passed position.
Make_up:   Integer
Source:
Destination:
Used In:    svideolow,  svideobold
*****

```

```

*****
Name:      ppoly
Type:      Procedure
Description: This procedure will get the name of a polynomial from
            the screen.
Global Variables Used:  abort_command, cmdbuffer
Global Variables Changed: None
Global Constants Used:  as_assigned
Passed Variables:      cmdbuffer, wordnumber
Returned:              None
Files Read:            None
Files Written:         None
Aliases:               None
Procedures Called:    clear, disp_msg, trim, disppoly,
                       get_poly_name, gotoxy, out_string,
                       polymanip, get_r_num, polymanip2,
                       pause
Called By:             disp

Version:              2.0
Date:                19 Sep 85
Author:              Susan K. Mashiko, Capt, USAF
                   Gary C. Tarczynski, Capt, USAF
Contained In File:   POLY.PAS
*****

```

```

*****
Name:      pr_cmd_col
Aliases:   None
Type Of Entry: Constant definition and data flow
Description: This is the column location that the logo appears
            at.
Make_up:   Integer
Source:    get_cmd
Destination:
Used In:    get_cmd
*****

```

```

*****
Name:      pr_cmd_row
Aliases:   None
Type Of Entry: Constant definition and data flow
Description: This is the row location that the logo appears
              at.
Make_up:   Integer
Source:    get_cmd
Destination:
Used In:    get_cmd
*****

```

```

*****
Name:      pr_hlp_row
Aliases:   None
Type Of Entry: Constant definition and data flow
Description: This is the row location that the command word
              prompt appears on.
Make_up:   Integer
Source:    get_cmd
Destination:
Used In:    get_cmd
*****

```

```

*****
Name:      print
Aliases:   None
Type Of Entry: Global file variable and data flow
Description: This is an array that contains the printer control
              codes.
Make_up:   print_array
Source:    Declared in icecappc
Destination:
Used In:    icecappc
*****

```

```

*****
Name:      print_array
Aliases:   None
Type Of Entry: Type definition
Description: This is the type declaration of the global variable
              that contains the printer's control codes.
Make_up:   array[ 1..printer_length ] of byte
Source:    Declared in icecappc
Destination: N/A
Used In:    icecappc,      get_data
*****

```

```

*****

```

Name: print_dat
 Aliases: None
 Type Of Entry: Variable
 Description: This variable is of the type print_array.
 Make_up: print_array
 Source:
 Destination:
 Used In: get_data

Name: print_line
 Aliases: None
 Type Of Entry: Variable
 Description: This is the variable name for output line.
 Make_up: string[screen_width]
 Source: disp_line
 Destination:
 Used In: disp_line

Name: print_msg
 Aliases: None
 Type Of Entry: Variable
 Description: This variable is the message number for help on the
 print command.
 Make_up: Integer
 Source:
 Destination:
 Used In: help

Name: printer
 Aliases: None
 Type Of Entry: Global variable
 Description: This variable is the flag that indicates whether
 the printer file is on. i.e. should all printer
 directed output be saved in the printer file.
 Make_up: Boolean
 Source: Declared in icecappc
 Destination:
 Used In: icecappc, out_string, out_int,
 bld_stat_line, get_data, out_real

Name: printer_length

Aliases: None
Type Of Entry: Global constant
Description: This is the length of the array for printer control codes.
Make_up: Integer
Source: Declared in msdwcons
Destination:
Used In: icecappc, msdwtype, get_data

Name: PRINTER.OUT
Type: File
Description: This file contains the output for the printer. It will only be filled if printer is true.
Used In: get_data

Name: print_line
Aliases: None
Type Of Entry: Variable
Description: This a the line the message line should be displayed on
Make_up: Integer
Source:
Destination:
Used In: disp_line

Name: PROCESER.PAS
Type: File
Description: This procedure handles command decoding errors. It prompts the user for proper action to take for error correction.
Procedures Contained: proces_error
Version: 1.1
Date: 16 Aug 83
Author: Vincent M Parisi II, Capt, USAF

Name: proces_error
Type: Procedure
Description: This procedure handles command decoding errors. It prompts the user for proper action to take for error correction.
Global Variables Used: help_level, cmdbuffer

Global Variables Changed: None
Global Constants Used: screen_width
Passed Variables: error_code, level, cmdbuffer,
bufferpointer
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: gotoxy, highlight, nohighlight,
pause, display_commandword, disp_msg,
clear_msg
Called By: get_cmd

Version: 1.1
Date: 16 Aug 83
Author: Vincent M Parisi II, Capt, USAF
Contained In File: PROCESER.PAS

Name: PROMPTCM.PAS
Type: File
Description: This file places the command line prompt at
the row and column input.

Procedures Contained: prompt_cmd
Version: 1.2
Date: 30 Oct 83
Author: Vincent M Parisi II, Capt, USAF

Name: prompt_cmd
Type: Procedure
Description: This procedure places the command line prompt at
the row and column input.

Global Variables Used: blanks
Global Variables Changed: None
Global Constants Used: as_assigned, crt_only
Passed Variables: row, col

Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: gotoxy, highlight, nohighlight,
out_string
Called By: get_cmd

Version: 1.2
Date: 30 Oct 83

Author: Vincent M Parisi II, Capt, USAF
Contained In File: PROMPTCM.PAS

Name: prompt_col_offset
Aliases: None
Type Of Entry: Constant
Description: This is the offset from column one for the prompt.
Make_up: Integer
Source: prompt
Destination:
Used In: prompt_help

Name: PROMPTHE.PAS
Type: File
Description: This file displays the acceptable command words
based on those already entered.
Procedure Contained: prompt_help
Version: 2.2
Date: 27 Sep 84
Author: Vincent M Parisi II, Capt, USAF

Name: prompt_help
Type: Procedure
Description: This procedure displays the acceptable command words
based on those already entered.
Global Variables Used: None
Global Variables Changed: None
Global Constants Used: crt_only, endword, doneword
Passed Variables: row, rec_num
Returned: rec_num
Files Read: None
Files Written: None
Aliases: None
Procedures Called: gotoxy, get_line, svideolow,
out_string, svideobold
Called By: get_cmd

Version: 2.1
Date: 20 Oct 83
Author: Vincent M Parisi II, Capt, USAF
Contained In File: PROMPTHE.PAS

```

*****
Name:          ptr_recs
Aliases:       None
Type Of Entry: Global type definition
Description:   Type definition of an element within type dict_buffer
                It defines the structure that contains the pointers
                for the command syntax data structure.
Make_up:       ptr_rec = array[ 1..num_ptr_recs ] of integer
Source:        Declared in msdwtype
Destination:
Used In:       msdwtype
*****

```

```

*****
Name:          r
Aliases:       None
Type Of Entry: Variable
Description:   This is a variable for the IBM unique function stdout.
Make_up:       Integer
Source:
Destination:
Used In:       stdout
*****

```

```

*****
Name:          rad
Aliases:       None
Type Of Entry: Variable
Description:   The radical of a term in roots.
Make_up:       Real
Source:
Destination:
Used In:       roots
*****

```

```

*****
Name:          readcom
Type:          Procedure
Description:   This procedure reads a command from the user and
                splits it into individual words in the command buffer.
Global Variables Used:  cmdbuffer, macro_error, blanks,
                        abort_command, strng
Global Variables Changed: cmdbuffer, strng
Global Constants Used:  as_assigned, buffersize, wordsize,
                        terminal_only
Passed Variables:  cmdbuffer, bufferpointer, abort_command
Returned:          cmdbuffer, bufferpointer
Files Read:        None
Files Written:     None

```

Aliases: None
Procedures Called: get_strng, ucase
Called By: get_cmd

Version: 1.1
Date: 28 Oct 83
Author: Vincent M Parisi II, Capt, USAF
Contained In File: READCOM.PAS

Name: READCOM.PAS
Type: File
Description: This file reads a command from the user and
splits it into individual words in the command buffer.
Procedures Contained: readcom
Version: 1.1
Date: 28 Oct 83
Author: Vincent M Parisi II, Capt, USAF

Name: real_error
Aliases: None
Type Of Entry: Global file variable
Description:
Make_up:
Source: Declared in icecappc
Destination:
Used In:

Name: real_root
Aliases: None
Type Of Entry: Variable
Description: This variable is a temporary storage location for
the procedure roots.
Make-up: Real
Source:
Destination:
Used In: roots

Name: REALS.PAS
Type: File
Description: This file contains the procedures to input and output
real numbers.

Procedure Contained: out_real, get_real
Version: 2.4
Date: 19 Aug 85
Author: Vincent M Parisi II, Capt, USAF
Modified By: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

Name: rec_loc
Aliases: None
Type Of Entry: Variable
Description: This variable is the integer value of the location
in a file.
Make-up: Integer
Source:
Destination:
Used In: get_location

Name: rec_num
Aliases: None
Type Of Entry: Variable
Description: Points to the record number of the syntax data
structure of interest.
Make-up: Variable and data flow
Source: get_line, prompt_help, disp_line,
disp_msg, get_cmd, val_n_dec
Destination:
Used In: get_lin, prompt_help, disp_line,
disp_msg, get_cmd, val_n_dec

Name: recover
Type: Procedure
Description: This procedure copies the user specified files into
ICECAP 'tf&pols.dat' and the 'matrix.dat' files.
Global Variables Used: abort_command
Global Variables Changed: None
Global Constants Used: as_assigned, blanks, crt_only,
Passed Variables: None
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: clear, gotoxy, disp_msg,
get_strng, pause, clear_msg,

Called By: out_string
select_routine

Version: 2.0
Date: 19 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: UPDATE.PAS

Name: recover_msg
Aliases: None
Type Of Entry: Variable
Description: This variable is the message number for help on the
recover command.
Make_up: Integer
Source:
Destination:
Used In: help

Name: RECOVER.PAS
Type: File
Description: This file copies the user specified files into
ICECAP 'tf&pols.dat' and the 'matrix.dat' files.
Procedures Contained: recover
Version: 2.0
Date: 19 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

Name: rectangle
Type: Procedure
Description: This procedure draws a rectangle of given dimensions
on the video screen.
Global Variables Used: term
Global Variables Changed: None
Global Constants Used: term_length
Passed Variables: line, column, width, height
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: graphics, gotoxy, nographics
Called By: title_slide

Version: 1.0
Date: 27 Sep 84
Author: Paul A Moore, Capt, USAF
Contained In File: TERMINAL.PAS

Name: remain_lines
Aliases: None
Type Of Entry: Variable
Description: Counter of the total number of lines remaining in
the selected message.

Make-up: Integer

Source:

Destination:

Used In: disp_msg

Name: repeat1
Aliases: None
Type Of Entry: Label
Description: Used for goto statement.

Make-up: Char

Source:

Destination:

Used In: get_unfact, get_fact, recover,
update

Name: repeat2
Aliases: None
Type Of Entry: Label
Description: Used for goto statement.

Make-up: Char

Source:

Destination:

Used In: recover, update

Name: repeatagain
Aliases: None
Type Of Entry: Label
Description: Used for goto statement.

Make-up: Char

Source:

Destination:

Used In: form

Name: resp
Aliases: None
Type Of Entry: Variable
Description: The variable is the character response from the keyboard. Used when the system is put into a pause state.

Make-up: Char
Source:
Destination:
Used In: pause, disp_msg

Name: result
Aliases: None
Type Of Entry: Variable
Description: The variable is used by the val conversion. Val converts a string to a number (real or integer) result is the location of any invalid characters.

Make-up: Integer
Source:
Destination:
Used In: get_int, get_real, polymanip, polymanip2, chgmat, matrxmanip2

Name: right_bracket
Type: Procedure
Description: This procedure will draw the right bracket for a matrix.

Global Variables Used: term
Global Variables Changed: None
Global Constants Used: None
Passed Variables: num_rows, num_cols
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: graphics, gotoxy, nographics
Called By: make_pretty_large_matrix_two, make_pretty_small_matrix

Version: 1.0
Date: 11 Sep 85

Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: GETMAT.PAS

Name: root_num
Aliases: None
Type Of Entry: Variable
Description: The variable the number of the root to be deleted
from the polynomial.

Make-up: Integer

Source:

Destination:

Used In: delroot,

Name: roots
Type: Procedure
Description: This procedure uses the Bairstow's method of finding
the roots of a polynomial.

Global Variables Used: degree

Global Variables Changed: degree

Global Constants Used: maxdegl

Passed Variables: poly

Returned: poly

Files Read: None

Files Written: None

Aliases: None

Procedures Called: gotoxy, highlight, disp_msg, nohighlight,
clear_msg, pause

Called By: poly, polymt, polyadd, spolymlt

Version: 2.0

Date: 6 Sep 85

Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

Contained In File: GETTF.PAS

Name: row
Aliases: None
Type Of Entry: Variable
Description: The variable is the row location on the screen.

Make-up: Integer

Source:

Destination:

Used In: prompt_cmd, prompt_help, get_r_num,
get_fact, get_unfact, disp_matrx,
get_poly_name, disppoly, get_matrix_entries,
disptf, chgmat, make_pretty_small_matrix,
make_pretty_large_matrix_one, gotoxy

Name: row_count
Aliases: None
Type Of Entry: Variable
Description: The variable is a counter for the row location
on the screen.

Make-up: Integer

Source:

Destination:

Used In: prompt_help

Name: screen_width
Aliases: None
Type Of Entry: Global constant
Description: This is the screen width in characters minus 1.

Make_up: Integer

Source: Declared in msdwcons

Destination:

Used In: icecappc, msdwtype, pause,
get_data, proces_error, make_pretty,
gettf, disp_line

Name: second
Aliases: None
Type Of Entry: Variable
Description: This variable if a matrix or a polynomial.

Make_up: polynomial or matrix

Source:

Destination:

Used In: mmatrix, polymanip, ppoly

Name: selection
Aliases: None
Type Of Entry: Variable
Description: This variable is used to indicate the option number
selected by the user.

Make_up: integer
Source:
Destination:
Used In: form

Name: SELECT.PAS
Type: File
Description: This file receives the name of the routine to call and calls it.
Procedures Contained: select_routine
Version: 6.0
Date: 11 Oct 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Paul A. Moore, Capt, USAF

Name: select_routine
Type: Procedure
Description: This procedure receives the name of the routine to call and calls it.
Global Variables Used: cmdbuffer
Global Variables Changed: None
Global Constants Used: None
Passed Variables: call_routine, cmdbuffer, number_of_commands
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: trim, help, define, disp,
ccopyy, modify, recover, update,
form, frequency_response
Called By: icecappc

Version: 6.0
Date: 11 Oct 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: SELECT.PAS

Name: short_int
Aliases: None
Type Of Entry: Global Variable
Description: This variable is a defined as a integer.

Make_up: Integer
Source: Declared in concons
Destination:
Used In: gettf, make_pretty

Name: show_abbreviation
Aliases: None
Type Of Entry: Global variable
Description: This variable is the flag that indicates whether
the abbreviation of command words should be displayed.
Make_up: Boolean
Source: Declared in icecappc
Destination:
Used In: icecappc, get_data

Name: smatrxmlt
Type: Procedure
Description: This procedure will multiply a matrix by a
scalar.
Global Variables Used: None
Global Variables Changed: None
Global Constants Used: None
Passed Variables: amat, bmat, number
Returned: bmat
Files Read: None
Files Written: None
Aliases: None
Procedures Called: None
Called By: matrx_manip2

Version: 1.0
Date: 20 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: MATRXMAN.PAS

Name: sorce_loc
Aliases: None
Type Of Entry: Variable
Description: This variable is integer value of the storage loca-
tion of the source for a copy.
Make_up: Integer
Source:

Destination:
Used In: ccopy, move_tf, move_poly,
move_matrix

Name: source
Aliases: None
Type Of Entry: Variable
Description: This variable is the name of the source for a copy.
Make_up: cmdword
Source:
Destination:
Used In: ccopy

Name: spolymt
Type: Procedure
Description: This procedure will multiply a polynomial by a
scalar.

Global Variables Used: None
Global Variables Changed: None
Global Constants Used: None
Passed Variables: apoly, bpoly, number
Returned: apoly, bpoly
Files Read: None
Files Written: None
Aliases: None
Procedures Called: None
Called By: polymanip2, form

Version: 1.0
Date: 7 Oct 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: POLYMAN.PAS

Name: standard_output
Type: Procedure
Description: This procedure will redirect the output from TURBO
pascal to the operating system (MS-DOS). This allows
the IBM PC to recognize escape codes.

Global Variables Used: None
Global Variables Changed: None
Global Constants Used: None
Passed Variables: c

Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: None
Called By: any write or writeln statement. By setting
ConOutPtr:=Ofs(standard_output) in the program
icecappc, then the address of this procedure
is put into the pointer ConOutPtr. This pointer
is used by write and writeln statements to locate
code for output to the terminal.

Version: 1.0
Date: 3 Aug 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: STDOUT.PAS

Name: stat_line_width
Aliases:
Type Of Entry: Global constant
Description: Width of the displayed status line
Make_up: Integer (77)
Source: Declared in icecappc
Destination: N/A
Used In: icecappc, clearscreen, pause

Name: stat_on
Aliases: None
Type Of Entry: Global variable and data flow
Description: This variable indicates whether the status line
should be displayed on the CRT.
Make_up: Boolean
Source: Declared in icecappc
Destination:
Used In: icecappc, clear, clearscreen,
pause, get_data

Name: status_line
Aliases: None
Type Of Entry: Global variable
Description: This variable determines whether or no the status
line will be displayed or not. Status line shows

```

the status of the printer, transaction, and temp
switches.
Make_up: string[ stat_line_width ]
Source: Declared in Icecappp
Destination:
Used In: clear, clearscreen, pause,
bld_stat_line, get_data
*****

```

```

*****
Name: STDOUT.PAS
Type: File
Description: This file will redirect the output from TURBO
pascal to the operating system (MS-DOS). This allows
the IBM PC to recognize escape codes.
Procedures Contained: standard_output
Version: 1.0
Date: 3 Aug 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
*****

```

```

*****
Name: step
Aliases: None
Type Of Entry: Variable
Description: This is a counter.
Make_up: Integer
Source:
Destination:
Used In: matrxinv
*****

```

```

*****
Name: stepper
Aliases: None
Type Of Entry: Variable
Description: This is a counter.
Make_up: Integer
Source:
Destination:
Used In: matrxinv
*****

```

```

*****
Name: stop_msg
Aliases: None
Type Of Entry: Variable
Description: This variable is the message number for help on the

```

stop command.
Make_up: Integer
Source:
Destination:
Used In: help

Name: store
Aliases: None
Type Of Entry: Variable
Description: This variable is a temporary storage location for
the procedure roots.

Make_up: Real
Source:
Destination:
Used In: roots

Name: stor_loc
Aliases: None
Type Of Entry: Variable
Description: This variable is a storage location in the files
'tf&pols.dat' and 'matrix.dat'.

Make_up: Integer
Source:
Destination:
Used In: gettf, matrxmanip1, matrxmanip2,
get_poly, getmat, inroot,
delroot, disppoly, disp_matrx,
disptf, chgmat, poly_into_storage,
polymanip, polymanip2, poly_from_storage

Name: strng
Aliases: None
Type Of Entry: Global variable and data flow
Description: This is a general purpose string buffer.

Make_up: String
Source: Declared in icecappc
Destination:
Used In: getchi, ck_chr, get_int,
get_strng, get_real

Name: sum

Aliases: None
Type Of Entry: Variable
Description: This variable is a temporary storage location for
the procedure roots.
Make_up: Real
Source:
Destination:
Used In: roots

Name: svideobold
Type: Procedure
Description: This procedure inserts the char string to put the
screen into bold video.
Global Variables Used: term
Global Variables Changed: None
Global Constants Used: term_length
Passed Variables: instring, pos
Returned: pos
Files Read: None
Files Written: None
Aliases: None
Procedures Called: None
Called By: prompt_help

Version: 1.0
Date: 27 Sep 84
Author: Paul A Moore, Capt, USAF
Contained In File: TERMINAL.PAS

Name: svideolow
Type: Procedure
Description: This procedure inserts the char string to put the
screen into low video.
Global Variables Used: term
Global Variables Changed: None
Global Constants Used: term_length
Passed Variables: instring, pos
Returned: pos
Files Read: None
Files Written: None
Aliases: None
Procedures Called: None
Called By: prompt_help

Version: 1.0

Date: 27 Sep 84
Author: Paul A Moore, Capt, USAF
Contained In File: TERMINAL.PAS

Name: switch
Aliases: None
Type Of Entry: Variable
Description: This variable is the flag that indicates whether
the selected terminal type is either the H19 or a
VT100.
Make_up: Boolean
Source:
Destination:
Used In: ttype

Name: swit_msg
Aliases: None
Type Of Entry: Variable
Description: This variable is the message number for help on the
switch command.
Make_up: Integer
Source:
Destination:
Used In: help

Name: system_msg
Aliases: None
Type Of Entry: Variable
Description: This variable is the message number for help on the
program ICECAP-PC.
Make_up: Integer
Source:
Destination:
Used In: help

Name: temp
Aliases: None
Type Of Entry: Global variable
Description: This variable is the flag that indicates whether
the current transactions should be saved in a temp-
orary file.

Make_up: Boolean
Source: Declared in icecappc
Destination:
Used In: icecappc, out_string, bld_stat_line,
out_int, get_data, out_real

Name: temp
Aliases: None
Type Of Entry: Variable
Description: This is a temporary storage area.
Make_up: Real
Source:
Destination:
Used In: matrxinv

Name: temp2
Aliases: None
Type Of Entry: Variable
Description: This is a temporary storage area.
Make_up: Real
Source:
Destination:
Used In: matrxinv

Name: temp_file
Aliases: None
Type Of Entry: Global file variable
Description: This file temporarily captures the user's instructions. This will allow the user to print out results after he/she has verified the results on the screen
Make_up: Text
Source: Declared in icecappc
Destination:
Used In: out_string, out_int

Name: TEMP.OUT
Type: File
Description: This file contains the user / SW transaction history (only the most recent).
Used In: get_data

```

*****
Name:          tempol
Aliases:       None
Type Of Entry: Variable
Description:   This is a temporary storage area for a polynomial.
Make_up:      polynomial
Source:
Destination:
Used In:      form
*****

```

```

*****
Name:          tempstr
Aliases:       None
Type Of Entry: Variable
Description:   This variable is a string used for temporary storage.
Make_up:      String[ 10 ]
Source:       svideolow,      svideobold
Destination:
Used In:      svideolow,      svideobold
*****

```

```

*****
Name:          term
Aliases:       None
Type Of Entry: Global file variable and data flow
Description:   This is an array that contains the terminal control
              codes. Additionally, it is an element of record type
              DATA.
Make_up:      term_array
Source:       Declared in icecappc
Destination:
Used In:      icecappc,      rectangle,      videobold,
              svideobold,   svideolow,      videolow,
              clearscreen,  clear,          gotoxy,
              nohighlight,  highlight,     nographics,
              graphics,     left_bracket,  right_bracket
*****

```

```

*****
Name:          term_array
Aliases:       None
Type Of Entry: Global type definition
Description:   Defines the type for the TERM data flow.
Make_up:      array[ 1.term_length ] of byte
Source:       Declared in icecappc
Destination:  N/A
Used In:      icecappc,      get_data
*****

```

```

*****
Name:      term_dat
Aliases:   None
Type Of Entry: Variable
Description: Is defined as term_array.
Make_up:   term_array
Source:    title_slide,  get_data
Destination: N/A
Used In:   title_slide,  get_data
*****

```

```

*****
Name:      TERMINAL.PAS
Type:      File
Description: This file contains the procedures that interface
            with the terminal.
Procedures Contained:  graphics,      nographics,      highlight,
                       nohighlight,   gotoxy,          clear,
                       clearscreen,   videolow,       svideolow,
                       videobold,     svideobold,     rectangle
Version:    3.0
Date:      12 Dec 84
Author:    Paul A Moore, Capt, USAF
*****

```

```

*****
Name:      terminal_only
Aliases:   None
Type Of Entry: Global constant and data flow
Description: Flag that indicates to get_string that input should
            come only from the terminal.
Make_up:   Char
Source:    Declared in icecappc
Destination:
Used In:
*****

```

```

*****
Name:      term_length
Aliases:   None
Type Of Entry: Global constant
Description: This is the length of the terminal data array.
Make_up:   Integer
Source:    Declared in msdwcons
Destination:
Used In:   icecappc,  msdwtype,  graphics,
           nographics, highlight,
           gotoxy,   clear,
           videolow, svideolow, videobold,

```

svideobold, get_data

Name: tf_file
Aliases: None
Type Of Entry: Variable
Description: This is a file of polynomial.
Make_up: File of polynomial
Source:
Destination:
Used In: move_tf

Name: third
Aliases: None
Type Of Entry: Variable
Description: This is either a polynomial or a matrix.
Make_up: polynomial or matrix
Source:
Destination:
Used In: ppoly, mmatrix

Name: title_slide
Type: Procedure
Description: This procedure displays the system title slide. It demonstrates that at a minimum the terminal control codes have been initialized properly.

Global Variables Used: None
Global Variables Changed: None
Global Constants Used: None
Passed Variables: term_dat
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: clear, gotoxy, highlight,
nohighlight, rectangle
Called By: get_data

Version: 3.0
Date: 22 Jul 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: GETDAT.PAS

```

*****
Name:      trans
Aliases:   None
Type Of Entry: Global variable
Description: This variable is the flag that indicates whether
              the transaction file is on. i.e. should all
              transactions be saved in a file.
Make_up:   Boolean
Source:    Declared in icecappc
Destination:
Used In:   icecappc,      out_string,      bld_stat_line,
              out_int,      get_data,      out_real
*****

```

```

*****
Name:      TRANSACTIONS
Type:      File
Description: This file contains the user / SW transaction
              history.
Used In:   get_data
*****

```

```

*****
Name:      trans_file
Aliases:   None
Type Of Entry: Global file variable
Description: This file contains the user / SW transaction history.
              It may be turned on and off by the user with trans.
Make_up:   Text
Source:    Declared in icecappc
Destination:
Used In:   out_string,      out_int
*****

```

```

*****
Name:      trans_file_name
Aliases:   None
Type Of Entry: Global variable
Description: The name of the file that contains the user / SW
              transaction history. It is turned on and off by the
              user.
Make_up:   Paramstring
Source:    Declared in ICECAPP
Destination:
Used In:   icecappc,      get_data
*****

```

```

*****
Name:      trim

```

Type: Procedure
 Description: This procedure removes any trailing characters from
 a cmdword.
 Global Variables Used: None
 Global Variables Changed: None
 Global Constants Used: wordsize
 Passed Variables: cmdword
 Returned: cmdword
 Files Read: None
 Files Written: None
 Aliases: None
 Procedures Called: None
 Called By: displa_commandword, readcom, val_n_dec,
 ccopyy, help, gettf,
 get_poly, matrxmanip1, matrxmanip2,
 define, disp_matrx, ppoly,
 chgmat, inroot, delroot,
 polymanip, polymanip2, get_poly_name,
 disppoly, select_routine, modify,
 disp, disptf, poly_into_storage,
 poly_from_storage, check_word, get_poly

Version: 1.1
 Date: 30 Jun 84
 Author: Paul A Moore, Capt, USAF
 Contained In File: TRIM.PAS

Name: TRIM.PAS
 Type: File
 Description: This file removes any trailing characters from
 a cmdword.

Procedures Contained: trim
 Version: 1.1
 Date: 30 Jun 84
 Author: Paul A Moore, Capt, USAF

Name: ttype
 Type: Procedure
 Description: This procedure acts as a switch between two different
 terminal types

Global Variables Used: None
 Global Variables Changed: None
 Global Constants Used: None
 Passed Variables: switch, wchar
 Returned: None

Files Read: None
Files Written: None
Aliases: None
Procedures Called: None
Called By: gotoxy

Version: 2.0
Date: 21 Oct 84
Author: Paul A Moore, Capt, USAF
Contained In File: TERMINAL.PAS

Name: tword
Aliases: None
Type Of Entry: Variable
Description: This variable is used a temporary storage location
for the command word while separating the individual
command words.

Make_up: msg_line
Source:
Destination:
Used In: readcom

Name: type_move
Aliases: None
Type Of Entry: Variable
Description: This variable is used as a flag to determine the
type to move that will be made: tf, poly, matrix.

Make_up: Char
Source:
Destination:
Used In: get_location, ccopy

Name: u
Aliases: None
Type Of Entry: Variable
Description: Used as a temporary storage location in roots.

Make_up: Real
Source:
Destination:
Used In: roots

Name: UCASE.PAS
Type: File
Description: This file contains the procedure to convert a lower case string to upper case.
Procedures Contained: ucase
Version: 1.1
Date: 28 Aug 83
Author: Vincent M. Parisi II, Capt, USAF

Name: ucase
Type: Procedure
Description: This procedure converts a lower case string to an upper case string.
Global Variables Used: None
Global Variables Changed: None
Global Constants Used: None
Passed Variables: instring
Returned: instring
Files Read: None
Files Written: None
Aliases: None
Procedures Called: None
Called By: readcom, chgmat, get_poly_name, get_matrx_name

Version: 1.1
Date: 28 Aug 83
Author: Vincent M. Parisi II, Capt, USAF
Contained In File: UCASE.PAS

Name: ui
Aliases: None
Type Of Entry: Variable
Description: Used as a temporary storage location in roots.
Make_up: Real
Source:
Destination:
Used In: roots

Name: update
Type: Procedure
Description: This procedure copies the ICECAP 'tf&pols.dat' and the 'matrix.dat' files into a user specified file.
Global Variables Used: abort_command

Global Variables Changed: None
Global Constants Used: as_assigned, blanks, crt_only,
Passed Variables: None
Returned: None
Files Read: TF&POLS.DAT, MATRIX.DAT
Files Written: user specified file name
Aliases: None
Procedures Called: clear, gotoxy, disp_msg,
get_strng, pause, clear_msg,
out_string
Called By: select_routine

Version: 2.0
Date: 19 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: UPDATE.PAS

Name: update_msg
Aliases: None
Type Of Entry: Variable
Description: This variable is the number of the help message for
the update command.

Make_up: Integer

Source:

Destination:

Used In: help

Name: UPDATE.PAS
Type: File
Description: This file copies the ICECAP 'tf&pols.dat' and
the 'matrix.dat' files into a user specified file.

Procedures Contained: update

Version: 2.0

Date: 19 Sep 85

Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

Name: v
Aliases: None
Type Of Entry: Variable
Description: Used as a temporary storage location in roots.
Make_up: Real

Source:
Destination:
Used In: roots

Name: val_n_dec
Type: Procedure

Description: This procedure validates and decodes the command line input by the user. The process begins by recovering record 1 from the syntax table, using getline. Compares to first word in the command buffer. If there is no error the process repeats itself until there is a valid command. If there are any anomalies a flag is set and a error message displayed to the user.

Global Variables Used: cmdbuffer, call_routine

Global Variables Changed: cmdbuffer, call_routine

Global Constants Used: DONEWORD, ENDCODE

Passed Variables: level, rec_num, error_code,
num_of_commands, cmdbuffer, call_routine

Returned: level, rec_num, error_code, num_of_commands
cmdbuffer, call_routine

Files Read: None

Files Written: None

Aliases: None

Procedures Called: check_word, get_line, trim

Called By: get_cmd

Version: 1.6

Date: 16 Aug 83

Author: Vincent M Parisi II, Capt, USAF

Contained In File: VALNDEC.PAS

Name: VALNDEC.PAS
Type: File

Description: This file validates and decodes the command line input by the user. The process begins by recovering record 1 from the syntax table, using getline. Compares to first word in the command buffer. If there is no error the process repeats itself until there is a valid command. If there are any anomalies a flag is set and a error message displayed to the user.

Procedures Contained: check_word, val_n_dec

Version: 1.7

Date: 29 Aug 83

Author: Vincent M Parisi II, Capt, USAF

Name: vi
Aliases: None
Type Of Entry: Variable
Description: Used as a temporary storage location in roots.
Make_up: Real
Source:
Destination:
Used In: roots

Name: videobold
Type: Procedure
Description: This procedure writes the char string to put the
screen into bold video.
Global Variables Used: term
Global Variables Changed: None
Global Constants Used: term_length
Passed Variables: None
Returned: None
Files Read: None
Files Written: None
Aliases: None
Procedures Called: None
Called By:

Version: 1.0
Date: 27 Sep 84
Author: Paul A. Moore, Capt, USAF
Contained In File: TERMINAL.PAS

Name: videolow
Type: Procedure
Description: This procedure writes the char string to put the
screen into low video.
Global Variables Used: term
Global Variables Changed: None
Global Constants Used: term_length
Passed Variables: None
Returned: None
Files Read: None
Files Written: None
Aliases: None

Procedures Called: None
Called By:

Version: 1.0
Date: 27 Sep 84
Author: Paul A. Moore, Capt, USAF
Contained In File: TERMINAL.PAS

Name: w
Aliases: None
Type Of Entry: Variable
Description: Generally used as a counter.
Make_up: Integer
Source:
Destination:
Used In: roots

Name: wchar
Aliases: None
Type Of Entry: Variable
Description: This is used as a switch between terminal types.
Make_up: Integer
Source:
Destination:
Used In: ttype

Name: width
Aliases: None
Type Of Entry: Variable
Description: This is the width of the triangle in columns.
Make_up: Integer
Source:
Destination:
Used In: rectangle

Name: wordlength
Aliases: None
Type Of Entry: Global constant
Description: This is the length of word in storage.
Make_up: Integer (9)
Source: Declared in msdwcons

Destination:
Used In: msdwtype, get_line

Name: word_num
Aliases: None
Type Of Entry: Variable
Description: Pointer to the word in the command buffer that is
to be displayed on the crt.
Make_up: Integer
Source: get_cmd, proces_error
Destination:
Used In: displa_commandword

Name: wordnumber
Aliases: None
Type Of Entry: Variable
Description: This is the number of words in the command buffer.
Make_up: Integer
Source:
Destination:
Used In: help, define, disp, mmatrix, modify,
ppoly, inroot, delroot, get_poly_name

Name: wordsize
Aliases: None
Type Of Entry: Global constant
Description: Size of string for internal command words
Make_up: Integer (12)
Source: Declared in ICECAPP
Destination: N/A
Used In: icecappc, get_line, trim,
val_n_dec, displa_command, help,
select

Name: wordtype
Aliases: None
Type Of Entry: Global type definition
Description: Type definition of a string of length wordlength.
Make_up: wordtype = string[wordlength]
Source: Declared in msdwtype
Destination:

Used In:

Name: yes
Aliases: None
Type Of Entry: Global constant
Description: Boolean variable
Make_up: Boolean
Source: Declared in ICECAPP
Destination: N/A
Used In:

Name: your_name
Aliases: None
Type Of Entry: Variable
Description: This is the user defined name for the disk files for
the storage of 'tf&pols.dat' and matrix.dat'.
Make_up: msg_line
Source:
Destination:
Used In: recover, update

Name: z
Aliases: None
Type Of Entry: Variable
Description: Generally used as a counter.
Make_up: Integer
Source:
Destination:
Used In: roots

Appendix E: System Software

Introduction

This file contains a cross index listing for procedures and source files. The source files are indicated by capital or upper case letters while the functions and procedures are indicated by the lower case letters.

BUILDDAT

This is the cross index for the menu installation program BUILDDAT. These files were not changed during this thesis effort, as a result, the code for the following BUILDDAT files is not included here. The code may be found in reference 15 and again in reference 12, Capt Parisi's MS Thesis and Capt Moore's MS Thesis respectively.

BUILDDAT.PAS

- make_param
- make_terminal
- make_printer
- make_help
- buildat

ADDKEY.PAS

- addkeywordtomenu

ADDTOMEN.PAS

- addtomenu

ADDWORD.PAS

- addword

AVL-1.PAS

- create
- isempty
- lchild
- rchild
- dataval

AVL-2.PAS

- makebt
- LNR
- displaytree
- printtree
- treedispose

AVL-3.PAS
leftbalance
rightbalance

AVL-4.PAS
avlinsert
btlocate

DEFCALL.PAS
findcall
addcall
define_call_routine

DEFMENU.PAS
define_menu

DISPROC.PAS
callsdispose

ERRORMSG.PAS
errormsg

GETWORD.PAS
get_word

LOCMENU.PAS
copymenuword
locate_menu_node

MAKEMENU.PAS
init_menu
make_menu

MAKEPROC.PAS
calc_min_chars
make_word_list
number_calls
make_call_records
make_keyw_records
make_decoding_paths

MENUCONS.PAS - Menu Constant Definition File

MENUPRNT.PAS
menu_print

MENUTYPE.PAS - Menu Type Definition File

MSDWCONS.PAS - MICROSDW.SYS Constant Def. File

MSDWTYPE.PAS - MICROSDW.SYS Type Def. File

READMENU.PAS
readmenu

ICECAP-PC

This list includes the modified user interface, MICROSDW files (12), as well as the ICECAP-PC files created as part of this thesis effort. The identification codes are as follows:

- The ICECAP-PC files are preceded by two asterisks (**).
- Any files that require different versions for the IBM-PC/AT, XT and the Z-100 is annotated with, Z-100 and IBM versions.
- Any file that is for the IBM-PC/AT/XT only is annotated with, IBM Unique.
- Any file that has different versions for a hard drive and a floppy system is annotated with, Hard and Floppy.

The files are presented in the order of this cross index.

```
** BODE.PAS
    frequency_response

** CONCONS.PAS - ICECAP-PC Constant Definition File

** COPY.PAS
    get_location
    move_tf
    move_poly
    move_matrix
    ccopy

** DEFINE.PAS
    get_poly
    define

** DELROOT.PAS
    delroot

** DISP.PAS
    disptf
    disp

DISPLAYC.PAS
    displa_commandword
```

```

** FORM.PAS
    poly_from_storage
    poly_into_storage
    form

GETCOM.PAS
    get_cmd

** GETDAT.PAS ( Z-100 and IBM versions )
    title_slide
    bld_stat_line
    get_data

GETINT.PAS
    getchi
    del_lst_ch
    ck_chr
    out_int
    get_int

GETLINE.PAS
    get_line

** GETMAT.PAS
    left_bracket
    right_bracket
    make_pretty_large_matrix_one
    make_pretty_large_matrix_two
    get_matrix_entries
    make_pretty_small_matrix
    getmat

GETSTRIN.PAS
    get_string

** GETTF.PAS
    get_r_num
    make_pretty
    get_fact
    form_poly
    get_unfact
    roots
    poly
    gettf

** HELP.PAS
    help

```

```

** ICECAPPC.PAS ( Z-100 and IBM-unique versions )
                ( Hard and Floppy )
                icecappc

** INROOT.PAS
                inroot

INSTRUCT.PAS
                instruction

** MATRIX.PAS
                disp_matrx
                matrx_manip1
                matrx_manip2
                get_matrx_name
                mmatrix

** MATRXMAN.PAS
                matrxadd
                matrxsub
                mmatrxmlt
                smatrxmlt
                matrxtran
                matrxinv

** MODIFY.PAS
                chgmat
                modify

MSDWCONS.PAS - MICROSDW.SYS Constant Def. File
MSDWTYPER.PAS - MICROSDW.SYS Type Def. File

MSG.PAS
                disp_line
                clear_msg
                disp_msg

OUTPUT.PAS
                out_string

PAUSE.PAS
                pause

```

```

** POLYMAN.PAS
    polyadd
    polysub
    polymlt
    spolymlt

** POLY.PAS
    disppoly
    polmanip
    polmanip2
    get_poly_name
    ppoly

PROCESER.PAS
    proces_error

PROMPTCM.PAS
    prompt_cmd

PROMPHE.PAS
    prompt_help

READCOM.PAS
    readcom

** REALS.PAS
    out_real
    get_real

** RECOVER.PAS
    recover

** SELECT.PAS
    select_routine

** STDOUT.PAS ( IBM Unique )
    standard_output

TERMINAL.PAS
    graphics
    nographics
    highlight
    nohighlight
    gotoxy
    clear
    ClearScreen
    VideoLow
    SVideoLow
    VideoBold

```

SVideoBold
Rectangle

TRIM.PAS

trim

UCASE.PAS

ucase

** UPDATE.PAS

update

VALNDEC.PAS

check_word
val_n_dec

```

*****
**
** file: BODE.PAS
**
** procedures contained: frequency_response
**
** version: TEST
**
** date: 27 September 1985
**
** description: This file contains all the procedures for
** frequency response calculations.
** Gary C. Tarczynski, Capt, USAF
** Susan K. Mashiko, Capt, USAF
**
**
*****
*****
**
** procedure: frequency_response
**
** version: TEST
**
** date: 27 September 1985
**
** description: This procedure acts as the executive for
** frequency response calculations. First,
** it prompts the user to choose units for
** frequency, magnitude, and phase angle.
** Then, it calls the appropriate functions
** and subroutines to perform the calculations.
** Finally, it displays the data in a tabular
** format.
**
** global variables used:
** global variables changed:
** global constants used:
** passed variables:
** returned variables:
** files created:
** files read:
** files written:
** procedures called:
** called by: Gary C. Tarczynski, Capt, USAF
** author: Susan K. Mashiko, Capt, USAF
**
*****
*****

```

```

procedure frequency_response;

```

```

var
  i : integer;

```

```

begin

```

```

  ClearScreen;

```



```
gotoxy( 5, 0 );
out_string( 'FREQUENCY RESPONSE', crt_only );
gotoxy( 7, 0 );
out_string( 'is not implemented yet.', crt_only );
pause;
ClearScreen;
exit;

end;
```

FILE: BUDE.PAS

```
(*****  
**  
** file: CONCONS.PAS  
** version: 1.0  
** date: 26 August 85  
** description: Contains the constants, type, and var  
** declarations for the controls procedures  
** and functions.  
** author: Susan K. Mashiko, Capt, USAF  
** Gary C. Tarczynski, Capt, USAF  
**  
**  
**  
**  
***)
```

```
const  
  max_deg = 10;  
  max_deg1 = 11;  
  max_rows = 10;  
  max_cols = 10;  
  pi = 3.1415926536;
```

```
type  
  short_int = integer;
```

```
  complex = record  
    realpart : real;  
    imagpart : real;  
  end;
```

```
  polynomial = record  
    change : boolean;  
    coefficient : real;  
    polydeg : integer;  
    polyfact : array[ 1..max_deg ] of complex;  
    polypoly : array[ 1..max_deg1 ] of real;  
  end;
```

```
  matrix = record  
    num_rows : integer;  
    num_cols : integer;  
    element : array[ 1..max_rows, 1..max_cols ] of real;  
  end;
```

```
var  
  degree : integer;  
  degree1 : integer;
```

FILE: CONCONS.PAS

```

*****
**
** file: COPY.PAS
**
** procedures contained: get_location, move_tf, move_poly,
**                      move_matrix, ccopy
**
** version: 3.0
** date: 22 September 1985
** description: This file contains the procedures that
**              will copy data from one transfer function
**              to another, from one polynomial to
**              another, or from one matrix to another.
**              vincent m. parisi ii, capt, usaf
**              Gary C. Tarczynski, Capt, USAF
**              Susan K. Mashiko, Capt, USAF
**
**
**
*****

```

```

*****
**
** procedure: get_location
** version: 2.0
** date: 04 September 1985
** description: This procedure determines the record
**              location of the source and destination
**              objects for a copy function.
**
** passed variables: location, rec_loc, type_move
**
** returned variables: rec_loc, type_move
**
** called by: ccopy
**
** author: vincent m. parisi ii, capt, usaf
**
** modified by: Gary C. Tarczynski, Capt, USAF
**              Susan K. Mashiko, Capt, USAF
**
** mod description: The entire procedure was modified to
**                  customize it for ICECAP.
**
** mod date: 04 September 1985
**
**
*****

```

```

overlay procedure get_location( location:cmdword; var rec_loc:integer;
                               var type_move:char);

```

```

begin
  if location = 'OLTF' then
  begin
    rec_loc := 0;
    type_move := 'T';
  end;
end;

```

FILE: COPY.PAS

```
if location = 'CLTF' then
begin
rec_loc := 2;
type_move := 'T';
end;

if location = 'GTF' then
begin
rec_loc := 4;
type_move := 'T';
end;

if location = 'HTF' then
begin
rec_loc := 6;
type_move := 'T';
end;

if location = 'TF1' then
begin
rec_loc := 8;
type_move := 'T';
end;

if location = 'TF2' then
begin
rec_loc := 10;
type_move := 'T';
end;

if location = 'TF3' then
begin
rec_loc := 12;
type_move := 'T';
end;

if location = 'TF4' then
begin
rec_loc := 14;
type_move := 'T';
end;

if location = 'TF5' then
begin
rec_loc := 16;
type_move := 'T';
end;

if location = 'ONPOLY' then
```

FILE: COPY.PAS

```
begin
  rec_loc := 0;
  type_move := 'P';
end;

if location = 'ODPOLY' then
begin
  rec_loc := 1;
  type_move := 'P';
end;

if location = 'CNPOLY' then
begin
  rec_loc := 2;
  type_move := 'P';
end;

if location = 'CDPOLY' then
begin
  rec_loc := 3;
  type_move := 'P';
end;

if location = 'GNPOLY' then
begin
  rec_loc := 4;
  type_move := 'P';
end;

if location = 'GDPOLY' then
begin
  rec_loc := 5;
  type_move := 'P';
end;

if location = 'HNPOLY' then
begin
  rec_loc := 6;
  type_move := 'P';
end;

if location = 'HDPOLY' then
begin
  rec_loc := 7;
  type_move := 'P';
end;

if location = 'POLYA' then
begin
```

FILE: COPY.PAS

```
rec_loc := 18;
type_move := 'p';
end;

if location = 'POLYB' then
begin
rec_loc := 19;
type_move := 'p';
end;

if location = 'POLYC' then
begin
rec_loc := 20;
type_move := 'p';
end;

if location = 'POLVD' then
begin
rec_loc := 21;
type_move := 'p';
end;

if location = 'POLVE' then
begin
rec_loc := 22;
type_move := 'p';
end;

if location = 'MATA' then
begin
rec_loc := 0;
type_move := 'M';
end;

if location = 'MATB' then
begin
rec_loc := 1;
type_move := 'M';
end;

if location = 'MATC' then
begin
rec_loc := 2;
type_move := 'M';
end;

if location = 'MATD' then
begin
rec_loc := 3;
```

FILE: COPY.PAS

```

type_move := 'M';
end;

if location = 'MATE' then
begin
rec_loc := 4;
type_move := 'M';
end;

end;

(*****
*
* procedure:      move_tf
* version:       2.0
* date:         04 September 1985
* description:   This procedure receives the source and
*               destination transfer function locations,
*               reads the source and moves it to the
*               destination.
* passed variables:  source_loc, dest_loc
* files read:      TF&POLS.DAT
* files written:   TF&POLS.DAT
* called by:       ccopy
* author:         vincent m. parisi ii, capt, usaf
* modified by:     Gary C. Tarczynski, Capt, USAF
*                 Susan K. Mashiko, Capt, USAF
* mod description: Converted from Pascal MT+ to TURBO
*                 pascal.
* mod date:       04 September 1985
*
* *****)
overlay procedure move_tf( source_loc:integer; dest_loc:integer );
var
tf_file      :   file of polynomial;
poly         :   polynomial;
i            :   integer;
begin
assign( tf_file, 'TF&POLS.DAT' );
reset( tf_file );
seek( tf_file, source_loc );
read( tf_file, poly );
seek( tf_file, dest_loc );

```

FILE: COPY.PAS


```

close( poly_file );
end;
(*****
*
* procedure:      move_matrix
* version:       2.0
* date:         04 September 1985
* description:   This procedure receives the source and
*               destination matrix locations, reads the
*               source and copies it to the destination.
*
* passed variables:  source_loc, dest_loc
* files read:      MATRIX.DAT
* files written:   MATRIX.DAT
* called by:      ccopy
* author:         Vincent M. Parisi, Lt. Capt, USAF
* modified by:    Gary C. Tarczynski, Capt, USAF
*               Susan K. Mashiko, Capt, USAF
* mod description: Converted from Pascal MT+ to TURBO
*               Pascal.
* mod date:      04 September 1985
*
* *****)
overlay procedure move_matrix( source_loc:integer; dest_loc:integer );
var
mat_file : file of matrix;
matrix : matrix;
i : integer;
begin
assign( mat_file, 'MATRIX.DAT' );
reset( mat_file );
seek( mat_file, source_loc );
read( mat_file, matrix );
seek( mat_file, dest_loc );
write( mat_file, matrix );
close( mat_file );
end;
(*****
*
```

FILE: COPY.PAS

```

*
* procedure:      ccopy (ie copy, called ccopy due to
*                pascal compiler having internal function
*                called copy)
*                2.0
*                22 September 1985
*
* version:
*
* date:
*
* description:   This procedure manages the copy function.
*                It determines the source and destination
*                locations of the object to be copied, and
*                then performs the copy operation.
*
* global variables used:  cmdbuffer
*
* Global constants used:  as_assigned
*
* passed variables:
*
* procedures called:
*
*                clear, gotoxy, trim, out_string,
*                get_location, move_matrix,
*                move_poly, move_tf, highlight,
*                nonhighlight, pause, disp_msg
*
* called by:
*
* author:
*
* modified by:
*
* mod description: Code was added to check if the source
*                and destination are the same type. If
*                not, the copy is not performed, an error
*                message is issued, and the user is exited
*                back to the main ICECAP menu.
*
* mod date:      04 September 1985
*
* mod description: Code was added to for a help option
*
* mod date:      22 September 85
*
* *****
*
* procedure ccopy( var cmdbuffer : buffer );
*
* var
*
* i          : integer;
* source     : cmdword;
* destination : cmdword;
* type_move  : char;
* dest_loc  : integer;
* sorce_loc  : integer;
*
* (**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**
*
* The following variables were added by
* Mashiko and Tarczynski to compare the
* source and destination types.*)
*
* type_move  : char;

```

FILE: COPY.PAS

```

type_move2 : char;

(**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**)
begin
  gotoxy(15,30);
  source := cndbuffer[2];
  trim( source );
  if source = 'HELP' then
    begin
      clear;
      disp_msg( 17 );
      pause;
      clear;
      exit;
    end
  else
    destination := cndbuffer[3];
    trim( destination );
    out_string( 'Copying ', as_assigned );
    out_string( source, as_assigned );
    out_string( ' --> ', as_assigned );
    out_string( destination, as_assigned );
  (**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**)

  The following code was added by Mashiko
  and Tarczynski to check if the source
  and destination are the same type. If
  not, an error message is issued and the
  copy is not performed.*)

  get_location( source, sorce_loc, type_move );
  type_move1 := type_move;
  get_location( destination, dest_loc, type_move );
  type_move2 := type_move;

  (* issue error message if types do not match *)
  begin
    gotoxy(16,20);
    highlight;
    out_string( 'Mismatched types. Copy not performed.', as_assigned );
    nohighlight;
    pause;
    exit;
  end
  else

```

FILE: COPY.PAS

```
(***INSERT***INSERT***INSERT***INSERT***INSERT***INSERT***INSERT***INSERT***)
if type_move = 'T' then
  move_if( source_loc, dest_loc )
else
  if type_move = 'P' then
    move_poly( source_loc, dest_loc )
  else
    move_matrix( source_loc, dest_loc );
  pause;
  gotoxy(15,0);
  out_string( blanks, crt_only );
end;
```

FILE: COPY.PAS

```

*****
**
** file: DEFINE.PAS
** procedures contained: get_poly, define
** version: 1.0
** date: 22 Sep 85
** description: This file contains the procedures that
** handle the logic for the definition of
** various inputs: transfer function, matrix,
** polynomial. The various input routines
** are called.
** author: Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt, USAF
**
*****
)

```

```

*****
**
** procedure: get_poly
** version: 1.0
** date: 28 Aug 85
** description: This procedure will input a polynomial in either
** factored or polynomial form.
** global variables used: abort_command
** global constants used: crt_only, as_assigned,
** max_deg
** passed variables: def_obj, method
** files written: TF&POLS.DAT
** procedures called: clear, gotoxy,
** disp_msg, out_string,
** get_int, disppoly,
** trim.
**
** called by: define
** author: Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt, USAF
**
*****
)

```

```

procedure get_poly( var def_obj : cmdword; var method : cmdword );

```

```

var
  pol_deg : short int;
  abort_command : boolean;
  stor_loc : integer;
  disp_row : integer;
  pol : polynomial;
  polys : file of polynomial;

```

```

FILE: DEFINE.PAS

```

```

i      : integer;

begin
  abort_command := false;
  clear;
  gotoxy( 10, 5 );
  disp_msg( 30 );

  (* get the order of the polynomial *)
  repeat
  begin
    gotoxy( 10, 58 );
    out_string( ' ', crt_only );
    gotoxy( 10, 58 );
    get_int( pol_deg, abort_command );
    if abort_command then exit;
  end;
  until (( pol_deg >= 0 ) and ( pol_deg <= max_deg ));

  clear;

  gotoxy( 0, 30 );
  disp_msg( 31 );
  gotoxy( 1, 36 );
  out_string( def_obj, as_assigned );

  (* get the polynomial *)
  gotoxy( 2, 34 );
  disp_msg( 32 );
  disp_row := 3;
  trim( method );
  poly( method, pol_disp_row, abort_command );
  if abort_command then exit;

  (* determine the storage areas for polynomials *)
  if def_obj = 'POLYA' then stor_loc := 18
  else
  if def_obj = 'POLVB' then stor_loc := 19
  else
  if def_obj = 'POLVC' then stor_loc := 20
  else
  if def_obj = 'POLVD' then stor_loc := 21
  else
  if def_obj = 'POLVE' then stor_loc := 22
  else
  if def_obj = 'ONPOLY' then stor_loc := 0
  else
  if def_obj = 'ODPOLY' then stor_loc := 1

```

FILE: DEFINE.PAS

```

else
  if def_obj = 'CNPOLY' then stor_loc := 2
else
  if def_obj = 'CDPOLY' then stor_loc := 3
else
  if def_obj = 'GNPOLY' then stor_loc := 4
else
  if def_obj = 'GDPOLY' then stor_loc := 5
else
  if def_obj = 'HNPOLY' then stor_loc := 6
else
  if def_obj = 'HDPOLY' then stor_loc := 7;

(* save polynomial to the file *)
assign( polys, 'f&polis.dat' );
reset( polys );
seek( polys, stor_loc );
write( polys, pot );
close( polys );

disppoly( def_obj );
pause;

```

```
end;
```

```

(*****
* procedure: define
* version: 2.0
* date: 22 Sep 85
* description: This procedure will call the correct subroutines
* for the definition of matrices, transfer functions
* and polynomials.
* global variables used: cmdbuffer, string
* global variables changed: string
* passed variables: cmdbuffer, wordnumber
* procedures called: trim, clear, get_mat, pause,
* get_tf, get_poly, get_msg
* authors: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
* mod description: Added the code for a help option in the primary
* menu level in define.
* modifier: Author
* mod date: 22 Sep 85
*
*****
procedure define( var cmdbuffer : buffer;

```

```
FILE: DEFINE.PAS
```

```

wordnumber : integer );

var
  def_obj : cmdword;
  check   : cmdword;

begin
  def_obj := cmdbuffer[ 2 ];
  trim( def_obj );
  clear;

  (* Code for help function *)
  check := cmdbuffer[ 3 ];
  trim( check );
  if check = 'HELP' then
    begin
      clear;
      disp_msg( 28 );
      pause;
      clear;
      exit;
    end;

  (* The following code call the transfer function input procedure *)
  if ( ( def_obj = 'OLTF' ) or ( def_obj = 'CLTF' ) or ( def_obj = 'GTF' ) or
      ( def_obj = 'HTF' ) or ( def_obj = 'TF1' ) or ( def_obj = 'TF2' ) or
      ( def_obj = 'TF3' ) or ( def_obj = 'TF4' ) or ( def_obj = 'TFS' ) ) then
    get_tf( def_obj, cmdbuffer[ 3 ] );

  (* Code for the help function *)
  if def_obj = 'HELP' then
    begin
      clear;
      disp_msg( 18 );
      pause;
      clear;
    end;

  (* The following code calls the matrix input procedure if the
  first three letters of def_obj are MAT *)
  string := copy( def_obj, 1, 3 );
  if string = 'MAT' then
    get_mat( def_obj );

  (* The following code calls the polynomial input procedure
  if the first four letters of def_obj are POLY *)
  string := copy( def_obj, 1, 4 );
  if string = 'POLY' then

```

FILE: DEFINE.PAS


```

get_poly( def_obj, cmbuffer[ 3 ] );

(* The following code calls the polynomial input procedure
   if the last four letters of def_obj are POLY *)
string := copy( def_obj, 3, 6 );
if string = 'POLY' then
  get_poly( def_obj, cmbuffer[ 3 ] );

(*DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE*

This section of Parisi's code was deleted by
Mashiko and Tarczynski. If a commandword
contains the string 'TF', then input is han-
dled by the get_tf procedure above. These
three lines of code are not needed.

string := copy( def_obj, 1, 2 );
if string = 'TF' then
  get_poly( def_obj, cmbuffer[ 3 ] );

*DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE*)
end;

```

FILE: DEFINE.PAS

```

*****
**
** file: DELROOT.PAS
** procedure contained: delroot
** version: 2.0
** date: 19 September 85
** description: This file contains the procedures to delete
** a root from a polynomial. If the root is com-
** plex the conjugate will also be removed.
** Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt, USAF
**
*****

```

```

*****
**
** procedure: delroot
** version: 2.0
** date: 22 September 85
** description: This procedure will delete a root from a polyno-
** mial. If the root is complex the conjugate will
** also be removed.
** global variables used: blanks, cmdbuffer, abort_command
** global constants used: crt_only, as_assigned
** passed variables: cmdbuffer, wordnumber
** files read: TF&POLS.DAT
** files written: TF&POLS.DAT
** procedures called:
** clear, gotoxy, disppoly,
** trim, highlight, out_string,
** nohighlight, get_int, disp_msg,
** pause, clear_msg, form_poly
**
** called by: select
** author: Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt, USAF
** modified by: author
** mod description: Code changed to compensate for the call being
** from lower level in menu structure. And changed
** message.
** mod date: 22 Sep 85
**
*****

```

```

overlay procedure delroot( var cmdbuffer : buffer;
var wordnumber : integer );

```

```

label once_more;

```

```

FILE: DELROOT.PAS

```

```

var
  choice      : cmdword;
  polys      : file of polynomial;
  i          : integer;
  oldpoly    : polynomial;
  newpoly    : polynomial;
  stor_loc   : integer;
  root_num   : integer;

begin
  clear;
  choice := cndbuffer[ 3 ];
  trim( choice );
  if choice = 'POLVA' then stor_loc := 18
  else
  if choice = 'POLVB' then stor_loc := 19
  else
  if choice = 'POLVC' then stor_loc := 20
  else
  if choice = 'POLVD' then stor_loc := 21
  else
  if choice = 'POLVE' then stor_loc := 22
  else
  if choice = 'ONPOLY' then stor_loc := 0
  else
  if choice = 'ODPOLY' then stor_loc := 1
  else
  if choice = 'CNPOLY' then stor_loc := 2
  else
  if choice = 'CDPOLY' then stor_loc := 3
  else
  if choice = 'GNPOLY' then stor_loc := 4
  else
  if choice = 'GDPOLY' then stor_loc := 5
  else
  if choice = 'HNPOLY' then stor_loc := 6
  else
  if choice = 'HDPOLY' then stor_loc := 7;

  (* pull the desired polynomial from storage *)
  assign( polys, 'tf&polys.dat' );
  reset( polys );
  seek( polys, stor_loc );
  read( polys, oldpoly );
  close( polys );

  (* display the polynomial and on the same screen ask *)
  (* which root should be eliminated *)
  disppoly( choice );

```

FILE: DELROOT.PAS

```

gotoxy( 20, 0 );
out_string( blanks, crt_only );
gotoxy( 20, 5 );
highlight;
out_string( 'The number of the root you wish to delete is....',
           as_assigned );
nonhighlight;
once_more;
gotoxy( 20, 52 );
out_string( ' ', crt_only );
gotoxy( 20, 53 );
get_int( root_num, abort_command );
if abort_command then exit;
if ( ( root_num > oldpoly.polydeg ) or ( root_num < 0 ) ) then
  begin
    gotoxy( 21, 5 );
    disp_msg( 9 );
    pause;
    gotoxy( 21, 0 );
    clear_msg( 9 );
    goto once_more;
  end;

```

```

(* if the root is complex you want to eliminate both parts *)
if oldpoly.polyfact[ root_num ].imagpart < 0 then
  begin
    for i := 1 to ( root_num - 1 ) do
      begin
        newpoly.polyfact[ i ].realpart :=
          oldpoly.polyfact[ i ].realpart;
        newpoly.polyfact[ i ].imagpart :=
          oldpoly.polyfact[ i ].imagpart;
      end;
    if root_num >= ( oldpoly.polydeg - 2 ) then
      begin
        for i := root_num to ( oldpoly.polydeg - 2 ) do
          begin
            newpoly.polyfact[ i ].realpart := oldpoly.polyfact[ i + 2
            ].realpart;
            newpoly.polyfact[ i ].imagpart := oldpoly.polyfact[ i + 2
            ].imagpart;
          end;
        end;
        newpoly.polydeg := oldpoly.polydeg - 2;
      end;
    if oldpoly.polyfact[ root_num ].imagpart > 0 then
      begin
        for i := 1 to ( root_num - 2 ) do

```

FILE: DELROOT.PAS

```

begin
  newpoly.polyfact[ i ].realpart :=
    oldpoly.polyfact[ i ].realpart;
  newpoly.polyfact[ i ].imagpart :=
    oldpoly.polyfact[ i ].imagpart;
end;
if root_num >= ( oldpoly.polydeg - 2 ) then
begin
  for i := ( root_num - 1 ) to ( oldpoly.polydeg - 2 ) do
    begin
      newpoly.polyfact[ i ].realpart :=
        oldpoly.polyfact[ i + 2 ].realpart;
      newpoly.polyfact[ i ].imagpart :=
        oldpoly.polyfact[ i + 2 ].imagpart;
    end;
  end;
  newpoly.polydeg := oldpoly.polydeg - 2;
end;
(* if there is no imaginary part only eliminate a single root *)
if oldpoly.polyfact[ root_num ].imagpart = 0 then
begin
  for i := 1 to ( root_num - 1 ) do
    begin
      newpoly.polyfact[ i ].realpart := oldpoly.polyfact[ i ].realpart;
      newpoly.polyfact[ i ].imagpart := oldpoly.polyfact[ i ].imagpart;
    end;
  end;
  for i := root_num to ( oldpoly.polydeg - 1 ) do
    begin
      newpoly.polyfact[ i ].realpart := oldpoly.polyfact[ i + 1 ].realpart;
      newpoly.polyfact[ i ].imagpart := oldpoly.polyfact[ i + 1 ].imagpart;
    end;
  end;
  newpoly.polydeg := oldpoly.polydeg - 1;
end;
newpoly.coefficient := oldpoly.coefficient;
(* form the polynomial *)
form_poly( newpoly );
(* store the new polynomial in the same stor_loc as the old *)
assign( polys, 'tfspols.dat' );
reset( polys );
seek( polys, stor_loc );
write( polys, newpoly );
(* display the new polynomial *)
disppoly( choice );
pause;

```

FILE: DELROOT.PAS

end:

FILE: DELROOT.PAS

```

*****
**
** file: DISP.PAS
** procedures contained: disp, disptf
** version: 3.0
** date: 6 November 85
** description: This file contains the procedures that handle
** the logic to display the various DISPLAY
** options on the screen.
** authors: Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt, USAF
**
*****
*****
*****

```

```

*****
**
** procedure: disptf
** version: 1.0
** date: 25 September 1985
** description: This procedure will display the requested trans-
** fer function on the screen. The displayed func-
** tion shall be taken from the designated storage
** location.
** global constants used: as_assigned
** passed variables: disp_obj
** files read: TF&POLS.DAT
** procedures called: trim, gotoxy,
** disp_msg, out_string,
** make_pretty, out_real,
** clear, pause
** called by: disp
** authors: Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt, USAF
**
*****

```

```

procedure disptf: (* forward referenced from gettf *)
(* ( var disp_obj : cmdword ); *)

```

```

var
stor_loc : integer;
numerator : polynomial;
denominator : polynomial;
polys : file of polynomial;
i : integer;
row : integer;
number : real;

```

FILE: DISP.PAS

```

begin
  if disp_obj = 'OLTF' then stor_loc := 0
  else
  if disp_obj = 'CLTF' then stor_loc := 2
  else
  if disp_obj = 'GTF' then stor_loc := 4
  else
  if disp_obj = 'HTF' then stor_loc := 6
  else
  if disp_obj = 'TF1' then stor_loc := 8
  else
  if disp_obj = 'TF2' then stor_loc := 10
  else
  if disp_obj = 'TF3' then stor_loc := 12
  else
  if disp_obj = 'TF4' then stor_loc := 14
  else
  if disp_obj = 'TF5' then stor_loc := 16;

  assign( polys, 'tf&polys.dat' );
  reset( polys );
  seek( polys, stor_loc );
  read( polys, numerator );
  seek( polys, (stor_loc + 1));
  read( polys, denominator );
  close( polys );

  (* put the title on the first page of the display *)
  gotoxy( 0, 27 );
  disp_msg( 33 );
  gotoxy( 1, 37 );
  out_string( disp_obj, as_assigned );
  gotoxy( 2, 34 );
  disp_msg( 6 );
  row := 3;

  (* draw the form on the screen *)
  make_pretty( row, numerator.polydeg );
  i := 1;

  (* get the term's coefficient and display it *)
  number := numerator.coefficient;
  gotoxy( ( row + 2 ), 19 );
  out_real( number, 12, as_assigned );
  gotoxy( ( row + 2 ), 57 );
  out_real( number, 12, as_assigned );

  (* get the numerator and display in factored form *)
  while i <= numerator.polydeg do

```

FILE: DISP.PAS


```

begin
  gotoxy( ( row + 3 + i ), 43 );
  out_real( numerator.polyfact[ i ].realpart, 12, as_assigned );
  gotoxy( ( row + 3 + i ), 59 );
  out_real( numerator.polyfact[ i ].imagpart, 12, as_assigned );
  i := i + 1;
end; (* end of while 'loop *)

(* now display the polynomial form of the numerator *)
i := 1;
while i <= ( numerator.polydeg + 1 ) do
  begin
    gotoxy( ( row + 3 + i ), 7 );
    out_real( numerator.polypoly[ i ], 12, as_assigned );
    i := i + 1;
  end;
pause;

(* if the numerator and denominator degrees combined are greater *)
(* than 7 the denominator will be displayed on the next page *)
if ( numerator.polydeg + denominator.polydeg ) <= 7 then
  begin
    row := numerator.polydeg + 9;
    gotoxy( ( row - 1 ), 33);
  end
else
  begin
    clear;
    gotoxy( 1, 36);
    out_string( disp_obj, as_assigned );
    gotoxy( 2, 33 );
    row := 3;
  end;

(* get the denominator of the transfer function *)
disp_msg( 7 );
make_pretty( row, denominator.polydeg );

(* get the term's coefficient and display it *)
number := denominator.coefficient;
gotoxy( ( row + 2 ), 19 );
out_real( number, 12, as_assigned );
gotoxy( ( row + 2 ), 57 );
out_real( number, 12, as_assigned );

(* get the denominator and display in factored form *)
i := 1;
while i <= denominator.polydeg do
  begin

```

FILE: DISP.PAS

```

gotoxy( row + 3 + i , 43 );
out_real( denominator.polyfact[ i ].realpart, 12, as_assigned );
gotoxy( row + 3 + i , 59 );
out_real( denominator.polyfact[ i ].imagpart, 12, as_assigned );
i := i + 1;
end; (* end of while loop *)

```

```

(* now display the polynomial form of the denominator *)
i := 1;
while i <= ( denominator.polydeg + 1 ) do
begin
gotoxy( row + 3 + i , 7 );
out_real( denominator.poly[ i ], 12, as_assigned );
i := i + 1;
end;
pause;
end;

```

```

(*****
* procedure: disp
* version: 2.0
* date: 6 November 85
* description: This procedure contains the logic to display the
* selected DISPLAY option on the screen.
* global variables used: cmdbuffer
* passed variables: cmdbuffer, wordnumber
* procedures called: disptf, trim, ppoly, mmatrix,
* clear, disp_msg, pause
* called by: select
* authors: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
* mod description: tells the user when an option is not available
* mod date: 6 November 85
*****
procedure disp( var cmdbuffer : buffer;
var wordnumber : integer);
var
disp_obj : cmdword;
begin
disp_obj := cmdbuffer[ 2 ];
trim( disp_obj );
clear;
(* catch code for the display of transfer functions *)

```

```
if ( ( disp_obj = 'OLTF' ) or ( disp_obj = 'CLTF' ) or
    ( disp_obj = 'GTF' ) or ( disp_obj = 'HTF' ) ) then
    dispf( disp_obj )
else
    if disp_obj = 'POLY' then
        ppol( cmdbuffer, number_of_commands )
    else
        if disp_obj = 'MATRIX' then
            mmatrix( cmdbuffer, number_of_commands )
        else
            if disp_obj = 'HELP' then
                begin
                    clear;
                    disp_msg( 19 );
                    pause;
                    clear;
                end
            else
                begin
                    clear;
                    gotoxy(8,30);
                    out_string(disp_obj, as_assigned);
                    gotoxy(10,17);
                    out_string('This routine is not implemented yet.', as_assigned);
                    pause;
                    exit;
                end;
            end;
        end;
    end;
```

FILE: DISP.PAS

```

*****
** file: DISPLAYC.PAS
** procedures contained: display_commandword
** version: 1.1
** date: 30 June 1984
** description: This file displays the commandword
** pointed to by word_num.
** author: vincent m. parisi ii, capt., usaf
**
*****
*****
** procedure: displa_commandword
** version: 1.1
** date: 30 June 1984
** description: this procedure displays the commandword
** pointed to by word_num.
** global variables used: cmdbuffer
** global constants used: wordsize, buffersize
** procedures called: outstring, trim
** called by: get_com
** author: vincent m. parisi ii, capt., usaf
** modifier: Paul A. Moore, Capt, USAF
**
*****
procedure displa_commandword(cmdbuffer : buffer; word_num : integer );
var cmd_word : cmdword;
begin
  cmd_word := cmdbuffer[ word_num ];
  trim( cmd_word );
  out_string( cmd_word, 'a' );
  out_string( ' ', 'a' );
end;

```

FILE: DISPLAYC.PAS

```

*****
** file: FORM.PAS
** procedures contained: form, poly_into_storage,
** poly_from_storage
** version: 1.0
** date: 7 October 85
** description: This file contains the procedures to form OLF's
** and CLTF's
** author: Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt, USAF
*****

```

```

*****
** procedure: poly_from_storage
** version: 1.0
** date: 7 October 85
** description: This file contains the procedures to get poly-
** nomial from storage
** passed variables: choice, pol
** returned variables: pol
** files read: TF&POL.S.DAT
** procedures called: trim
** called by: form
** author: Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt, USAF
*****

```

```

procedure poly_from_storage( var choice : cmdword ;
var pol : polynomial );

```

```

var
  polys : file of polynomial;
  stor_loc : integer;
begin
  trim( choice );
  if choice = 'POLVA' then stor_loc := 18
  else
  if choice = 'POLVB' then stor_loc := 19
  else
  if choice = 'POLVC' then stor_loc := 20
  else
  if choice = 'POLVD' then stor_loc := 21

```

FILE: FORM.PAS

```

else
if choice = 'POLVE' then stor_loc := 22
else
if choice = 'ONPOLY' then stor_loc := 0
else
if choice = 'ODPOLY' then stor_loc := 1
else
if choice = 'CNPOLY' then stor_loc := 2
else
if choice = 'CDPOLY' then stor_loc := 3
else
if choice = 'GNPOLY' then stor_loc := 4
else
if choice = 'GDPOLY' then stor_loc := 5
else
if choice = 'HNPOLY' then stor_loc := 6
else
if choice = 'HDPOLY' then stor_loc := 7;

assign( polys, 'tf&polys.dat' );
reset( polys );
seek( polys, stor_loc);
read( polys, pol );

close( polys );

```

```
end;
```

```

(*****
*
* procedure: poly_into_storage
* version: 1.0
* date: 7 October 85
* description: This file contains the procedures to place a poly-
* nomial into storage
* passed variables: choice, pol
* returned variables: pol
* files written: TF&POLS.DAT
* procedures called: trim
* called by: form
* author: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
*
*****
)

```

```

procedure poly_into_storage( var choice : cmdword ;
var pol : polynomial );

```

```
var
```

```
FILE: FURM.PAS
```

```
      file of polynomial;  
      integer;
```

```
      ( until choice );  
      if choice = 'POLVA' then stor_loc := 18  
      else  
      if choice = 'POLVB' then stor_loc := 19  
      else  
      if choice = 'POLYC' then stor_loc := 20  
      else  
      if choice = 'POLVD' then stor_loc := 21  
      else  
      if choice = 'POLVE' then stor_loc := 22  
      else  
      if choice = 'ONPOLY' then stor_loc := 0  
      else  
      if choice = 'ODPOLY' then stor_loc := 1  
      else  
      if choice = 'CNPOLY' then stor_loc := 2  
      else  
      if choice = 'CDPOLY' then stor_loc := 3  
      else  
      if choice = 'GNPOLY' then stor_loc := 4  
      else  
      if choice = 'GDPOLY' then stor_loc := 5  
      else  
      if choice = 'HNPOLY' then stor_loc := 6  
      else  
      if choice = 'HDPOLY' then stor_loc := 7;  
      assign( polys, 'tf&polys.dat' );  
      reset( polys );  
      seek( polys, stor_loc );  
      write( polys, pol );  
      close( polys );  
end;  
(* *****  
* procedure:      form  
* version:      1.0  
* date:      7 October 85  
* description: This file contains the procedures to form OLTF's  
*      and CLTF's  
* global variables used:      abort_command  
* global constants used:      crt_only  
*  
***** *)
```

```
FILE: FORM.PAS
```

```

* procedures called:      clear,      gotoxy,
*                        disp_msg,   out_string,
*                        get_int,   polylmt,
*                        pause,    clear_msg,
*                        poly_from_storage, highlight,
*                        poly_into_storage, nohighlight,
*                        spolylmt, polyadd, disptf
*
* called by:      select
* author:      Susan K. Mashiko, Capt, USAF
*              Gary C. Tarczynski, Capt, USAF
*
* *****

```

```

(* NOTE: the procedure disptf was forward referenced in the *)
(* file gettf. If you changed the relative position of *)
(* that file to this one insure the reference precedes *)
(* this file *)

```

```

procedure form;

```

```

label
  abort, abort2, repeat_again;

```

```

var
  selection      : integer;
  gnpol, hnpol  : polynomial;
  onpol         : polynomial;
  gdpol, hdpol  : polynomial;
  odpol        : polynomial;
  cnpol, cdpol  : polynomial;
  tempgo       : polynomial;
  choice       : cmdword;
  gain        : real;

```

```

begin
  (* get the selection from the user *)
  clear;
  repeat
    begin
      gotoxy( 5, 0 );
      disp_msg( 59 );
      gotoxy( 12, 30 );
      out_string( ' ', crt_only );
      gotoxy( 12, 30 );
      get_int( selection, abort_command );
      if abort_command then exit;
      if (( selection > 4 ) or ( selection < 1 )) then
        begin

```

```

FILE: FCRM.PAS

```



```

gotoxy( 14, 5 );
disp_msg( 9 );
pause;
gotoxy( 14, 5 );
clear_msg( 9 );
end;
until ( ( selection > 0 ) and ( selection < 5 ) );
(* selection 1 forms an OLTf from the GTF and the HTF *)
if selection = 1 then
begin
choice := 'GNPOLY';
poly_from_storage( choice, gnpol );
choice := 'HNPOLY';
poly_from_storage( choice, hnpol );
choice := 'GDPOLY';
poly_from_storage( choice, gapol );
choice := 'HDPOLY';
poly_from_storage( choice, hapol );
(* check to see if the resulting tf will have a degree *)
(* greater than 10
if ( gnpol.polydeg + hnpol.polydeg ) > 10 then
goto abort;
if ( gnpol.polydeg + hnpol.polydeg ) > 10 then
begin
abort;
clear;
gotoxy( 8, 10 );
highlight;
writeln( ' Degree of result greater than 10, option aborted ' );
nonhighlight;
writeln( ' Due to the storage space limitations your
resulting ');
writeln( ' polynomial is limited to 10 th order');
exit;
end
else
begin
polymlt( gnpol, hnpol, onpol );
polymlt( gapol, hapol, odpol );
(* gain change code commented out for phase 1 *)
( gotoxy( 14, 0 );
writeln( ' OPEN-LOOP GAIN = GAIN * ( OLN-GAIN / OLD-GAIN ) ');
writeln( ' GAIN = ');
writeln( ' Enter <CR> for default value of 1 ');

```

FILE: FORM.PAS

```

(* get the gain from the user *)
repeat again;
  gotoxy( 15, 13 );
  out_string( '          ', crt_only );
  gotoxy( 15, 13 );
  get_r_num( gain, 15, 13, abort_command );
  if abort_command then exit;

  onpol.coefficient := gain * ( onpol.coefficient / odpol.coefficient
);
  odpol.coefficient := 1.0;

  choice := 'ONPOLY';
  poly_into_storage( choice, onpol );
  choice := 'ODPOLY';
  poly_into_storage( choice, odpol );
  choice := 'OLTF';
  clear;
  disptf( choice );
end;

end;

(* this selection forms the CLTF from GTF and HTF *)
(* CLTF = ( GAIN * GTF ) / ( 1 + GAIN * GTF * HTF ) *)
if selection = 2 then
  begin
    choice := 'GNPOLY';
    poly_from_storage( choice, gnpol );
    choice := 'HNPOLY';
    poly_from_storage( choice, hnpol );
    choice := 'GDPOLY';
    poly_from_storage( choice, gdpol );
    choice := 'HDPOLY';
    poly_from_storage( choice, hdpol );

    (* check to see if the resulting tf will have *)
    (* a degree greater than 10 *)
    if ( gnpol.polydeg + hnpol.polydeg ) > 10 then
      goto abort;
    if ( gnpol.polydeg + hnpol.polydeg ) > 10 then
      begin
        abort2;
        clear;
        gotoxy( 8, 10 );
        highlight;
        writein( ' Degree of result greater than 10, option aborted ');
        highlight;
        writein( '          Due to the storage space limitations your
resulting ');

```

FILE: FORM.PAS

```

writein('      polynomial is limited to 10 th order');
exit;
end
else
begin
  polymit( gnpol, hnpol, cnpol );

  (* gain change code commented out for phase 1 *)
  gotoxy( 14, 0 );
  writein('      CLOSED-LOOP GAIN = GAIN * ( CLN-GAIN / CLD-GAIN ) ');
  writein('      GAIN =');
  writein('      Enter real number or default value of 1 ');

  (* get the gain from the user *)
  repeat_again:
  gotoxy( 15, 13 );
  out_string('      ', crt_only );
  gotoxy( 15, 13 );
  get_r_num( gain, 15, 13, abort_command );
  if abort_command then exit;

  cnpol.coefficient := gain * ( cnpol.coefficient / cdpol.coefficient
);
  cdpol.coefficient := 1.0;

  gain := 1;
  polymit( cnpol, tempol, gain );
  polymit( gdpol, hdpol, cnpol );
  polyadd( cnpol, tempol, cdpol );
  polymit( gnpol, hnpol, tempol );
  spolymlt( tempol, cnpol, gain );

  choice := 'ONPOLY';
  poly_into_storage( choice, cnpol );
  choice := 'CDPOLY';
  poly_into_storage( choice, cdpol );
  clear;
  disptf( choice );
end;

end;

(* selection 3 forms the CLTF from the OLTf *)
if selection = 3 then
begin
  choice := 'ONPOLY';
  poly_from_storage( choice, onpol );
  choice := 'ODPOLY';
  poly_from_storage( choice, adpol );

```

FILE: FORM.PAS

```

(* gain change code commented out for phase 1 *)
( gotoxy( 14, 0 );
  writeln('   CLOSED-LOOP GAIN = GAIN * ( CLN-GAIN / CLD-GAIN ) ');
  writeln('   GAIN =');
  writeln('   Enter real number or default value of 1 ');

  (* get the gain from the user *)
  repeat_again:
  gotoxy( 15, 13 );
  out_string(' ', crt_only );
  gotoxy( 15, 13 );
  get_r_num( gain, 15, 13, abort_command );
  if abort_command then exit;

  cnpol.coefficient := gain * ( cnpol.coefficient / cdpol.coefficient );
  cdpol.coefficient := 1.0;

  gain := 1;

  polymlt( onpol, cnpol, gain );
  polyadd( cnpol, odpol, cdpol );

  choice := 'CNPOLY';
  poly_into_storage( choice, cnpol );
  choice := 'CDPOLY';
  poly_into_storage( choice, cdpol );
  choice := 'CLTF';
  clear;
  disp( choice );
end;

(* selection 4 forms the CLTF from the GTF and the HTF in parallel *)
if selection = 4 then
begin
  choice := 'GNPOLY';
  poly_from_storage( choice, gnpol );
  choice := 'HNPOLY';
  poly_from_storage( choice, hnpol );
  choice := 'GDPOLY';
  poly_from_storage( choice, gdpol );
  choice := 'HDPOLY';
  poly_from_storage( choice, hdpol );
  polymlt( gnpol, hdpol, tempol );

  (* gain change code commented out for phase 1 *)
  ( gotoxy( 14, 0 );
    writeln('   CLOSED-LOOP GAIN = GAIN * ( CLN-GAIN / CLD-GAIN ) ');
    writeln('   GAIN =');

```

FILE: FORM.PAS

```

writeln( '      Enter real number or default value of 1.0 );
(* get the gain from the user *)
repeat again;
gotoxy( 15, 13 );
out_string( '      ', crt_only );
gotoxy( 15, 13 );
get_r_num( gain, 15, 13, abort_command );
if abort_command then exit;
cnpol.coefficient := gain * ( cnpol.coefficient / cdpol.coefficient );
cdpol.coefficient := 1.0;
gain := 1;
polymlt( gdpol, hnpol, cdpol );
polyadd( tempol, cdpol, cnpol );
polymlt( gdpol, hdpol, cdpol );
choice := 'CNPOLY';
poly_into_storage( choice, cnpol );
choice := 'CDPOLY';
poly_into_storage( choice, cdpol );
choice := 'CLTF';
clear;
disptf( choice );
end; (* end of selection 4 *)
end; (* end of procedure *)

```

FILE: FORM.PAS

```

*****
**
** file: GETCOM.PAS
** procedures contained: get_cmd
** version: 3.1
** date: 16 august 1983
** description: This file contains the procedure
** get_cmd which handles all processing
** associated with getting a valid command
** from the user. It is called by the
** program and operation remains here
** until a valid command is entered.
** vincent m. parisi ii, capt., usaf
**
**
**
**
**
*****

```

```

*****
**
** procedure: get_cmd
** version: 3.1
** date: 16 august 1983
** description: This procedure handles all processing
** associated with getting a valid command
** from the user. It is called by the
** program and operation is maintained
** here until a decoded and validated com-
** mand is entered.
**
** global variables used: help_level, cmdbuffer,
** call_routine, abort_command
**
** global variables changed: abort_command
**
** global constants used: yes
**
** passed variables: cmdbuffer, call_routine,
** num_of_commands
**
** returned variables:
** num_of_commands
**
** procedures called: gotoxy, readcom, get_line,
** val_n_dec, prompt_help,
** displa_commandword, prompt_cmd,
** instruction, proces_error_clear
**
** called by: ICECAPPC
** author: vincent m. parisi ii, capt., usaf
**
**
**
**
*****

```

```

procedure g_cmd( var cmdbuffer : buffer; var call_routine : cmdword ;
var num_of_commands : integer );

```

const

FILE: GETCOM.PAS

```

pr_cmd_row = 11;
pr_cmd_col = 5;
pr_hlp_row = 5;
cmd_row = 11;
cmd_col = 20;
instr_row = 2;
instr_col = 5;

var
i : integer;
level : integer;
rec_num : integer;
abort_command : boolean;
bufferpointer : integer;
error_code : char;

begin
abort_command := yes; (* initialize so first level of command *)
error_code := 'a'; words are displayed.
repeat
if abort_command = yes then
begin
bufferpointer := 1; (* set bufferpointer on initial entry and when
user wanted to abort previous command. *)
rec_num := 1; (* get first dictionary entry *)
level := 1; (* initialize for instruction *)
(* and valdec *)
end;
clear; (* clear the screen of everything *)
if help_level = 3 then (* issue instructions based on help
level *)
instruction( level, instr_row, instr_col );
if help_level > 1 then
prompt_help( rec_num, pr_hlp_row ); (* display the appropriate command *)
(* words based on pointer(s) in decode
*)
prompt_cmd( pr_cmd_row, pr_cmd_col ); (* put up the logo *)

(* display the contents of the command buffer. if it is empty, nothing *)
(* will be displayed, but if there are some commands in it form an unre-*)

```

FILE: GETCOM.PAS

```

(*) solved command, then display them.
gotoxy( cmd_row, cmd_col );      (* position for command *)
for i := 1 to ( bufferpointer - 1 ) do
  displa_commandword( cmdbuffer, i );
abort_command := no;      (* set abort command for read command *)
(* get command from user *)
num_of_commands := 0;
readcom( cmdbuffer, bufferpointer, abort_command );
if (( abort_command = no ) and ( bufferpointer > 1 )) then
  begin
    (* set parameters for entry into val- *)
    (* date and decode (val_n_dec) *)
    num_of_commands := ( bufferpointer - 1 );
    rec_num := 1;
    level := 1;      (* enter in with first commandword *)
    error_code := 'a';      (* initial code, not really an error *)
    (* now decode the entered command *)
    val_n_dec( level, rec_num, error_code, num_of_commands, cmdbuffer,
      call_routine );
  if ( error_code = 'b' ) then
    begin
      if help_level > 1 then
        begin
          gotoxy( cmd_row, cmd_col );
          proces_error( error_code, level, cmdbuffer, bufferpointer );
        end;
        bufferpointer := level;      (* get word that is bad *)
      end;
    end;
  until (error_code = 'n') or (error_code = 'c');
end;

```

FILE: GETCOM.PAS


```

*****
**
** file: GETDAT.PAS
** procedure contained: get_data, bid_stat_line,
** title_slide
**
** version: 2.0
** date: 22 July 85
** description: This file contains the procedures that
** read the DATA.DAT file, initialize the
** program variables, initialize and build
** the status line, and display the title
** slide.
** author: Vincent m. parisi ii, capt., usaf
** Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt, USAF
**
*****
*****
*****
*****
**
** procedure: title_slide
** version: 3.0
** date: 22 July 85
** description: This procedure displays the system title
** slide. By this, it demonstrates that at
** least the terminal control codes have been
** initialized properly, and probably the
** rest of the parameters.
** passed variables: term_dat
** procedures called: clear, gotoxy, highlight,
** nohighlight, rectangle
** called by: get_dat
** author: Gary C. Tarczynski, Capt, USAF
** Susan K. Mashiko, Capt, USAF
**
*****
*****
*****
*****
**
** procedure title_slide( var term_dat : term_array );
**
** begin
** clear;
** gotoxy(2,34);
** write( ' WELCOME TO ' );
** rectangle(4,19,41,7);
**
** (*----- begin writing "ICECAP" in big letters -----*)
** highlight;

```

FILE: GETDAT.PAS

```

gotoxy(5,21); write(' ');
gotoxy(5,26); write(' ');
gotoxy(5,33); write(' ');
gotoxy(5,39); write(' ');
gotoxy(5,46); write(' ');
gotoxy(5,53); write(' ');

gotoxy(6,22); write(' ');
gotoxy(6,26); write(' ');
gotoxy(6,30); write(' ');
gotoxy(6,33); write(' ');
gotoxy(6,39); write(' ');
gotoxy(6,43); write(' ');
gotoxy(6,46); write(' ');
gotoxy(6,50); write(' ');
gotoxy(6,53); write(' ');
gotoxy(6,57); write(' ');

gotoxy(7,22); write(' ');
gotoxy(7,26); write(' ');
gotoxy(7,33); write(' ');
gotoxy(7,39); write(' ');
gotoxy(7,46); write(' ');
gotoxy(7,50); write(' ');
gotoxy(7,53); write(' ');

gotoxy(8,22); write(' ');
gotoxy(8,26); write(' ');
gotoxy(8,30); write(' ');
gotoxy(8,33); write(' ');
gotoxy(8,39); write(' ');
gotoxy(8,43); write(' ');
gotoxy(8,46); write(' ');
gotoxy(8,53); write(' ');

gotoxy(9,21); write(' ');
gotoxy(9,26); write(' ');
gotoxy(9,33); write(' ');
gotoxy(9,39); write(' ');
gotoxy(9,46); write(' ');
gotoxy(9,50); write(' ');
gotoxy(9,53); write(' ');
nohighlight;
(*----- end writing "ICECAP" in big letters -----*)

gotoxy(12,11);
write(' INTERACTIVE CONTROL ENGINEERING COMPUTER ANALYSIS PACKAGE ');
gotoxy(14,26);
write(' ZENITH Z-100 - VERSION 1.0 ');

```

FILE: GETDAT.PAS

```

rectangle(16,4,42,6);
gotoxy(17,5);
write(' DEVELOPED AT: ');
gotoxy(18,5);
write(' The Air Force Institute of Technology ');
gotoxy(19,5);
write(' Electrical Engineering Dept ');
gotoxy(20,5);
write(' Wright-Patterson AFB, OH 45433 ');
gotoxy(22,21);
highlight;
write(' >>> Press <CR> Key to Continue... <<<< ');
nohighlight;
readln;
gotoxy(5,20);
write(' COPYRIGHT 1985 ');
gotoxy(6,20);
write(' ');
gotoxy(7,20);
write(' WRITTEN BY: Capt Susan K. Mashiko ');
gotoxy(8,20);
write(' Capt Gary C. Tarczynski ');
gotoxy(9,20);
write(' MENU DEVELOPED BY: Capt Paul A. Moore ');
gotoxy(17,5);
write('For more information on ICECAP, contact:');
gotoxy(18,5);
write(' Dr. Gary B. Lamont ');
gotoxy(19,5);
write(' Electrical Engineering Dept ');
gotoxy(20,5);
write(' Air Force Institute of Technology ');
end;

```

```

(*
* procedure: bld_stat_line
* version: 1.2
* date: 18 oct 83
* description: This module builds the status line from
* initialization data from disk storage.
* Data is in param group one.
* global variables used: status_line, help_level,
* temp, printer, trans
* global variables changed: status_line
* passed variables: help_level, temp, printer,
* trans
* called by: get_dat
* author: vincent m. parisi ii, capt., usaf
*)

```

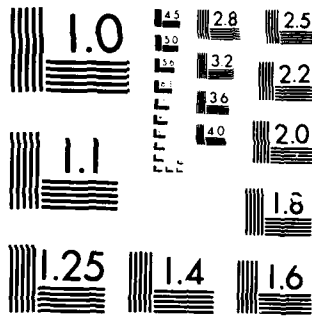
FILE: GETDAT.PAS

```

*
* *****
*
procedure bld_stat_line
( help_level : integer; temp : boolean; printer : boolean;
  var help : char; on_off : string[ 3 ] );
begin
  status_line := concat( status_line, ' Help level = ' );
  if help_level = 3 then help := '3';
  else
    if help_level = 2 then help := '2';
    else
      help := '1';
  status_line := concat( status_line, help );
  status_line := concat( status_line, ' Echo Print - ' );
  if printer then on_off := 'ON';
  else
    on_off := 'OFF';
  status_line := concat( status_line, on_off );
  status_line := concat( status_line, ' Transaction copy - ' );
  if trans then on_off := 'ON';
  else
    on_off := 'OFF';
  status_line := concat( status_line, on_off );
  status_line := concat( status_line, ' Temporary - ' );
  if temp then on_off := 'ON';
  else
    on_off := 'OFF';
  status_line := concat( status_line, on_off );
end;
( *****
*
* procedure: get_data
* version: 4.0
* date: 22 July 85
* description: This procedure reads the DATA.DAT file and
* initializes the program variables passed to
* it. Also calls title_slide and
* bld_stat_line.
*
* global variables used: blanks, call_routine,
* status_line, msg_dir,
* decode_dict, printer, trans,
* temp, crt, show_abbreviation,
*
*

```

FILE: GETDAT.PAS



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

```

* in_terminal, stat_on,
* macro_error, help_level,
* list_dev_name, trans_file_name,
* macro_file_name
* same as global variables used
* global variables changed:
* global constants used:
* term_length, screen_width,
* printer_length, num_msg_dir,
* num_ptr, num_words
* passed variables:
* term_dat, print_dat, msg_dir,
* decode_dict, printer, trans,
* temp, crt, show_abbreviation,
* in_terminal, stat_on,
* macro_error, help_level,
* list_dev_name, trans_file_name,
* macro_file_name
* returned variables:
* all passed variables are changed
* except term_dat and print_dat
* files created:
* PRINTER.OUT, TRANSACT.ION,
* TEMP.OUT, MACRO.INP
* files read:
* MICROSDW.SYS, HELP.SYS
* procedures called:
* clear, title_slide,
* bid_stat_line
*
* called by:
* ICECAPPC
* author:
* Vincent m. parisi, capt., usaf
* Susan K. Mashiko, Capt, USAF
* modified by:
* Gary C. Tarczynski, Capt, USAF
*
* mod description:
* Modified file assignment statements for
* the files PRINTER.OUT, TRANSACT.ION,
* MACRO.INP, and TEMP.OUT.
*
*
* mod date:
* 22 July 85
*
* *****
* procedure get_data( var term_dat : term_array;
* var print_dat : print_array;
* var msg_dir : msg_array;
* var decode_dict : dict_buffer;
* var printer : boolean;
* var trans : boolean;
* var temp : boolean;
* var crt : boolean;
* var show_abbreviation : boolean;
* var in_terminal : boolean;
* var stat_on : boolean;
* var macro_error : boolean;
* var help_level : byte ;
* var list_devname : paramstring;
* var trans_file_name : paramstring;
*

```

```

var macro_file_name : paramstring);

type   datarecord = record
      tdata : data;
      end;
      dataptr = ^datarecord;

var   data_file : file of data;
      data_recs : dataptr; { use a pointer to a record containing }
      { a record so the initialization data }
      { can be disposed of after it is }
      { transferred to global storage areas }
      i : integer;

begin
  writeln('Initializing the MICROSDW Menu Structure ',
        'and Hardware configuration');
  assign( data_file, 'MICROSDW.SYS');
  reset( data_file );
  new(data_recs);
  read( data_file, data_recs.tdata );
  close( data_file );
  move( data_recs.tdata.term, term_dat, ( term_length + term_length ));
  blanks := '';
  callRoutine := '';
  for i := 1 to screen_width do
    blanks := concat( blanks, ' ');
  status_line := '';
  stat_on := false;
  title_slide( term_dat );
  (* ----- Transfer the rest of the data to Global Storage areas *)
  for i := 1 to printer_length do
    print_dat[ i ] := data_recs.tdata.printr[ i ];
  for i := 1 to num_msg_dir do
    begin
      msg_dir[ i ].loc_rec := data_recs.tdata.msg_dir[ i ].loc_rec;
      msg_dir[ i ].length := data_recs.tdata.msg_dir[ i ].length;
    end;
  for i := 1 to num_ptrs do

```

FILE: GETDAT.PAS


```

begin
  decode_dict_ptrs[ 1, 1 ] := data_recs^.tdata.decode_dict_ptrs[ 1, 1 ];
  decode_dict_ptrs[ 1, 2 ] := data_recs^.tdata.decode_dict_ptrs[ 1, 2 ];
  decode_dict_ptrs[ 1, 3 ] := data_recs^.tdata.decode_dict_ptrs[ 1, 3 ];
end;

```

```

for i := 0 to num_words do

```

```

  begin
    decode_dict_words[ i ] := data_recs^.tdata.decode_dict_words[ i ];
    decode_dict_abbrev[ i ] := data_recs^.tdata.decode_dict_abbrev[ i ];
  end;

```

```

with data_recs^.tdata do

```

```

  begin
    printer := param[1].bools[1];
    trans := param[1].bools[2];
    temp := param[1].bools[3];
    crt := param[1].bools[4];
    show_abbreviation := param[1].bools[5];
    in_terminal := param[1].bools[6];
    station := param[1].bools[7];
    macro_error := param[1].bools[8];
    help_level := param[1].ints[1];
    list_dev_name := param[1].strings[1];
    trans_file_name := param[1].strings[2];
    macro_file_name := param[1].strings[3];
  end;

```

```

(* dispose of the temporary "data_recs" storage *)

```

```

dispose(data_recs);

```

```

(* build the status line to be shown after every pause and clear*)

```

```

bid_stat_line( help_level, temp, printer, trans );

```

```

(*****
 * now open other files used in this system so they are
 * ready for use.
 *****)

```

```

(*DELETE**DELETE**DELETE**DELETE**DELETE**DELETE**DELETE**DELETE*

```

```

This section of Moore's original code was commented
out by Mashiko and Tarczyński due to problems with
the file assignment statements.

```

```

assign( list_dev, list_dev_name );

```

```

rewrite( list_dev );

```

```

assign( trans_file, trans_file_name );

```

```

if trans then rewrite( trans_file );

```

```

FILE: GETDAT.PAS

```

```
rewrite( temp_file );
assign( macro_file, macro_file_name );
reset( macro_file );
assign( msg_txt, 'help.sys' );
reset( msg_txt );
*DELETE**DELETED**DELETED**DELETED**DELETED**DELETED**DELETED**
(*INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT*
  This section of code was added by Mashiko and
  Tarczynski to solve the file assignment problems.**
assign( list_dev, 'PRINTER.OUT' );
if printer then rewrite( list_dev );
assign( trans_file, 'TRANSACTION' );
if trans then rewrite( trans_file );
assign( temp_file, 'TEMP.OUT' );
if temp then rewrite( temp_file );
assign( macro_file, 'MACRO.INP' );
rewrite( macro_file );
assign( msg_txt, 'HELP.SYS' );
reset( msg_txt );
(*INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT*
end;
```

FILE: GETDAT.PAS

```

*****
** file: GETDAT.PAS *** IBM ONLY ***
** procedure contained: get_data, bid_stat_line,
** title_slide
**
** version: 2.0
** date: 22 July 85
** description: This file contains the procedures that
** read the DATA.DAT file, initialize the
** program variables, initialize and build
** the status line, and display the title
** slide.
** author: Vincent m. parist 11. capt., usaf
** Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt, USAF
**
**
**
*****
** procedure: title_slide *** IBM ONLY ***
** version: 3.0
** date: 22 July 85
** description: This procedure displays the system title
** slide. By this, it demonstrates that at
** least the terminal control codes have been
** initialized properly, and probably the
** rest of the parameters.
** passed variables: term_dat
** procedures called: clear, gotoxy, highlight,
** nohighlight, rectangle
** called by: get_dat
** author: Gary C. Tarczynski, Capt, USAF
** Susan K. Mashiko, Capt, USAF
**
**
*****
procedure title_slide( var term_dat : term_array );
begin
  clear;
  gotoxy(2,34);
  write(' WELCOME TO ');
  rectangle(4,19,41,7);
  highlight;
(*----- begin writing "ICECAP" in big letters -----*)

FILE: GETDAT.PAS *** IBM ONLY ***

```

```

gotoxy(5,21); write(' ');
gotoxy(5,26); write(' ');
gotoxy(5,33); write(' ');
gotoxy(5,39); write(' ');
gotoxy(5,46); write(' ');
gotoxy(5,53); write(' ');

gotoxy(6,22); write(' ');
gotoxy(6,26); write(' ');
gotoxy(6,30); write(' ');
gotoxy(6,33); write(' ');
gotoxy(6,39); write(' ');
gotoxy(6,43); write(' ');
gotoxy(6,46); write(' ');
gotoxy(6,50); write(' ');
gotoxy(6,53); write(' ');
gotoxy(6,57); write(' ');

gotoxy(7,22); write(' ');
gotoxy(7,26); write(' ');
gotoxy(7,33); write(' ');
gotoxy(7,39); write(' ');
gotoxy(7,46); write(' ');
gotoxy(7,50); write(' ');
gotoxy(7,53); write(' ');

gotoxy(8,22); write(' ');
gotoxy(8,26); write(' ');
gotoxy(8,30); write(' ');
gotoxy(8,33); write(' ');
gotoxy(8,39); write(' ');
gotoxy(8,43); write(' ');
gotoxy(8,46); write(' ');
gotoxy(8,53); write(' ');

gotoxy(9,21); write(' ');
gotoxy(9,26); write(' ');
gotoxy(9,33); write(' ');
gotoxy(9,39); write(' ');
gotoxy(9,46); write(' ');
gotoxy(9,50); write(' ');
gotoxy(9,53); write(' ');
nohighlight;
(*----- end writing "ICECAP" in big letters -----*)

gotoxy(12,11);
write(' INTERACTIVE CONTROL ENGINEERING COMPUTER ANALYSIS PACKAGE ');
gotoxy(14,26);
write(' IBM PC/XT/AT - VERSION 1.0 ');

```

FILE: GETDAT.PAS *** IBM ONLY ***

```

rectangle(16,4,42,6);
gotoxy(17,5);
write(' DEVELOPED AT: ');
gotoxy(18,5);
write(' The Air Force Institute of Technology ');
gotoxy(19,5);
write(' Electrical Engineering Dept ');
gotoxy(20,5);
write(' Wright-Patterson AFB, OH 45433 ');
gotoxy(22,21);
highlight;
write(' >>> Press <CR> Key to Continue... <<<< ');
nohighlight;
readln;
gotoxy(5,20);
write(' COPYRIGHT 1985 ');
gotoxy(6,20);
write(' ');
gotoxy(7,20);
write(' WRITTEN BY: Capt Susan K. Mashiko ');
gotoxy(8,20);
write(' Capt Gary C. Tarczynski ');
gotoxy(9,20);
write(' MENU DEVELOPED BY: Capt Paul A. Moore ');
gotoxy(17,5);
write(' For more information on ICECAP, contact: ');
gotoxy(18,5);
write(' Dr. Gary B. Lamont ');
gotoxy(19,5);
write(' Electrical Engineering Dept ');
gotoxy(20,5);
write(' Air Force Institute of Technology ');
end;

```

```

(*****
*
* procedure:   bid_stat_line
* version:    1.2
* date:       18 oct 83
* description: This module builds the status line from
*              initialization data from disk storage.
*              Data is in param group one.
*
* global variables used:
*   status_line, help_level,
*   temp, printer, trans
*
* global variables changed:
*   status_line
*
* passed variables:
*   help_level, temp, printer,
*   trans
*
* called by:   vincent m. parisi, capt., usaf
* author:
*
*****
)

```

FILE: GETDAT.PAS *** IBM ONLY ***

```

*****
procedure bid_stat_line
  ( help_level : integer; temp : boolean; printer : boolean;
  var help      : char;
  var on_off    : string[ 3 ] );
begin
  status_line := concat( status_line, ' Help level = ' );
  if help_level = 3 then help := '3'
  else
    if help_level = 2 then help := '2'
    else
      help := '1';
  status_line := concat( status_line, help );
  if printer then on_off := 'ON'
  else
    on_off := 'OFF';
  status_line := concat( status_line, on_off );
  if trans then on_off := 'ON'
  else
    on_off := 'OFF';
  status_line := concat( status_line, on_off );
  if temp then on_off := 'ON'
  else
    on_off := 'OFF';
  status_line := concat( status_line, on_off );
end;
(*****
*
* procedure:      get_data
* version:       4.0
* date:         22 July 85
* description:   This procedure reads the DATA.DAT file and
*               initializes the program variables passed to
*               it. Also calls title_slide and
*               bid_stat_line.
*
* global variables used:  blanks, call_routine,
*                       status_line, msg_dir,
*                       decode_dict, printer, trans,
*                       temp, crt, show_abbreviation,
*
*
*)

```

FILE: GETDAT.PAS *** IBM ONLY ***


```

var macro_file_name : paramstring);

type   datarecord   = record
      tdata : data;
      end;
      dataptr   = ^datarecord;

var   data_file   : file of data;
      data_recs   : dataptr;
      i           : integer;
begin
  writeln('Initializing the MICROSDW Menu Structure ',
        'and Hardware configuration');
  assign( data_file, 'MICROSDW.SYS' );
  reset( data_file );
  new(data_recs);
  read( data_file, data_recs^.tdata );
  close( data_file );
  move( data_recs^.tdata.term, term_dat, ( term_length + term_length ));
  blanks := '';
  callRoutine := '';
  for i := 1 to screen_width do
    blanks := concat( blanks, ' ');
  status_line := '';
  stat_on := false;
  title_slide( term_dat );
  (* ----- Transfer the rest of the data to Global Storage areas *)
  for i := 1 to printer_length do
    print_dat[ i ] := data_recs^.tdata.printr[ i ];
  for i := 1 to num_msg_dir do
    begin
      msg_dir[ i ].loc_rec := data_recs^.tdata.msg_dir[ i ].loc_rec;
      msg_dir[ i ].length := data_recs^.tdata.msg_dir[ i ].length;
    end;
  for i := 1 to num_ptrs do

```

FILE: GETDAT.PAS *** IBM ONLY ***


```

begin
  decode_dict_ptrs[ 1, 1 ] := data_recs^.tdata.decode_dict_ptrs[ 1, 1 ];
  decode_dict_ptrs[ 1, 2 ] := data_recs^.tdata.decode_dict_ptrs[ 1, 2 ];
  decode_dict_ptrs[ 1, 3 ] := data_recs^.tdata.decode_dict_ptrs[ 1, 3 ];
end;

```

```

for i := 0 to num_words do
begin
  decode_dict_words[ i ] := data_recs^.tdata.decode_dict_words[ i ];
  decode_dict_abbrev[ i ] := data_recs^.tdata.decode_dict_abbrev[ i ];
end;

```

```

with data_recs^.tdata do
begin
  printer := param[1].bools[1];
  trans := param[1].bools[2];
  temp := param[1].bools[3];
  crt := param[1].bools[4];
  show_abbreviation := param[1].bools[5];
  in_terminal := param[1].bools[6];
  stat_on := param[1].bools[7];
  macro_error := param[1].bools[8];
  help_level := param[1].ints[1];
  list_dev_name := param[1].strings[1];
  trans_file_name := param[1].strings[2];
  macro_file_name := param[1].strings[3];
end;

```

```

(* dispose of the temporary "data_recs" storage *)
dispose(data_recs);

```

```

(* build the status line to be shown after every pause and clear*)
bid_stat_line( help_level, temp, printer, trans );

```

```

(*****
* now open other files used in this system so they are *
* ready for use. *
*****

```

```

(*DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE*
This section of Moore's original code was commented
out by Mashiko and Tarczynski due to problems with
the file assignment statements.

```

```

assign( list_dev, list_dev_name );
rewrite( list_dev );

```

```

assign( trans_file, trans_file_name );
if trans then rewrite( trans_file );

```

```

FILE: GETDAT.PAS *** IBM ONLY ***

```

```
rewrite( temp_file );
assign( macro_file, macro_file_name );
reset( macro_file );
assign( msg_txt, 'help.sys' );
reset( msg_txt );
*DELETE**DELETE**DELETE**DELETE**DELETE**DELETE**DELETE**DELETE**DELETE**
(*INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT*
This section of code was added by Mashiko and
Tarczynski to solve the file assignment problems.*)
assign( list_dev, 'PRINTER.OUT' );
if printer then rewrite( list_dev );
assign( trans_file, 'TRANSACTION' );
if trans then rewrite( trans_file );
assign( temp_file, 'TEMP.OUT' );
if temp then rewrite( temp_file );
assign( macro_file, 'MACRO.INP' );
rewrite( macro_file );
assign( msg_txt, 'HELP.SYS' );
reset( msg_txt );
(*INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT*)
end;
```

FILE: GETDAT.PAS *** IBM ONLY ***

```

*****
**
**      GETINT.PAS
**
** procedures contained:  del_lst_ch
**                       ck_chr
**                       out_int
**                       get_int
**                       getch
**
**
** version: 1.2
** date: 18 Oct 83
** description: This file contains procedures that
**             handle I/O for integers. Since normal
**             program operation does not handle integer
**             error exceptions very well, these
**             routines do the job.
**             vincent m. parisi ii, capt., usaf
**
*****
)
*****
**
**      function:  getch
**      version:  1.0
**      date:     18 oct 83
**      description: This procedure gets one character from a
**                  string and returns it to the read function
**                  for conversion. Routine taken from the
**                  Pascal MT+ instruction manual, section on
**                  redirected I/O.
**
** global variables used:  string
** global variables changed:  string
** returned variables:  getch
** called by:  get_int, get_real
** author:  vincent m. parisi ii, capt., usaf
**
*****
)
*****
function getch : char;
begin
  if length( string ) > 0 then
    begin
      getch := string[ 1 ];
      delete( string, 1, 1 );
    end
  else
    getch := ' ';
end;

```

FILE: GETINT.PAS

```

*****
* procedure: del_lst_ch
* version: 1.0
* date: 18 oct 83
* description: This procedure deletes the last character
* from the CRT.
* global constants used: backspace
* called by: ck_chr
* author: vincent m. parisi Tt, capt., usaf
*****

```

```

procedure del_lst_ch;

```

```

begin
  write( chr( backspace ) );
  write( ' ' );
  write( chr( backspace ) );
end;

```

```

*****
* procedure: ck_chr
* version: 1.2
* date: 18 oct 83
* description: This procedure checks each character finput
* to see if it is a delete or a backspace,
* If it is, the screen is updated appropriately,
* and the destination string is changed.
* global variables used: string
* global constants changed: string
* global constants used: del, backspace
* passed variables: ch, string
* returned variables: string
* procedures called: del_lst_ch
* called by: get_int_ch
* author: vincent m. parisi Tt, capt., usaf
*****

```

```

procedure ck_chr( ch : char; var string : msg_line );

```

```

begin
  if ((( ord( ch ) = del ) or ( ord( ch ) = backspace )) and ( length( string
)
> 0 )) then
    delete( string, length( string ), 1 );

```

FILE: GETINT.PAS

```
if ord( ch ) = del then
del_lst_ch;
end;
```

```
(*****
*
* procedure: out_int
* version: 1.2_
* date: 18 oct 83
* description: This procedure directs the output of
* integers.
* global variables used: crt, trans, printer, temp,
* temp_file, trans_file, list_dev
* global variables changed: temp_file, trans_file, list_dev
* passed variables: number_field, dest_file, list_dev
* files written: TEMP.OUT, TRANSACT.ION, PRINTER.OUT
* called by: make_pretty
* author: vincent m. parisi ff, capt.. usaf
*
*****)
```

```
procedure out_int( number : integer; field : integer; dest : char );
```

```
begin
```

```
case dest of
'C', 'c' : write( number:field );
'P', 'p' : writeln( number:field );
'B', 'b' : begin
write( number:field );
writeln( number:field );
end;
'A', 'a' : begin
if crt then write( number:field );
if trans then
writeln( trans_file, number:field );
if printer then
writeln( list_dev, number:field );
if temp then
writeln( temp_file, number:field );
end;
end;
```

```
end;
```

```
(*****
*
* procedure: get_int
* version: 1.3_
* date: 18 oct 83
*
*****)
```

FILE: GETINT.PAS

```

* description: This procedure handles the input of integers.*
* Normal program integer input does not have *
* edit capability to exclude inputs such as *
* letters for integers. It just displays an *
* error message and asks for the number again. *
* This procedure does not accept anything *
* but valid integer constructs. *
* global variables used: string, abort_command *
* global variables changed: string, abort_command *
* global constants used: as_assigned *
* passed variables: number, abort_command *
* returned variables: number, abort_command *
* procedure called: get_string *
* called by: get_tf *
* author: vincent m. parisi ii, capt., usaf *
* *****
(* forward reference to get_string *)
procedure get_string(var string : msg_line; var abort_command : boolean;
in_dev : char; chr1, chr2 : char); forward;

procedure get_int( var number : integer; var abort_command : boolean );

var
ch : char;
result : integer;

begin
number := 0;
get_string( string, abort_command, as_assigned, '0', '9' );
(* Val converts a string to a number, result = 0 if no errors, else *)
(* result is the position of the invalid character in the string *)
if not abort_command then
Val(string,number,result);
end;

```

```

*****
**
** file: GETLINE.PAS
**
** procedure contained: get_line
** version: 3.1
** date: 27 Sept 1984
** description: This module contains the procedure that
** builds a decoded entry from the record
** pointed to on entry.
** author: vincent m. parisi ii, capt., usaf
**
**
**
*****

```

```

*****
**
** procedure: get_line
** version: 3.0
** date: 17 July 1983
** description: This procedure builds a decoded entry
** from the record pointed to on entry.
** The pointers come from the ptrs part
** of dict buffer and the word comes from
** the words part of dict buffer which is
** pointed to by the first pointer of ptrs.
**
** global variables used: blanks
** global constants used: wordsize, screen_width,
** buffersize, word_length
**
** passed variables: decode, rec_num
** returned variables: decode
** called by: get_cmd
**
** author: vincent m. parisi ii, capt., usaf
**
**
*****

```

```

procedure get_line( var decode : dictionary; rec_num : integer );
var dlen : integer;
begin
  decode.dictword := decode_dict.words[ decode_dict.ptrs[ rec_num, 1 ] ];
  decode.abbrev := decode_dict.abbrev[decode_dict.ptrs[ rec_num, 1 ] ];
  decode.matchp := decode_dict.ptrs[ rec_num, 2 ];
  decode.nomatchp := decode_dict.ptrs[ rec_num, 3 ];
  (* blank pad "decode.dictword" *)
  dlen := length(decode.dictword);

```

FILE: GETLINE.PAS

```
Insert(blanks,decode.dictword,dlen+1);  
end;
```

FILE: GETLINE.PAS


```

*****
**
** GETMAT.PAS
**
** procedures contained:  make_pretty_large_matrix_one,
**                       make_pretty_large_matrix_two,
**                       make_pretty_small_matrix,
**                       left_bracket, right_bracket, get_mat
**                       get_matrix_entries
**
** version:              1.0
** date:                 12 Sep 85
** description:          The procedures contained in this file will get a
**                       matrix and store it in its proper location.
**
** author:               Susan K. Mashiko, Capt, USAF
**                       Gary C. Tarczynski, Capt, USAF
**
*****

```

```

*****
**
** procedure:            left_bracket
**
** version:              1.0
** date:                 11 September 1985
** description:          This procedure draws a left bracket
**                       around a matrix displayed on the terminal.
**
** global variables used:  term      num_rows
**
** passed variables:      graphics, gotoxy, nographics
**
** procedures called:     make_pretty_small_matrix,
**                       make_pretty_large_matrix_one,
**                       make_pretty_large_matrix_two
**
** called by:             Gary C. Tarczynski, Capt, USAF
**
** author:               Susan K. Mashiko, Capt, USAF
**
*****

```

```

procedure left_bracket( var num_rows : integer );
var
  column_length : integer;
  i              : integer;
begin
  column_length := 5 + num_rows;
  graphics;
  gotoxy(5,7);

```

(* draw upper left corner *)

FILE: GETMAT.PAS

```

write( chr( term[64] ));
for i := 6 to column_length do
begin
gotoxy(i,7);
write( chr( term[54] ));
end;

gotoxy((column_length + 1), 7); (* draw lower left corner *)
write( chr( term[63] ));

nographics;

end;
(*****
*
* procedure: right_bracket
* version: 1.0
* date: 11 September 1985
* description: This procedure draws a right bracket
* around a matrix displayed on the terminal.
*
* global variables used: term num_rows, num_cols
*
* passed variables: graphics, gotoxy, nographics
*
* procedures called: make_pretty_small_matrix,
* make_pretty_large_matrix_two
*
* called by: Gary C. Tarczynski, Capt, USAF
* Susan K. Mashiko, Capt, USAF
*
*
*
*****
)

procedure right_bracket( var num_rows, num_cols : integer );
var
column_location : integer;
column_length : integer;
i : integer;
begin
column_location := 8 + ( num_cols * 13 ) + 2;
column_length := 5 + num_rows;
if num_cols >= 6 then
column_location := column_location - 65;
nographics;
gotoxy(5,column_location);
write( chr( term[61] ));
(* draw upper right corner *)

```

FILE: GETMAT.PAS

```

for i := 6 to column_length do
  begin
    gotoxy(i, column_location);
    write( chr( term[54] ) );
  end;
  (* draw column *)

```

```

gotoxy(( column_length + 1 ), column_location);
write( chr( term[62] ) );

```

```

nographics;

```

```

end;

```

```

(*****
*
* procedure: make_pretty_large_matrix_one
* version: 1.0
* date: 11 September 85
* description: This procedure will draw the left bracket and place
* row and col numbers on the first display screen of
* a matrix with more than 5 col.
* global constants used: crt_only, as assigned
* passed variables: num_row, num_col
* procedures called: gotoxy, out_string,
* out_int, left_bracket
* called by: get_mat
* author: Susan_K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
*
*****
)

```

```

procedure make_pretty_large_matrix_one( var num_row : integer;
var num_col : integer);

```

```

var
  i : integer;
  row : integer;
  col : integer;

```

```

begin
  row := 5;

```

```

  (* put up the row numbers and draw the left bracket *)
  for i := 1 to num_row do
    begin
      gotoxy(( row + i ), 2 );
      out_string( 'ROW', as assigned );
      gotoxy(( row + i ), 5 );
      out_int( i, 2, crt_only );
    end;

```

```

FILE: GETMAT.PAS

```

```

end;
left_bracket( num_row );
(* put up the column numbers *)
col := 14;
for i := 0 to 4 do
begin
gotoxy( 4, ( col + (i * 13)));
out_string( 'COL', as_assigned );
gotoxy( 4, ( col + (i * 13) + 3 ));
out_int( (i + 1), 2, crt_only );
end;
end;
(*****
*
* procedure: make_pretty_large_matrix_two
* version: 1.0
* date: 11 September 85
* description: This procedure will draw the right bracket of a
* matrix with more than 5 columns. It will also write
* the col and row identifiers on the screen.
* Global constants used: crt_only, as_assigned
* passed variables: num_row, num_col
* procedures called: gotoxy, out_string,
* out_int, right_bracket
* called by: get_matrix_entries
* author: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
*
*****
)

```

```

procedure make_pretty_large_matrix_two(var num_row : integer;
var num_col : integer);

```

```

var
i : integer;
row : integer;
col : integer;
begin
row := 5;
(* put up the row numbers *)
for i := 1 to num_row do
begin
gotoxy(( row + i ), 2 );
out_string( 'ROW', as_assigned );
gotoxy(( row + i ), 5 );
end;
end;

```

```

        out_int( 1, 2, crt_only );
    end;

    (* put up the column numbers *)
    col := 14;
    for i := 0 to ( num_col - 6 ) do
    begin
        gotoxy( 4, ( col + ( i * 13) ));
        out_string( 'COL', as_assigned );
        gotoxy( 4, ( col + ( i * 13) + 3 ));
        out_int( ( i + 6 ), 2, crt_only );
    end;

    (* draw the right bracket *)
    right_bracket( num_row, num_col );
end;

(*****
*
* procedure:  get_matrix_entries
* version:   1.0
* date:      11 September 85
* description: This procedure will get a matrix entry
* global variables used:  abort_command
* passed variables:      matrix, abort_command
* returned variables:
* procedures called:     get_r_num,  pause,
*                          clear,
*                          make_pretty_large_matrix_two
*
* called by:  get_mat
* author:    Susan K. Mashiko, Capt, USAF
*            Gary C. Tarczynski, Capt, USAF
*
*****
)

procedure get_matrix_entries( var matrix : matrix;
                             var abort_command : boolean );

var
    i      : integer;
    j      : integer;
    number : real;
    row    : integer;
    col    : integer;
    num_row : integer;
    num_col : integer;
    col_element : integer;

```

FILE: GETMAT.PAS

```

begin
  (* get the matrix entries *)
  row := 5;
  col := 10;
  num_row := matrix.num_rows;
  num_col := matrix.num_cols;

  (* i is the counter for rows and j is the counter for cols *)
  if num_col <= 5 then
    begin
      for j := 0 to ( num_col - 1 ) do
        begin
          col_element := j + 1;
          for i := 1 to num_row do
            begin
              get_r_num( number, (row + i), (col + (j * 13)), abort_command );
              if abort_command then exit;
              matrix.element[ i , col_element ] := number;
            end;
          end;
        end
      else
        begin
          for j := 0 to 4 do
            begin
              col_element := j + 1;
              for i := 1 to num_row do
                begin
                  get_r_num( number, (row + i), (col + (j * 13)), abort_command );
                  if abort_command then exit;
                  matrix.element[ i , col_element ] := number;
                end;
              end;
            end;
          end;
        end;
      pause;
      clear;
      make_pretty_large_matrix_two( num_row, num_col );

      for j := 0 to num_col - 6 do
        begin
          col_element := j + 6;
          for i := 1 to num_row do
            begin
              get_r_num( number, (row + i), (col + (j * 13)), abort_command );
              if abort_command then exit;
              matrix.element[ i , col_element ] := number;
            end;
          end;
        end;
      end;
    end;
  end;

```

FILE: GETMAT.PAS

```

end;
(*****
*
* procedure:  make_pretty_small_matrix
* version:    1.0
* date:      11 September 85
* description: This procedure will draw the brackets and label the
*             rows and columns for a matrix with less than 5 col.
*
* Global variables used:  none
* Global constants used:  crt_only, as_assigned
* Passed variables:      num_row, num_col
* Procedures called:     gotoxy, out_string,
*                       out_int, left_bracket,
*                       right_bracket
*
* called by:  get_mat
* author:    Susan K. Mashiko, Capt. USAF
*           Gary C. Tarczynski, Capt. USAF
*
*****)

procedure make_pretty_small_matrix(var num_row : integer;
var num_col : integer);

var
  i      : integer;
  row    : integer;
  col    : integer;

begin
  row := 5;

  (* put up the row numbers and draw the left bracket *)
  for i := 1 to num_row do
    begin
      gotoxy(( row + i ), 2 );
      out_string( 'ROW', as_assigned );
      gotoxy(( row + i ), 5);
      out_int( 1, 2, crt_only );
    end;
    left_bracket( num_row );

  (* put up the column numbers *)
  col := 14;
  for i := 0 to ( num_col - 1 ) do
    begin
      gotoxy( 4, ( col + ( i * 13) ) );
      out_string( 'COL', as_assigned );
    end;
  end;
end;

```

FILE: GETMAT.PAS

```
        gotoxy( 4, ( col + ( i * 13) + 3 ));
        out_int( ( i + 1), 2, crt_only );
    end;
```

```
(* draw the right bracket *)
right_bracket( num_row, num_col );
end;
```

```
(*****
*
* procedure:   get_mat
* version:    1.0
* date:       11 September 85
* description: This procedure will get a matrix and store it in
*              its proper location.
* global variables used:  abort_command
* global constants used:  crt_only,   max_rows,
*                          as_assigned, max_cols
* passed variables:      def_obj
* files written:         MATRIX.DAT
* procedures called:     clear,  out_string,  pause,  get_matrix_entries,
*                          gotoxy,  get_int,  get_matrix_entries,
*                          disp_msg, clear_msg,
*                          make_pretty_large_matrix_one,
*                          make_pretty_small_matrix
*
* called by:   Susan K. Mashiko, Capt, USAF
* author:      Gary C. Tarczynski, Capt, USAF
*
* *****)
```

```
procedure get_mat( var def_obj : cmdword );
```

```
var
    i          : integer;
    number     : real;
    num_row    : short_int;
    num_col    : short_int;
    matrices   : file_of_matrix;
    mats       : matrix;
    stor_loc   : integer;

begin
    abort_command := false;
    clear;
    gotoxy( 10, 5 );
    disp_msg( 41 );

    (* get the number of rows of the matrix *)
```

FILE: GETMAT.PAS


```

repeat
  begin
    gotoxy( 10, 50 );
    out_string( ' ', crt_only );
    gotoxy( 10, 50 );
    get_int( num_row, abort_command );
    if abort_command then exit;
    if (( num_row > max_rows ) or ( num_row < 0 )) then
      begin
        gotoxy( 12, 5 );
        disp_msg( 40 );
      end;
    end;
    until (( num_row > 0 ) and ( num_row <= max_rows ));

  clear_msg( 40 );

  (* get the number of columns of the matrix *)
  repeat
    begin
      gotoxy( 12, 5 );
      disp_msg( 42 );
      gotoxy( 12, 50 );
      out_string( ' ', crt_only );
      gotoxy( 12, 50 );
      get_int( num_col, abort_command );
      if abort_command then exit;
      if (( num_col > max_cols ) or ( num_col < 0 )) then
        begin
          gotoxy( 14, 5 );
          disp_msg( 40 );
        end;
      end;
      until (( num_col > 0 ) and ( num_col <= max_cols ));

    clear;

    (* display the title on the screen *)
    gotoxy( 1, 33 );
    disp_msg( 43 );
    gotoxy( 2, 37 );
    out_string( def_obj, as_assigned );
    mats.num_rows := num_row;
    mats.num_cols := num_col;

    (* if the matrix is small then get the entries *)
    (* small is less than 5 columns *)
    if num_col <= 5 then
      begin

```

FILE: GETMAT.PAS

```

make_pretty_small_matrix( num_row, num_col );
get_matrix_entries( mats, abort_command );
if abort_command then exit;
end
else
(* if the matrix is large then get the entries *)
if num_col > 5 then
begin
make_pretty_large_matrix_one( num_row, num_col );
get_matrix_entries( mats, abort_command );
if abort_command then exit;
end;
(* find the storage location for the matrix *)
if def_obj = 'MATA' then stor_loc := 0
else
if def_obj = 'MATB' then stor_loc := 1
else
if def_obj = 'MATC' then stor_loc := 2
else
if def_obj = 'MATD' then stor_loc := 3
else
if def_obj = 'MATE' then stor_loc := 4;
assign( matrices, 'matrix.dat' );
reset( matrices );
seek( matrices, stor_loc );
write( matrices, mats );
close( matrices );
pause;
end;

```

FILE: GETMAT.PAS

```
(*****  
**  
** file: GETSTRIN.PAS  
** procedure contained: get_string  
** version: 2.0  
** date: 28 august 1983  
** description: This module contains the procedure that  
** gets ASCII input from terminal keyboard  
** or macro command file as specified in  
** the input parameter. Collects charac-  
** ters until a <CR> is entered.  
** author: vincent m. parisi ii, capt., usaf  
**  
*****)  
(*****  
**  
** procedure: get_string  
** version: 2.0  
** date: 28 august 1983  
** description: Gets ascii input from terminal keyboard  
** or macro command file as specified in  
** the input parameter. Collects charac-  
** ters until a <CR> is entered.  
** global variables used: in_terminal, macro_file, string,  
** abort_command  
** global variables changed: string, abort_command  
** global constants used: screen_width, abort_str  
** passed variables: string, abort_command, in_dev,  
** chr1, chr2  
** returned variables: string, abort_command  
** files read: MACRO.INP  
** called by: readcom  
** author: vincent m. parisi ii, capt., usaf  
**  
*****)  
procedure get_string;  
(* arguments are specified in an earlier "forward" reference  
( var string : msg_line; var abort_command : boolean;  
 in_dev : char; chr1, chr2 : char );  
*)  
begin  
string := ''; (* clear the string *)  
abort_command := false;
```

FILE: GETSTRIN.PAS

```
case in_dav of
'A','a' : if in_terminal then
begin
BufLen := screen_width;
read( string);
end
else
read( macro_file, string );
'M','m' : read( macro_file, string );
'T','t' : begin
BufLen := screen_width;
read( string );
end
else
begin
BufLen := screen_width;
read( string );
end; (* end case *)
if (length(string) = 1) and (string[1] = abort_str) then
abort_command := true;
end;
```

FILE: GETSTRIN.PAS

```

*****
**
** file: GETTF.PAS
**
** procedure contained: get_r_num, make_pretty
** disp_poly, form_poly,
** roots,
** disp_fact, get_unfact,
** poly, get_tf
**
** version: 8.0
** date: 25 September 1985
** description: This file contains the procedures that input
** transfer functions in either factored or poly-
** nomial form.
** author: vjncnt m. parisi ii, capt., usaf
** Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt, USAF
**
**
**
*****

```

```

*****
**
** procedure: get_r_num
** version: 1.2
** date: 20 August 1985
** description: This procedure is entered with the location
** of where the real number is positioned, however
** the actual input is at the bottom of the screen
** which allows for various real number input
** formats on input (ie. exponential, decimal etc)
** once the number is entered, it is converted
** to exponential notation and displayed at the
** proper place.
**
** global variables used: blanks, abort_command
** global variables changed: blanks
** global constants used: crt_only, as_assigned
** passed variables: number, row, col, abort_command
**
** procedures called: gotoxy, highlight,
** nohighlight, out_string,
** get_real, out_real
**
** called by:
** author: vjncnt m. parisi ii, capt., usaf
** modified by: Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt, USAF
**
** mod description: changed the highlighted area for number is...
** mod date: 20 August 85
**
**
*****

```

FILE: GETTF.PAS

```

procedure get_c_num( var number : real; row : integer; col : integer;
var abort_command : boolean );

begin
  gotoxy( row, col );
  highlight;
  out_string( copy( blanks, 1, 12), crt_only );
  nohighlight;
  gotoxy( 20, 0 );
  out_string( blanks, crt_only );
  gotoxy( 20, 10 );
  highlight;
  out_string( 'You, number...', as_assigned );
  nohighlight;
  get_real( number, abort_command );
  if abort_command then exit;
  gotoxy( row, col );
  out_real( number, 12, as_assigned );
end;

```

```

(*****
*
* procedure:      make_pretty
* version:       2.3
* date:          26 Aug 85
* description:   This procedure pretties up the screen for
*               transfer function input.
*
* global variables used:
*   term, degree
*
* global constants used:
*   as_assigned
*
* passed variables:
*   row, degree
*
* procedures called:
*   gotoxy, disp_msg,
*   graphics, out_string,
*   nographics, out_int,
*
* called by:     get_fact, get_unfact
* author:        Vincent M. Parisi II, Capt, USAF
* modified by:   Susan K. Mashiko, Capt, USAF
*               Gary C. Tarczynski, Capt, USAF
*
* mod description: Changed the code to prevent over writing
*
* mod date:      26 Aug 85
*
*****)

```

```

procedure make_pretty( var row : integer; degree : integer );
var i : integer;
begin

```

FILE: GETTF.PAS

```

(* put up the titles and coefficient words *)
gotoxy( ( row + 1 ), 0 );
disp_msg( 14 );

graphics;

(* put up vertical line to divide screen at column 38 *)
i := row + 1;
repeat
begin
gotoxy( i, 38 );
out_string( chr( term[ 54 ] ), crt_only );
i := i + 1;
end;
until i = 20;

(* put up a horizontal line at row = entry row *)
for i := 0 to screen_width do
begin
gotoxy( row, i );
out_string( chr( term[ 55 ] ), crt_only );
end;

(* Now put up the "T" at the intersection *)
gotoxy( row, 38 );
out_string( chr( term[ 60 ] ), crt_only );

nographics;

(* put up the numbers to represent the roots to get and the degree
of the polynomial coefficient *)
for i := 1 to degree do
begin
gotoxy( ( row + 3 + i ), 40 );
out_int( i, 2, crt_only );
end;

(* put up "j's" in the root section *)
for i := 1 to degree do
begin
gotoxy( ( row + i + 3 ), 57 );
out_string( 'j', as_assigned );
end;

(* put up the powers of the coefficients in the coefficient section *)
for i := 0 to degree do
begin
gotoxy( ( row + 4 + i ), 24 );

```

FILE: GETTF.PAS

```

out_string( 'S**', crt_only);
out_int( ( degree - i ), 2, crt_only );
end;
end;

```

```

*****
*
* procedure:      get_fact
* version:       1.2
* date:          4 november 83
* description:   This procedure gets the factored form of the
*               polynomial.
*
* global variables used:  abort_command
*
* global constants used:  as_assigned
*
* passed variables:      poly, row, abort_command
*
* returned variables:    poly
*
* procedures called:     make_pretty, get_r_number,
*                       gotoxy, out_real,
*                       disp_msg, pause,
*                       clear_msg
*
* called by:            poly
*
* author:               vincent m. parisi {}, capt., usaf
*
* mod description:      Changed the order of conjugate storage
*
* modified by:          Susan K. Mashiko, Capt., USAF
*
* mod date:            8 Sep 85
*
*****

```

```

procedure get_fact( var poly : polynomial; var row : integer;
var abort_command : boolean );

```

```

label repeat1;

```

```

var i : integer;
number : real;

```

```

begin
make_pretty( row, poly.polydeg );
i := 1;

```

```

(* get the term's coefficient, display it and save it *)
(* if the coefficient is zero display message and get another *)
repeat1:
get_r_num( number, ( row + 2 ), 57, abort_command );
if abort_command then exit;
if number = 0 then
begin

```

```

FILE: GETTF.PAS

```



```

gotoxy( 20, 0 );
disp_msg( 47 );
pause;
gotoxy( 20, 0 );
clear_msg( 47 );
goto repeat1;
end;
gotoxy( ( row + 2 ), 19 );
out_real( number, 12, as_assigned );
poly_coefficient := number;

while i <= poly.polydeg do
begin
  (* get the real portion of the pole or zero for this root *)
  get_r_num( number, ( row + 3 + i ), 43, abort_command );
  if abort_command then exit;
  poly.polyfact[ i ].realpart := number;

  (* get the imaginary portion of the pole or zero for this root *)
  get_r_num( number, ( row + 3 + i ), 59, abort_command );
  if abort_command then exit;
  poly.polyfact[ i ].imagpart := number;

  (* if the root is complex, generate its conjugate and show it *)
  (* always put the negative imaginary part first in the list with *)
  (* its positive conjugate second. if the imag part is <> 0 *)
  (* and this is the last position in the array, issue an error *)
  (* message and get the real part again. *)
  if (( number < -0.000001 ) or ( number > 0.000001 )) then
  begin
    if i = poly.polydeg then
    begin
      gotoxy( 20, 5 );
      disp_msg( 10 );
      pause;
      gotoxy( 20, 0 );
      clear_msg( 10 );
      i := i - 1;
    end
    else
    begin
      poly.polyfact[ i + 1 ].realpart := poly.polyfact[ i ].realpart;
      gotoxy( ( row + i + 4 ), 43 );
      out_real( poly.polyfact[ i ].realpart, 12, as_assigned );
      gotoxy( ( row + i + 3 ), 59 );
    end
  end
end;

```

FILE: GETTF.PAS

```

if number < 0.0 then
begin
poly.polyfact[ i ].imagpart := number;
out_real( number, 12, as_assigned );
poly.polyfact[ i + 1 ].imagpart := ( number * (-1.0) );
end
else
begin
poly.polyfact[ i ].imagpart := ( number * (-1.0) );
out_real( poly.polyfact[ i ].imagpart, 12, as_assigned );
poly.polyfact[ i + 1 ].imagpart := number;
end;
i := i + 1;
gotoxy( ( row + i + 3 ), 59 );
out_real( poly.polyfact[ i ].imagpart, 12, as_assigned );
end;
end;
i := i + 1;
end; (* while *)
end; (* end of procedure *)
(*****
* procedure: form_poly
* version: 1.0
* date: 26 Aug 85
* description: This procedure forms the polynomial from the
* factors.
* global constants used: maxdeg1
* passed variables: poly
* returned variables: poly
* called by: poly
* author: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
* *****)
procedure form_poly( var poly : polynomial );
var
real_coeff : array[1..maxdeg1] of real;
imag_coeff : array[1..maxdeg1] of real;
i, j, k : integer;
number_of_roots : integer;
number_of_coeff : integer;
begin
number_of_roots := poly.polydeg;
number_of_coeff := number_of_roots + 1;

```

FILE: GETTF.PAS

```

for i := 2 to number_of_coeff do
begin
  real_coeff[ i ] := 0.0;
  imag_coeff[ i ] := 0.0;
end;

real_coeff[ 1 ] := 1.0;
imag_coeff[ 1 ] := 0.0;
for i := 1 to number_of_roots do
begin
  k := i + 1;
  for j := 1 to i do
begin
  real_coeff[ k ] := -real_coeff[k-1] * poly.polyfact[i].realpart
+imag_coeff[k-1] * poly.polyfact[i].imagpart
+real_coeff[k];
  imag_coeff[ k ] := -imag_coeff[k-1] * poly.polyfact[i].realpart
-realm_coeff[k-1] * poly.polyfact[i].imagpart
+imag_coeff[k];
  k := k - 1;
end;
end;
for i := 1 to number_of_coeff do
begin
  poly.polypoly[i] := real_coeff[i];
end;
end;

```

```

(*****
* procedure: get_unfact
* version: 2.0
* date: 25 September 85
* description: This procedure gets the polynomial form of the
* polynomial of interest.
* global variables used: abort_command
* global constants used: as_assigned
* passed variables: poly, row, abort_command
* returned variables: poly
* procedures called: make_pretty, get_r_num, gotoxy,
* out_real, disp_msg, pause, clear_msg
* called by: Susan K. Mashtko, Capt, USAF
* author: Gary C. Tarczynski, Capt, USAF
* modified by: author
* mod description: Corrected the gain handling procedures
* mod date: 25 Sep 85
*****
)

```

FILE: GETTF.PAS

```

procedure get_unfact( var poly : polynomial; row : integer;
var abort_command : boolean );

label  repeat1,
      repeat2;
var i : integer;
var number : real;

begin
  make_pretty( row, poly.polydeg );
  i := 1;
  (* get the term's coefficient, display it and save it *)
  repeat1:
    get_r_num( number, ( row + 2 ), 19, abort_command );
    if abort_command then exit;
    if number = 0 then
      begin
        gotoxy( 20, 0 );
        disp_msg( 47 );
        pause;
        gotoxy( 20, 0 );
        clear_msg( 47 );
        goto repeat1;
      end;
    gotoxy( ( row + 2 ), 57 );
    out_real( number, 12, as_assigned );
    poly.coefficient := number;

  repeat2:
    while i <= ( poly.polydeg + 1 ) do
      begin
        (* get the coefficients for the polynomial, *)
        (* display them and save them *)
        get_r_num( number, ( row + 3 + i ), 7, abort_command );
        if abort_command then exit;
        poly.polypoly[ i ] := number;

        i := i + 1;
      end;
    (* end while *)

  (* the leading coefficient of the polynomial cannot be zero *)
  if poly.polypoly[ 1 ] = 0 then
    begin
      gotoxy( 20, 0 );
      disp_msg( 62 );
      pause;
    end;

```

FILE: GETF.PAS

```

      gotoxy( 20 ,0 );
      clear_msg( 62 );
      i := 1;
      goto repeat2;
    end;

    (* the poly gain is the input coefficient * leading poly coefficient *)
    poly.coefficient := poly.coefficient * poly.poly[ 1 ];

    (* code insures the leading poly coefficient is 1 *)
    for i := 1 to poly.polydeg + 1 do
      begin
        poly.poly[ 1 + i ] := (poly.poly[ 1 + i ])/(poly.poly[ 1 ]);
      end;
    end;
  end;
end;

(* end procedure *)

(*****
*
* procedure:      roots
* version :      2.0
* date:          6 September 85
* description:   This procedure uses the Birstow's method of find-
*               ing the roots of a polynomial.
*
* global variables used:  degree
* global variables changed:  degree
* global constants used:  maxdeg1
* passed variables:      poly
* returned variables:    poly
* procedures called:      gotoxy, highlight,
*                       disp_msg, nohighlight,
*                       clear_msg, pause
*
* called by:      poly
* author:         Susan K. Mashiko, Capt, USAF
*                Gary C. Tarczynski, Capt, USAF
*
* mod description: Code will no longer overwrite the polynomial.
* modifier:      author
* mod date:      6 September 85
*
* *****)

```

```

procedure roots(var poly : polynomial);
label
  8, 9, 10, 12, 13, 16, 17, 18, 19, 20, 21, 28, 30, 40, 50, 90;
type
  b1 = array[1..30] of real;

```

FILE: GETTF.PAS

```

cl := array[1..30] of real;

var
  u1 : real;
  v1 : real;
  eps1 : real;
  i, k, m : integer;
  real_root : real;
  imag_root : real;
  iteration : integer;
  b : bl;
  c : cl;
  u, v, w, z : real;
  rad : real;
  delu, delv : real;
  denom : real;
  sum, store : real;
  degree : integer;
  coeff_poly : array[1..maxdeg] of real;

begin
  (* initializing values *)
  u1 := 0.0;
  v1 := 0.0;
  eps1 := 1.0E-8;
  (* in order to avoid changing the degree and the coefficients of the *)
  (* polynomial the following dummy variables are used *)
  for i := 1 to (poly.polydeg + 1) do
    coeff_poly[ i ] := poly.poly[ i ];
  degree := poly.polydeg;

  (* Bairstow's method expects the highest coefficient to be 1 *)
  if coeff_poly[ 1 ] <> 0 then
    for i := 1 to degree do
      begin
        coeff_poly[ 1 + i ] := ( coeff_poly[ 1 + i ] ) /
          ( coeff_poly[ 1 ] );
      end;
    coeff_poly[ 1 ] := 1;
  for i := 1 to degree do
    coeff_poly[ i ] := coeff_poly[ i + 1 ];
  (* see if degree of polynomial is 0, 1, or greater than 1 *)
  40: if ( degree - 1 ) < 0 then
    exit;
  if degree-1 = 0 then

```

FILE: GETTF.PAS

```

begin
  real_root := -coeff_poly[1];
  imag_root := 0.0;
  iteration := 1;
  poly.polyfact[ degree ].realpart := real_root;
  poly.polyfact[ degree ].imagpart := imag_root;
  gotoxy( 21, 10 );
  clear_msg( 29 );
  exit;
end
else
  if degree = 2 then goto 8;
  if degree > 2 then goto 13;
  8: u := coeff_poly[1];
  v := coeff_poly[2];
  iteration := 1;
  9: real_root := -( u / 2 );
  rad := sqrt(u - 4.0 * v);
  (* check the sign of (u ** 2 - 4.0 * v) *)
  if rad > 0 then goto 12;
  (* for the case of rad <= 0 continue here *)
  rad := -rad;
  imag_root := sqrt( rad ) / 2.0;
  poly.polyfact[ degree ].realpart := real_root;
  poly.polyfact[ degree ].imagpart := imag_root;
  degree := degree - 1;
  imag_root := - imag_root;
  90: poly.polyfact[ degree ].realpart := real_root;
  poly.polyfact[ degree ].imagpart := imag_root;
  10: degree := degree - 1;
  (* check to see if polydeg is greater than zero *)
  (* if less than or equal to zero exit *)
  if degree <= 0 then
    exit;
  (* polydeg is greater than zero continue *)
  for i := 1 to degree do
    begin
      coeff_poly[i] := b[i];
    end;
  goto 40;
  12: imag_root := sqrt( rad ) / 2;
  w := real_root;

```

FILE: GETTF.PAS

```

z := imag_root;
real_root := real_root + imag_root;
imag_root := 0;
poly.polyfact[degree].realpart := real_root;
poly.polyfact[degree].imagpart := imag_root;
degree := degree - 1;
real_root := w - z;
goto 90;

13: u := ut;
v := vt;
iteration := 1;

(* calculate the b values *)
50: b[1] := coeff_poly[1] - u;
b[2] := coeff_poly[2] - b[1] * u - v;
for k := 3 to degree do
  b[k] := coeff_poly[k] - b[k-1] * u - b[k-2] * v;

(* calculate the c values *)
c[1] := b[1] - u;
c[2] := b[2] - c[1] * u - v;
m := degree - 1;
for k := 3 to m do
  c[k] := b[k] - c[k-1] * u - c[k-2] * v;

(* calculate delu and delv *)
if degree > 3 then goto 17;
denom := c[degree - 1] - sqrt( c[degree - 2] );
if denom = 0 then goto 30;
delu := ( b[degree] - b[degree - 1] * c[degree - 2] ) / denom;
16: delv := ( c[degree - 1] * b[degree - 1] - c[degree - 2] *
  b[degree] ) / denom;
goto 16;

17: denom := c[degree - 1] * c[degree - 3]
  - sqrt( c[degree - 2] );
if denom = 0 then goto 30;
18: delu := ( b[degree] * c[degree - 3] - b[degree - 1]
  * c[degree - 2] ) / denom;
goto 16;

(* calculate new u and v values *)
18: u := u + delu;
v := v + delv;
sum := abs(delu) + abs(delv);

(* store the first sum calculated
if iteration = 1 then goto 19;
goto 20;

```



```

19: store := sum;
20: if iteration = 50 then goto 28;
    if iteration >= 5000 then
    begin
        gotoxy( 21, 1 );
        highlight;
        writeln( ' ITERATING STOPPED AFTER 5000 ITERATIONS ');
        nonhighlight;
        pause;
        gotoxy( 21, 1 );
        writeln( ' ');
        exit;
    end;

21: if sum <= epsi then goto 9;
    if iteration = 100 then
    begin
        highlight;
        gotoxy( 21, 10 );
        disp_msg( 29 );
        nonhighlight;
    end;

    iteration := iteration + 1;
    goto 50;

(* see if sum after 50 iterations exceeds first sum stored *)
28: if sum < store then goto 21;
    gotoxy( 21, 10 );
    clear_msg( 29 );
    gotoxy( 21, 1 );
    highlight;
    writeln( ' ERROR - Unable to converge on root of polynomial ');
    nonhighlight;
    pause;
    gotoxy( 21, 1 );
    writeln( ' Enter correct polynomial ');
    exit;

30: gotoxy( 21, 1 );
    highlight;
    writeln( ' ERROR - Calculated denominator is zero ');
    nonhighlight;
    pause;

```

FILE: GETTF.PAS

```
);
);
```

```
gotoxy( 21, 1 );
writeln(
writeln(
exit;
```

```
end;
```

```
(*****
*
* procedure: poly
* version: 1.2
* date: 16 august 1983
* description: This procedure gets a polynomial in either
* factored or polynomial form.
*
* Global variables used: abort_command
* passed variables: method, poly, disp_row, abort_command
* procedures called: get_unfact, roots,
* get_fact, form_poly,
* get_tf
* called by: vincent m. Parisi II, capt. usaf
* author: Susan K. Mashiko, Capt, USAF
* modified by: Gary C. Tarczynski, Capt, USAF
*
* mod description: display now done by disptf
*
* mod date: 25 Sep 85
*
* *****)
```

```
procedure poly( method : cmdword; var poly : polynomial;
var disp_row : integer; var abort_command : boolean );
begin
```

```
if method = 'POLY' then
begin
get_unfact( poly, disp_row, abort_command );
if abort_command then exit;
roots( poly );
end
else
begin
get_fact( poly, disp_row, abort_command );
if abort_command then exit;
form_poly( poly );
end;
```

```
end;
```

```
(*****
*
* procedure: get_tf
* version: 3.0
*
* *****)
```

```
FILE: GETTF.PAS
```

```

*
*
*   date:          25 September 85
*   description:   This procedure gets a transfer function in
*                 either polynomial or factored form.
*
*   global variables used:  abort_command
*   global variables changed:  abort_command
*   global constants used:  crt_only, as_assigned, max_deg
*   passed variables:
*   files written:
*   TF&POLS.DAT
*   procedures called:
*   clear, gotoxy, out_string,
*   disp_msg, clear_msg,
*   get_int, trim, poly,
*   disptf, pause
*
*   called by:
*   author:      vincent m. parisi ll, capt., usaf
*   modified by: Susan K. Mashiko, Capt. USAF
*               Gary C. Tarczynski, Capt. USAF
*
*   mod description: This modification converted the code from
*                   PASCAL MT+ to TURBO.
*
*   mod date:      27 August 85
*   mod description: Calls disptf instead of internal procedures
*   mod date:      25 Sep 85
*
* *****
*
* procedure disptf( var disp_obj : cmdword ); forward;
*
* procedure get_tf( var def_obj : cmdword; var method : cmdword );
*
* var
*   num_deg      : short_int;
*   denom_deg    : short_int;
*   abort_command : boolean;
*   stor_loc     : integer;
*   disp_row     : integer;
*   numerator    : polynomial;
*   denominator  : polynomial;
*   polys       : file of polynomial;
*   i            : integer;
*
* begin
*
*   abort_command := false;
*   clear;
*   gotoxy( 10, 5 );
*   disp_msg( 2 );
*
*   (* get the degree of the numerator *)

```

FILE: GETTF.PAS

```

repeat
  begin
    gotoxy( 10, 58 );
    out_string( ' ', crt_only );
    gotoxy( 10, 58 );
    get_int( num_deg, abort_command );
    if abort_command then exit;
    if (( num_deg > max_deg ) or ( num_deg < 0 )) then
      begin
        gotoxy( 12, 5 );
        disp_msg( 1 );
      end;
    end;
    until (( num_deg >= 0 ) and ( num_deg <= max_deg ));
  clear_msg( 1 );
  (* get the degree of the denominator *)
repeat
  begin
    gotoxy( 12, 5 );
    disp_msg( 3 );
    gotoxy( 12, 58 );
    out_string( ' ', crt_only );
    gotoxy( 12, 58 );
    get_int( denom_deg, abort_command );
    if abort_command then exit;
    if (( denom_deg > max_deg ) or ( denom_deg < 0 )) then
      begin
        gotoxy( 14, 5 );
        disp_msg( 1 );
      end;
    end;
    until (( denom_deg >= 0 ) and ( denom_deg <= max_deg ));
  clear;
  gotoxy( 0, 27 );
  disp_msg( 8 );
  gotoxy( 1, 37 );
  out_string( def_obj, as_assigned );
  (* get the numerator of the transfer function *)
  gotoxy( 2, 34 );
  disp_msg( 6 );
  disp_row := 3;

```

FILE: GETIF.PAS

```

trim( method );
numerator.polydeg := num_deg;
poly( method, numerator, disp_row, abort_command );
if abort_command then exit;

pause;

(* if the numerator and denominator degree total less than or equal to 6 *)
(* then both are placed on the same screen, if they total greater than *)
(* greater than eight, then get each on a separate screen. Modified to *)
(* prevent over write by the error message *)

if ( num_deg + denom_deg ) <= 6 then
begin
  disp_row := num_deg + 9;
  gotoxy( ( disp_row - 1 ), 33 );
end
else
begin
  clear;
  gotoxy( 1, 36 );
  out_string( def_obj, as_assigned );
  gotoxy( 2, 33 );
  disp_row := 3;
end;

(* get the denominator of the transfer function *)

disp_msg( 7 );
denominator.polydeg := denom_deg;
poly( method, denominator, disp_row, abort_command );
if abort_command then exit;

(* determine the storage location for the transfer function *)

if def_obj = 'OLTF' then stor_loc := 0
else
if def_obj = 'CLTF' then stor_loc := 2
else
if def_obj = 'GTF' then stor_loc := 4
else
if def_obj = 'HTF' then stor_loc := 6
else
if def_obj = 'TF1' then stor_loc := 8
else
if def_obj = 'TF2' then stor_loc := 10
else
if def_obj = 'TF3' then stor_loc := 12
else

```

FILE: GETTF.PAS

```

if def_obj = 'TF4' then stor_loc := 14
else
if def_obj = 'TFS' then stor_loc := 16;

(* Now save the transfer function in the file *)

assign( polys, 'tf&polys.dat' );
reset( polys );

seek( polys, stor_loc );
write( polys, numerator );
seek( polys, ( stor_loc + 1 ) );
write( polys, denominator );

(* commented out by Mashiko and Tarczynski to convert to TURBO *)
(* polys^:= numerator;
(* seekwrite( polys, stor_loc );
(* polys^:= denominator;
(* seekwrite( polys, ( stor_loc + 1 ) );

close( polys );
clear;
disptf( def_obj );

end;

```

FILE: GETTF.PAS

```

*****
**
** file: HELP.PAS
** procedure contained: help
** version: 3.0
** date: 1 November 1985
** description: This file contains the procedure that
** handles the logic for providing on-line help.
** author: vincent m. parisi ii, capt., usaf
** Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt, USAF
**
**
*****
(*****
*
* procedure: help
* version: 3.0
* date: 1 November 1985
* description: This procedure handles the logic for
* providing on line help. The valid command
* is scanned to determine what help is
* requested. The display message routine is
* then called with the number of the help
* message requested.
*
* global variables used: cmdbuffer
* global constants used: wordsize, buffersize
* passed variables: cmdbuffer, wordnumber
* procedures called: pause, clear, disp_msg, trim
* called by: select_routine
* author: vincent m. parisi ii, capt., usaf
* modified by: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
*
* mod description: The entire procedure was replaced with
* code customized for ICECAP.
*
* mod date: 9 August 1985
* mod description: Changed the help calls to correspond with
* the changed menu
*
* mod date: 18 September 1985
* mod description: Same as 18 Sep 85
*
* mod date: 1 Nov 85
*
*****
)

```

```

procedure help( var cmdbuffer ; buffer; wordnumber ; integer );
const

```

```

FILE: HELP.PAS

```

```

system_msg = 15;
change_msg = 16;
copy_msg   = 17;
define_msg = 18;
display_msg = 19;
form_msg   = 20;
print_msg  = 22;
recover_msg = 23;
stop_msg   = 24;
mod_msg    = 25;
swit_msg   = 26;
update_msg = 27;
help_msg   = 61;

var help_obj : cmdword;

begin
  help_obj := cmdbuffer( wordnumber );
  trim( help_obj );
  clear;

  if help_obj = 'SYSTEM' then disp_msg( system_msg )
  else
  if help_obj = 'CHANGE' then disp_msg( change_msg )
  else
  if help_obj = 'COPY' then disp_msg( copy_msg )
  else
  if help_obj = 'DEFINE' then disp_msg( define_msg )
  else
  if help_obj = 'DISPLAY' then disp_msg( display_msg )
  else
  if help_obj = 'FORM' then disp_msg( form_msg )
  else
  if help_obj = 'PRINT' then disp_msg( print_msg )
  else
  if help_obj = 'RECOVER' then disp_msg( recover_msg )
  else
  if help_obj = 'STOP' then disp_msg( stop_msg )
  else
  if help_obj = 'MODIFY' then disp_msg( mod_msg )
  else
  if help_obj = 'SWITCHES' then disp_msg( swit_msg )
  else
  if help_obj = 'UPDATE' then disp_msg( update_msg )
  else
  if help_obj = 'HELP' then disp_msg( help_msg );
  pause;
  clear;

```

FILE: HELP.PAS

end;

FILE: HELP.PAS

```

*****
**
** file: ICECAPPC.PAS ** Z100
** program contained: ICECAPPC ** floppy disk version
** version: 9.00 *****
** date: 11 Oct 1985
** description: This file contains the main program for
** the MICROSDW menu system and the subroutines
** for ICECAP-PC. These subroutines comprise a
** CAD package for control system design and
** analysis.
** author: Gary C. Tarczynski, Capt, USAF
** Susan K. Mashiko, Capt, USAF
**
*****

```

```

*****
**
** program: ICECAPPC ** Z100
** version: 9.0 ** floppy disk version
** date: 11 Oct 1985 *****
** description: This program provides a flexible user
** interface for software development or
** other applications. This program also con-
** tains the ICECAP-PC subroutines.
** procedures called: get_cmd, pause, ClearScreen, get_data,
** select_routine
** authors: Gary C. Tarczynski, Capt, USAF
** Susan K. Mashiko, Capt, USAF
**
*****

```

```

program ICECAPPC;
($I msdwcons.pas)
($I msdwtype.pas)

const
  wordsize = 12;
  buffersize = 6;
  stat_line_width = 77;
  crt_only = 'c';
  terminal_only = 't';
  as_assigned = 'a';
  backspace = 8;
  del = 127;
  yes = true;

```

FILE: ICECAPPC.PAS *** Z100 floppy disk version ***

```

no                = false;
abort_str        = '$';

type
term_array = array[ 1..term_length ] of byte;
print_array = array[ 1..printer_length ] of byte;
msg_array = array[ 1..num_msg_dir ] of msg;
buffer = array[ 1..buffer_size ] of string[ wordsize ];
cmdword = string[ wordsize ];
msg_line = string[ screen_width ];
dictionary = record
dictword : cmdword;
matchp : integer;
nomatchp : integer;
abbrev : byte; (* minimum length of abbreviation *)
end;

var
cmdbuffer : buffer; (* buffer of command words *)
blanks : string[ screen_width ];
status_line : string[ stat_line_width ];
call_routine : cmdword;
abort_command : boolean;
trans : boolean;
printer : boolean;
temp : boolean;
prt : boolean;
macro_error : boolean;
show_abbreviation : boolean;
in_terminal : boolean;
stat_on : boolean;
macro_file : text;
trans_file : text;
list_dev : text;
temp_file : text;
real_error : byte;
help_level : byte;
term : term_array;
print : print_array;
msg_dir : msg_array;
decode_dict : dict buffer;
msg_tx : file of msg_line;
string : msg_line;
decode : dictionary;
list_dev_name : paramstring;
trans_file_name : paramstring;

```

FILE: ICECAPPC.PAS *** Z100 floppy disk version ***

```

macro_file_name : paramstring:
number_of_commands : integer:

($I concons.pas ) (* added 14 Aug 85 these declarations are *)
(* unique to the controls package ICECAP *)

(*****
* Include the sources for the routines called by MICROSDW *
*****
($I ucase.pas )
($I terminal.pas )
($I output.pas )
($I pause.pas )
($I getdat.pas )
($I msg.pas )
($I instruct.pas )
($I getline.pas )
($I prompthe.pas )
($I promptcm.pas )
($I trim.pas )
($I display.pas )
($I getint.pas )
($I getstrin.pas )
($I readcom.pas )
($I proceser.pas )
($I valndec.pas )
($I getcom.pas )

($I b:recover.pas ) (* added 9 Sep 85 *)
($I b:update.pas ) (* added 9 Sep 85 *)
($I b:copy.pas ) (* added 5 Sep 85 *)
($I b:help.pas ) (* added 9 Aug 85 *)
($I b:reals.pas ) (* added 13 Aug 85 *)
($I b:gettf.pas ) (* added 13 Aug 85 *)
($I b:getmat.pas ) (* added 11 Sep 85 *)
($I b:matrixman.pas ) (* added 20 Sep 85 *)
($I b:matrix.pas ) (* added 20 Sep 85 *)
($I b:polyman.pas ) (* added 4 Sep 85 *)
($I b:poly.pas ) (* added 5 Sep 85 *)
($I b:form.pas ) (* added 7 Oct 85 *)
($I b:define.pas ) (* added 12 Aug 85 *)
($I b:inroot.pas ) (* added 8 Sep 85 *)
($I b:delroot.pas ) (* added 7 Sep 85 *)
($I b:modify.pas ) (* added 23 Sep 85 *)
($I b:disp.pas ) (* added 4 Sep 85 *)
($I b:bode.pas ) (* added 11 Oct 85 *)
($I b:select.pas ) (* modified Sep 85 *)

```

```

(*****
*      main program code
*      *****)
begin      (* begin main program *)
(*****
*      initialize the program; read in all the initializing
*      parameters, the command syntax data structure. put up
*      title slide to show CRT interface is working and give
*      user something to look at.
*      Also initialize all files used by the MICROSDW.
*      *****)
get_data( term, print, msg_dir, decode_dict, printer,
          trans, temp, crt_show_abbreviation, in_terminal,
          stat_on, macro_error_help_level,
          list_dev_name, trans_file_name, macro_file_name );

pause;

(*****
*      begin main logic statements
*      *****)
repeat
  get_cmd( cmdbuffer, call_routine, number_of_commands );
  select_routine( call_routine, cmdbuffer, number_of_commands );
until (call_routine = 'STOP');
ClearScreen;
end.      (* end of main program *)

```

```

*****
** file: ICEAPP.PAS ** Z100
** program contained: ICECAPPC ** hard disk version **
** version: 9.00 *****
** date: 11 Oct 1985 **
** description: This file contains the main program for
** the MICROSDW menu system and the subroutines**
** for ICECAP-PC. These subroutines comprise a
** CAD package for control system design and
** analysis.
** author: Gary C. Tarczynski, Capt, USAF
** Susan K. Mashiko, Capt, USAF
**
*****
)

```

```

*****
** program: ICECAPPC * Z100
** version: 9.0 * hard disk version
** date: 11 Oct 1985 *****
** description: This program provides a flexible user
** interface for software development or
** other applications. This program also con-
** tains the ICECAP-PC subroutines.
** procedures called: get_cmd, pause, ClearScreen, get_data,
** select_routine
** authors: Gary C. Tarczynski, Capt, USAF
** Susan K. Mashiko, Capt, USAF
**
*****
)

```

```

program ICECAPPC;
($I msdwcons.pas)
($I msdwtype.pas)

const
  wordsize = 12;
  buffersize = 6;
  stat_line_width = 77;
  crt_only = 'c';
  terminal_only = 't';
  as_assigned = 'a';
  backspace = 8;
  del = 127;
  yes = true;

FILE: ICEAPP.PAS *** Z100 hard disk version ***

```

```

no          = false;
abort_str  = '$';

type
term_array = array[ 1..term_length ] of byte;
print_array = array[ 1..printer_length ] of byte;
msg_array  = array[ 1..num_msg_dir ] of msg;
buffer     = array[ 1..buffersize ] of string[ wordsize ];

cmdword    = string[ wordsize ];
msg_line   = string[ screen_width ];

dictionary = record
dictword  : cmdword;
matchp   : integer;
nomatchp : integer;
abbrev   : byte;      (* minimum length of abbreviation *)
end;

var
cmdbuffer : buffer;      (* buffer of command words *)
blanks    : string[ screen_width ];
status_line : string[ stat_line_width ];
call_routine : cmdword;
abort_command : boolean;
trans       : boolean;
printer     : boolean;
temp       : boolean;
crt        : boolean;
macro_error : boolean;
show_abbreviation : boolean;
in_terminal : boolean;
stat_on     : boolean;
macro_file : text;
trans_file : text;
list_dev   : text;
temp_file  : text;
real_error : byte;
help_level : byte;
term       : term_array;
print      : print_array;
msg_dir   : msg_array;
decode_dict : dict_buffer;
msg_txt   : file of msg_line;
string    : msg_line;
decode    : dictionary;
list_dev_name : paramstring;
trans_file_name : paramstring;

```

FILE: ICECAPPC.PAS *** Z100 hard disk version ***

```

macro_file_name : paramstring;
number_of_commands : integer;

($I concons.pas ) (* added 14 Aug 85 these declarations are *)
(* unique to the controls package ICECAP *)

(*****
* Include the sources for the routines called by MICROSDW *
*****
($I ucase.pas )
($I terminal.pas )
($I output.pas )
($I pause.pas )
($I getdat.pas )
($I msg.pas )
($I instruct.pas )
($I getline.pas )
($I prompthe.pas )
($I promptcm.pas )
($I trim.pas )
($I displays.pas )
($I getint.pas )
($I getstrin.pas )
($I readcom.pas )
($I proceser.pas )
($I valdec.pas )
($I getcom.pas )

($I recover.pas ) (* added 9 Sep 85 *)
($I update.pas ) (* added 9 Sep 85 *)
($I copy.pas ) (* added 5 Sep 85 *)
($I help.pas ) (* added 9 Aug 85 *)
($I reals.pas ) (* added 13 Aug 85 *)
($I gettf.pas ) (* added 11 Sep 85 *)
($I getmat.pas ) (* added 20 Sep 85 *)
($I matrix.pas ) (* added 20 Sep 85 *)
($I polyman.pas ) (* added 4 Sep 85 *)
($I poly.pas ) (* added 5 Sep 85 *)
($I form.pas ) (* added 7 Oct 85 *)
($I define.pas ) (* added 12 Aug 85 *)
($I inroot.pas ) (* added 8 Sep 85 *)
($I delroot.pas ) (* added 7 Sep 85 *)
($I modify.pas ) (* added 23 Sep 85 *)
($I disp.pas ) (* added 4 Sep 85 *)
($I bode.pas ) (* added 11 Oct 85 *)
($I select.pas ) (* modified Sep 85 *)

```



```

(*****
*
* main program code
*
*****
)

begin      (* begin main program *)
(*****
*
* initialize the program; read in all the initializing
* parameters, the command syntax data structure. put up
* title slide to show CRT interface is working and give
* user something to look at.
* Also initialize all files used by the MICROSDW.
*
*****
)

get_data( term, print, msg_dir, decode_dict, printer,
          trans, temp, crt, show_abbreviation, in_terminal,
          stat_on, macro_error, help_level,
          list_dev_name, trans_file_name, macro_file_name );

pause;

(*****
*
* begin main logic statements
*
*****
)

repeat
  get_cmd( cmdbuffer, call_routine, number_of_commands );
  select_routine( call_routine, cmdbuffer, number_of_commands );
until (call_routine = 'Stop');
ClearScreen;
end.      (* end of main program *)

```

```

*****
** file: ICECAPPC.PAS
** program contained: ICECAPPC
** version: 9.00
** date: 11 Oct 1985
** description: This file contains the main program for
the MICROSDW menu system and the subroutines
for ICECAP-PC. These subroutines comprise a
CAD package for control system design and
analysis.
** author: Gary C. Tarczynski, Capt, USAF
Susan K. Mashiko, Capt, USAF
**
*****
)

```

```

*****
** program: ICECAPPC
** version: 9.0
** date: 11 Oct 1985
** description: This program provides a flexible user
interface for software development or
other applications. This program also con-
tains the ICECAP-PC subroutines.
** procedures called: get_cmd, pause, ClearScreen, get_data,
select routine
** authors: Gary C. Tarczynski, Capt, USAF
Susan K. Mashiko, Capt, USAF
**
*****
)

```

```

program ICECAPPC;

```

```

($I msdwcons.pas)
($I msdwtype.pas)

```

```

const
  wordsize = 12;
  buffersize = 6;
  stat_line_width = 77;
  crt_only = 'c';
  terminal_only = 't';
  as_assigned = 'a';
  backspace = 8;
  del = 127;
  yes = true;

```

```

FILE: ICECAPPC.PAS *** IBM ONLY floppy disk version ***

```

```

no          = false;
abort_str  = '$';

type
term_array = array[ 1..term_length ] of byte;
print_array = array[ 1..printer_length ] of byte;
msg_array  = array[ 1..num_msg_dir ] of msg;
buffer     = array[ 1..buffersize ] of string[ wordsize ];
cmdword   = string[ wordsize ];
msg_line  = string[ screen_width ];

dictionary = record
dictword  : cmdword;
matchp   : integer;
nomatchp : integer;
abbrev   : byte;      (* minimum length of abbreviation *)
end;

```

```
(*INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT***)
```

This type declaration was added by Tarczynski and Mashiko to be able to use the MS-DOS function call in the procedure standard_output.*)

```

regpak = record
al,ah,bl,bh,cl,ch,dl,dh : byte;
ax,bx,cx,dx,ip,sp,si,di,ds,es,flags : integer;
end;

```

```
(*INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT***)
```

```

var
cmdbuffer : buffer;      (* buffer of command words *)
blanks   : string[ screen_width ];
status_line : string[ stat_line_width ];
call_routine : cmdword;
abort_command : boolean;
trans       : boolean;
printer     : boolean;
temp       : boolean;
crt        : boolean;
macro_error : boolean;
show_abbreviation : boolean;
in_terminal : boolean;
stat_on    : boolean;
macro_file : text;
trans_file : text;

```

```
FILE: ICECAPPC.PAS      *** IBM ONLY floppy disk version ***
```

```

list_dev      : text;
temp_file    : text;
real_error   : byte;
help_level   : byte;
term         : term_array;
print        : msg_array;
msg_dir      : dict_buffer;
decode_dict  : file_of_msg_line;
string       : msg_line;
decode       : dictionary;
list_dev_name : paramstring;
trans_file_name : paramstring;
macro_file_name : paramstring;
number_of_commands : integer;

($I concons.pas ) (* added 14 Aug 85 these declarations are *)
(* unique to the controls package ICECAP *)

(*****
 * Include the sources for the routines called by MICROSDW *
 *****)

(*INSERT***INSERT***INSERT***INSERT***INSERT***INSERT***INSERT**

```

This statement was added by Tarczynski and
Mashiko to include the file containing the
procedure standard_output.*

```

($I stdout.pas)

(*INSERT***INSERT***INSERT***INSERT***INSERT***INSERT***INSERT*
($I ucase.pas )
($I terminal.pas )
($I output.pas )
($I pause.pas )
($I getdat.pas )
($I msg.pas )
($I instruct.pas )
($I getline.pas )
($I prompthe.pas )
($I promptcm.pas )
($I trim.pas )
($I displayc.pas )
($I getint.pas )
($I getstrin.pas )
($I readcom.pas )
($I proceser.pas )

```

```

($I valndec.pas )
($I getcom.pas )

($I b:recover.pas ) (* added 9 Sep 85 *)
($I b:update.pas ) (* added 9 Sep 85 *)
($I b:copy.pas ) (* added 5 Sep 85 *)
($I b:help.pas ) (* added 9 Aug 85 *)
($I b:reals.pas ) (* added 13 Aug 85 *)
($I b:gettf.pas ) (* added 13 Aug 85 *)
($I b:getmat.pas ) (* added 11 Sep 85 *)
($I b:matrxman.pas ) (* added 20 Sep 85 *)
($I b:matrix.pas ) (* added 20 Sep 85 *)
($I b:poly.pas ) (* added 4 Sep 85 *)
($I b:form.pas ) (* added 7 Oct 85 *)
($I b:define.pas ) (* added 12 Aug 85 *)
($I b:inroot.pas ) (* added 8 Sep 85 *)
($I b:delroot.pas ) (* added 7 Sep 85 *)
($I b:modify.pas ) (* added 23 Sep 85 *)
($I b:disp.pas ) (* added 4 Sep 85 *)
($I b:bode.pas ) (* added 11 Oct 85 *)
($I b:iselect.pas ) (* modified Sep 85 *)

(*****
* main program code
* *****)

begin (* begin main program *)

(*
* initialize the program; read in all the initializing
* parameters, the command syntax data structure. put up
* title slide to show CRT interface is working and give
* user something to look at.
* Also initialize all files used by the MICROSDW.
* *****)

(*INSERT***INSERT***INSERT***INSERT***INSERT***INSERT***INSERT**
This statement was added by Tarczynski and
Mashiko to reroute screen output through the
procedure standard_output. From now on, all
output to the terminal via WRITE or WRITELN
will be redirected through ANSI.SYS, thereby
allowing the IBM PC to recognize escape codes.*)

ConOutPtr := ofs(standard_output);

```

FILE: ICECAPPC.PAS *** IBM ONLY floppy disk version ***

```
(*INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT*)
get_data( term, print, msg_dir, decode_dict, printer,
          trans, temp, crt, show_abbreviation, in_terminal,
          stat_on, macro_error, help_level,
          list_dev_name, trans_file_name, macro_file_name );
pause;
(*****
 *          begin main logic statements
 *****)
repeat
  get_cmd( cmbuffer, call_routine, number_of_commands );
  select_routine( call_routine, cmbuffer, number_of_commands );
until (call_routine = 'STOP');
ClearScreen;
end. (* end of main program *)
```

```

*****
** file: ICECAPPC.PAS ** IBM ONLY **
** program contained: ICECAPPC ** hard disk version **
** version: 9.00 **
** date: 11 Oct 1985 **
** description: This file contains the main program for **
** the MICROSDW menu system and the subroutines **
** for ICECAP-PC. These subroutines comprise a **
** CAD package for control system design and **
** analysis. **
** author: Gary C. Tarczynski, Capt, USAF **
** Susan K. Mashiko, Capt, USAF **
*****
)

```

```

*****
** program: ICECAPPC ** IBM ONLY **
** version: 9.0 ** hard disk version **
** date: 11 Oct 1985 **
** description: This program provides a flexible user **
** interface for software development or **
** other applications. This program also con- **
** tains the ICECAP-PC subroutines. **
** procedures called: get_cmd, pause, ClearScreen, get_data, **
** select_routine **
** authors: Gary C. Tarczynski, Capt, USAF **
** Susan K. Mashiko, Capt, USAF **
*****
)

```

```

program ICECAPPC;

```

```

($I msdwcons.pas)
($I msdwtype.pas)

```

```

const
  wordsize = 12;
  buffersize = 6;
  stat_line_width = 77;
  crt_only = 'c';
  terminal_only = 't';
  as_assigned = 'a';
  backspace = 8;
  del = 127;
  yes = true;

```

```

FILE: ICECAPPC.PAS *** IBM ONLY hard disk version ***

```

```

no          = false;
abort_str  = '$';

type
term_array = array[ 1..term_length ] of byte;
print_array = array[ 1..printer_length ] of byte;
msg_array   = array[ 1..num_msg_dir ] of msg;

buffer     = array[ 1..bufferize ] of string[ wordsize ];

cmdword    = string[ wordsize ];
msg_line   = string[ screen_width ];

dictionary = record
dictword   : cmdword;
matchp    : integer;
nomatchp  : integer;
abbrev    : byte;      (* minimum length of abbreviation *)
end;

```

```
(*INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**
```

```

This type declaration was added by Tarczynski
and Mashiko to be able to use the MS-DOS
function call in the procedure standard_output.*

```

```

regpak = record
al,ah,bl,bh,cl,ch,dl,dh : byte;
ax,bx,cx,dx,sp,si,di,ds,es,flags : integer;
end;

```

```
(*INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**
```

```

var
cmdbuffer   : buffer;      (* buffer of command words *)
blanks     : string[ screen_width ];
status_line : string[ stat_line_width ];
call_routine : cmdword;
abort_command : boolean;
trans      : boolean;
printer    : boolean;
temp      : boolean;
crt       : boolean;
macro_error : boolean;
show_abbreviation : boolean;
in_terminal : boolean;
stat_on    : boolean;
macro_file : text;
trans_file : text;

```

```
FILE: ICECAPPC.PAS      *** IBM ONLY hard disk version ***
```



```

list_dev      : text;
temp_file    : text;
real_error   : byte;
help_level   : byte;
term         : term_array;
print        : print_array;
msg_dir      : msg_array;
decode_dict  : dict_buffer;
msg_txt      : file of msg_line;
string       : msg_line;
decode       : dictionary;
list_dev_name : paramstring;
trans_file_name : paramstring;
macro_file_name : paramstring;
number_of_commands : integer;

```

```

($I concns.pas ) (* added 14 Aug 85 these declarations are *)
                  (* unique to the controls package ICECAP *)

```

```

(*****
 * Include the sources for the routines called by MICROSDW *
 *****)

```

```

(*INSERT***INSERT***INSERT***INSERT***INSERT***INSERT**

```

This statement was added by Tarczynski and
Mashiko to include the file containing the
procedure standard_output.*)

```

($I stdout.pas)

```

```

(*INSERT***INSERT***INSERT***INSERT***INSERT***INSERT**

```

```

($I ucase.pas )
($I terminal.pas )
($I output.pas )
($I pause.pas )
($I getdat.pas )
($I msg.pas )
($I instruct.pas )
($I getline.pas )
($I prompthe.pas )
($I promptcm.pas )
($I trim.pas )
($I displayc.pas )
($I getint.pas )
($I getstrin.pas )
($I readcom.pas )
($I proceser.pas )

```

```

($I valdec.pas )
($I getcom.pas )

($I recover.pas ) (* added 9 Sep 85 *)
($I update.pas ) (* added 9 Sep 85 *)
($I copy.pas ) (* added 5 Sep 85 *)
($I help.pas ) (* added 9 Aug 85 *)
($I reals.pas ) (* added 13 Aug 85 *)
($I getf.pas ) (* added 13 Aug 85 *)
($I getmat.pas ) (* added 11 Sep 85 *)
($I matrman.pas ) (* added 20 Sep 85 *)
($I matrix.pas ) (* added 20 Sep 85 *)
($I polyman.pas ) (* added 4 Sep 85 *)
($I poly.pas ) (* added 5 Sep 85 *)
($I form.pas ) (* added 7 Oct 85 *)
($I define.pas ) (* added 12 Aug 85 *)
($I inroot.pas ) (* added 8 Sep 85 *)
($I delroot.pas ) (* added 7 Sep 85 *)
($I modify.pas ) (* added 23 Sep 85 *)
($I disp.pas ) (* added 4 Sep 85 *)
($I bode.pas ) (* added 11 Oct 85 *)
($I select.pas ) (* modified Sep 85 *)

(*****
* main program code
*
*****

begin (* begin main program *)

(*****
*
* initialize the program; read in all the initializing
* parameters, the command syntax data structure. put up
* title slide to show CRT interface is working and give
* user something to look at.
* Also initialize all files used by the MICROSDW.
*
*****
(*INSERT***INSERT***INSERT***INSERT***INSERT***INSERT**
This statement was added by Tarczynski and
Mashiko to reroute screen output through the
procedure standard_output. From now on, all
output to the terminal via WRITE or WRITELN
will be redirected through ANSI.SYS, thereby
allowing the IBM PC to recognize escape codes.*)

ConOutPtr := OfS(standard_output);

```

FILE: ICEAPPC.PAS *** IBM ONLY hard disk version ***

```
(*INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT**INSERT*)
```

```
get_data( term, print, msg_dir, decode_dict, printer,  
          trans, temp, crt, show_abbreviation, in_terminal,  
          stat_on, macro_error, help_level,  
          list_dev_name, trans_file_name, macro_file_name );
```

```
pause;
```

```
(*****  
*          begin main logic statements  
*****)
```

```
repeat  
  get_cmd( cmdbuffer, call_routine, number_of_commands );  
  select_routine( call_routine, cmdbuffer, number_of_commands );  
until (call_routine = 'STOP');  
ClearScreen;  
end. (* end of main program *)
```

```

*****
**      file:      INROOT.PAS
**      procedure contained: inroot
**      version:   2.0
**      date:      19 September 85
**      description: This file contains the procedures to insert
**                   a root from a polynomial. If the root is com-
**                   plex the conjugate will also be added.
**      author:    Susan K. Mashiko, Capt, USAF
**                   Gary C. Tarczynski, Capt, USAF
**
*****

```

```

(*****
**      procedure: inroot
**      version:   2.0
**      date:      19 September 85
**      description: This procedure will insert a root from a polyno-
**                   mial. If the root is complex the conjugate will
**                   also be inserted.
**      global variables used:  cmdbuffer, abort_command
**      global constants used:  crt_only, as_assigned
**      passed variables:      cmdbuffer, wordnumber
**      files read:            TF&POL.S:DAT
**      files written:         TF&POL.S:DAT
**      procedures called:     clear, gotoxy, disppoly,
**                               trim, out_real, out_string,
**                               get_r_num, out_int, form_poly,
**                               pause
**
**      called by:  select
**      author:    Susan K. Mashiko, Capt, USAF
**                   Gary C. Tarczynski, Capt, USAF
**      mod description: Changes wer made to he code to compensate for
**                       a call from a lower level
**      modified by:  author
**      mod date:    19 Sep 85
**
*****

```

```

overlay procedure inroot( var cmdbuffer : buffer; var wordnumber : integer );

```

```

var
choice      : cmdword;
polys      : file of polynomial;
i          : integer;

```

```

FILE: INROOT.PAS

```

```

oldpoly      : polynomial;
newpoly      : polynomial;
stor_loc     : integer;
number       : real;

begin
clear;
choice := cndbuffer[ 3 ];
trim( choice );
if choice = 'POLYA' then stor_loc := 18
else
if choice = 'POLVB' then stor_loc := 19
else
if choice = 'POLYC' then stor_loc := 20
else
if choice = 'POLVD' then stor_loc := 21
else
if choice = 'POLVE' then stor_loc := 22
else
if choice = 'ONPOLY' then stor_loc := 0
else
if choice = 'ODPOLY' then stor_loc := 1
else
if choice = 'CNPOLY' then stor_loc := 2
else
if choice = 'CDPOLY' then stor_loc := 3
else
if choice = 'GNPOLY' then stor_loc := 4
else
if choice = 'GDPOLY' then stor_loc := 5
else
if choice = 'HNPOLY' then stor_loc := 6
else
if choice = 'HDPOLY' then stor_loc := 7;

(* pull the desired polynomial from storage *)
assign( polys, 'tf&polys.dat' );
reset( polys );
seek( polys, stor_loc );
read( polys, oldpoly );
close( polys );

(* display the polynomial *)
disppoly( choice );

(* copy all of the existing roots into the new polynomial *)
for i := 1 to oldpoly.polydeg do
begin
newpoly.polyfact[ i ].realpart := oldpoly.polyfact[ i ].realpart;

```

FILE: INROOT.PAS

```

newpoly.polyfact[ 1 ].imagpart := oldpoly.polyfact[ 1 ].imagpart;
end;

(* add the new root number to the form on the screen *)
gotoxy( ( 7 + oldpoly.polydeg ), 40 );
out_int( ( oldpoly.polydeg + 1 ), 2, crt_only );

(* add the j to the form on the screen *)
gotoxy( ( 7 + oldpoly.polydeg ), 57 );
out_string( 'j', as_assigned );

(* get the new roots from the terminal *)
get_r_num( number, ( 7 + oldpoly.polydeg ), 43, abort_command );
if abort_command then exit;
newpoly.polyfact[ oldpoly.polydeg + 1 ].realpart := number;

(* get the imaginary portion of the pole or zero for this root *)
get_r_num( number, ( 7 + oldpoly.polydeg ), 59, abort_command );
if abort_command then exit;
newpoly.polyfact[ oldpoly.polydeg + 1 ].imagpart := number;

(* If the root is complex the conjugate will be generated and *)
(* displayed. The negative conjugate will be displayed first *)
if ( ( number < -0.000001 ) or ( number > 0.000001 ) ) then
begin
newpoly.polyfact[ oldpoly.polydeg + 2 ].realpart :=
newpoly.polyfact[ oldpoly.polydeg + 1 ].realpart;
gotoxy( ( 8 + oldpoly.polydeg ), 43 );
out_real( newpoly.polyfact[ oldpoly.polydeg + 2 ].realpart,
12, as_assigned);
gotoxy( ( 7 + oldpoly.polydeg ), 59 );
if number < 0.0 then
begin
newpoly.polyfact[ oldpoly.polydeg + 1 ].imagpart := number;
out_real( number, 12, as_assigned );
newpoly.polyfact[ oldpoly.polydeg + 2 ].imagpart :=
( number * (-1.0) );
end
else
begin
newpoly.polyfact[ oldpoly.polydeg + 1 ].imagpart :=
( number * (-1.0) );
out_real( number, 12, as_assigned );
newpoly.polyfact[ oldpoly.polydeg + 2 ].imagpart := number;
end;
gotoxy( ( 8 + oldpoly.polydeg ), 59 );
out_real( newpoly.polyfact[ oldpoly.polydeg + 1 ].imagpart,
12, as_assigned );
newpoly.polydeg := oldpoly.polydeg + 2;

```

FILE: INROOT.PAS

```
end
else
  newpoly.polydeg := oldpoly.polydeg + 1;
  newpoly.coefficient := oldpoly.coefficient;
  (* form the polynomial *)
  form_poly( newpoly );
  (* store the new polynomial in the same stor_loc as the old *)
  assign( polys, 'tf&polys.dat' );
  reset( polys );
  seek( polys, stor_loc );
  write( polys, newpoly );
  (* display the new polynomial *)
  dispoly( choice );
  pause;
end;
```

FILE: INROOT.PAS

```

*****
**
**      file:          INSTRUCT.PAS
**
**      procedures contained: instruction
**      version:       1.1
**      date:          16 August 1983
**      description:   This file contains the procedure that
**                    issues instructions for entering command-
**                    words.
**      author:        vincent m. parisi ii, capt., usaf
**
*****
*****
*****
**      procedure:    instruction
**      version:      1.1
**      date:         16 august 1983
**      description:  This procedure issues the appropriate
**                    instruction for entering a command
**                    based on the number of command words
**                    already entered.
**      passed variables:  level, instr_row, instr_col
**      procedures called:  out_string
**                        getcom
**      called by:        vincent m. parisi ii, capt., usaf
**
*****
*****
*****
procedure instruction
  ( level : integer; instr_row : integer; instr_col : integer );
begin
  if level = 1 then
    out_string( 'Enter one of the following initial command words....', 'c' )
  else
    out_string
      out_( 'Now enter one of these commandwords, or "$" to abort...', 'c' );
  end;

```

FILE: INSTRUCT.PAS


```

*****
**
** file: MATRIX.PAS
** procedures contained: disp_matrix,  matrix_manip1,
**                      matrix_manip2,  get_matrix_name,
**                      mmatrix
**
** version: 1.0
** date: 22 Sep 85
** description: This file contains the procedures to display and
**              manipulate matrices.
** author: Susan K. Mashiko, Capt, USAF
**          Gary C. Tarczynski, Capt, USAF
**
*****

```

```

*****
**
** procedure: disp_matrix
** version: 1.0
** date: 20 September85
** description: This procedure will display a matrix from a
**              record in matrix.dat The user should place a pause
**              in his/her code after the calling subroutine to
**              keep the display on the screen.
**
** global constants used:  as_assigned
**
** passed variables:      choice
**
** returned variables:   choice
**
** files read:           MATRIX.DAT
**
** procedures called:    clear,          gotoxy,
**                      out_string, disp_msg,
**                      make_pretty_small_matrix,
**                      make_pretty_large_matrix_one,
**                      make_pretty_large_matrix_two,
**                      out_real,
**                      pause
**
** called by:            matrix_manip1, matrix_manip2,  mmatrix
** author: Susan K. Mashiko, Capt, USAF
**          Gary C. Tarczynski, Capt, USAF
**
*****

```

```

procedure disp_matrix( var choice : cword );
var
stor_loc : integer;
mats : file of matrix;
col_element : integer;

```

FILE: MATRIX.PAS

```

j      : integer;
number : integer;
mat     : real;
num_row : integer;
num_col : integer;
row     : integer;
col     : integer;

begin
  trim( choice );
  if choice = 'MATA' then stor_loc := 0
  else
  if choice = 'MATB' then stor_loc := 1
  else
  if choice = 'MATC' then stor_loc := 2
  else
  if choice = 'MATD' then stor_loc := 3
  else
  if choice = 'MATE' then stor_loc := 4;

  assign( mats, 'matrix.dat' );
  reset( mats );
  seek( mats, stor_loc );
  read( mats, mat );
  close( mats );

  (* put the title on the screen *)
  clear;
  gotoxy( 1, 32 );
  disp_msg( 50 );
  gotoxy( 2, 37 );
  out_string( choice, as_assigned );
  num_col := mat.num_cols;
  num_row := mat.num_rows;
  row := 5;
  col := 10;

  (* if the matrix is small (i.e. 5 columns or less) display *)
  if num_col <= 5 then
  begin
    make_pretty_small_matrix( num_row, num_col );
    for j := 0 to ( num_col - 1 ) do
    begin
      col_element := j + 1;
      for i := 1 to num_row do
      begin
        gotoxy( ( row + i ), ( col + ( j * 13 ) ) );
        out_real( mat.element[ i, col_element ], 12, as_assigned );
      end
    end
  end

```

FILE: MATRIX.PAS

```

end;
end;
else
begin
(* display the first page of a large matrix *)
make_pretty_large_matrix_one( num_row, num_col );
for j := 0 to 4 do
begin
col_element := j + 1;
for i := 1 to num_row do
begin
gotoxy( ( row + i ), ( col + ( j * 13 ) ) );
out_real( mat_element[ i, col_element ], 12, as_assigned );
end;
end;
pause;
clear;
(* display the second page of a large matrix *)
make_pretty_large_matrix_two( num_row, num_col );
for j := 0 to num_col - 6 do
begin
col_element := j + 6;
for i := 1 to num_row do
begin
gotoxy( ( row + i ), ( col + ( j * 13 ) ) );
out_real( mat_element[ i, col_element ], 12, as_assigned );
end;
end;
end;
end;
end;

```

```

end;
end;
(*****
* procedure:  matrx_manipl
* version:    1.0
* date:       21 September 85
* description: This procedure will add, subtract, or multiply two
*              matrices.
* Global variables used:
* passed variables:
* files read:
* files written:
* procedures called:
*              abort_command
*              first_second_result_mat_obj
*              MATRIX.DAT
*              MATRIX.DAT
*              trim,
*              disp_matrix,
*              matrxadd,
*              mmatrixmit,
*              matrxsub,
*              pause
* called by:  mmatrix
* author:    Susan K. Mashiko, Capt, USAF
*****

```

FILE: MATRIX.PAS

```
* Gary C. Tarczynski, Capt, USAF *  
* *  
*****
```

```
procedure matrx_manip( var first : cmdword; var second : cmdword;  
var result : cmdword; var mat_obj : cmdword );
```

```
var  
stor_loc : integer;  
mats : file of matrix;  
mat1,mat2 : matrix;  
mat3 : matrix;  
i : integer;  
  
begin  
trim( first );  
if first = 'MATA' then stor_loc := 0  
else  
if first = 'MATB' then stor_loc := 1  
else  
if first = 'MATIC' then stor_loc := 2  
else  
if first = 'MATD' then stor_loc := 3  
else  
if first = 'MATE' then stor_loc := 4;  
  
assign( mats, 'matrix.dat' );  
reset( mats );  
seek( mats, stor_loc );  
read( mats, mat1 );  
close( mats );  
  
trim( second );  
if second = 'MATA' then stor_loc := 0  
else  
if second = 'MATB' then stor_loc := 1  
else  
if second = 'MATIC' then stor_loc := 2  
else  
if second = 'MATD' then stor_loc := 3  
else  
if second = 'MATE' then stor_loc := 4;  
  
assign( mats, 'matrix.dat' );  
reset( mats );  
seek( mats, stor_loc );  
read( mats, mat2 );  
close( mats );
```

```

if mat_obj = 'ADD' then
  begin
    matrixadd( mat1, mat2, mat3, abort_command );
    if abort_command then exit;
  end
else
  if mat_obj = 'MATXMULT' then
    begin
      matrixmlt( mat1, mat2, mat3, abort_command );
      if abort_command then exit;
    end
  else
    if mat_obj = 'SUBTRACT' then
      begin
        matrixsub( mat1, mat2, mat3, abort_command );
        if abort_command then exit;
      end;
    trim( result );
    if result = 'MATA' then stor_loc := 0
    else
      if result = 'MATB' then stor_loc := 1
    else
      if result = 'MATC' then stor_loc := 2
    else
      if result = 'MATD' then stor_loc := 3
    else
      if result = 'MATE' then stor_loc := 4;
    assign( mats, 'matrix.dat' );
    reset( mats );
    seek( mats, stor_loc );
    write( mats, mat3 );
    close( mats );
    disp_matrix( result );
    pause;
  end;
(*****
* procedure: matrix_manip2
* version: 1.0
* date: 21 September 85
* description: This procedure will invert, transpose or multiply
* a matrix by a scalar
* global variables used: abort_command
*)

```

FILE: MATRIX.PAS

```

* passed variables:      number, first, result, mat_obj
* files read:          MATRIX.DAT
* files written:       MATRIX.DAT
* procedures called:   trim, disp_matrix,
*                   smatrixmlt, matrixtran,
*                   matrixinv, pause
* called by:          mmatrix
* author:             Susan K. Mashiko, Capt, USAF
*                   Gary C. Tarczynski, Capt, USAF
* *****

```

```

procedure matrix_manip2( var number : real; var first : cmdword;
var result : cmdword; var mat_obj : cmdword );

```

```

var
  stor_loc : integer;
  mats : file of matrix;
  mat1,mat2 : matrix;
  i : integer;
begin
  trim( first );
  if first = 'MATA' then stor_loc := 0
  else
  if first = 'MATB' then stor_loc := 1
  else
  if first = 'MATC' then stor_loc := 2
  else
  if first = 'MATD' then stor_loc := 3
  else
  if first = 'MATE' then stor_loc := 4;
  assign( mats, 'matrix.dat' );
  reset( mats );
  seek( mats, stor_loc );
  read( mats, mat1 );
  close( mats );
  if mat_obj = 'TRANPOSE' then
    matrixtran( mat1, mat2 )
  else
  if mat_obj = 'INVERSE' then
    begin
      matrixinv( mat1, mat2, abort_command );
      if abort_command then exit;
    end
  else

```

FILE: MATRIX.PAS

```

if mat_obj = 'SCLRMULT' then
  smatrixmit( number, mat1, mat2 );
trim( result );
if result = 'MATA' then stor_loc := 0
else
if result = 'MATB' then stor_loc := 1
else
if result = 'MATC' then stor_loc := 2
else
if result = 'MATD' then stor_loc := 3
else
if result = 'MATE' then stor_loc := 4;
assign( mats, 'matrix.dat' );
reset( mats );
seek( mats, stor_loc );
write( mats, mat2 );
close( mats );
disp_matrix( result );
pause;
end;
(*****
* procedure:  get_matrix_name
* version:    1.0
* date:       20 September 85
* description: This procedure will get the name of a matrix
*              from the screen
* global variables used:  abort_command, blanks
* global variables changed:  blanks
* global constants used:  as_assigned, crt_only
* passed variables:      mat_name, row, col, abort_command
* returned variables:    mat_name
* procedures called:      highlight,  nohighlight,
*                          gotoxy,    out_string,
*                          ucase,
*                          clear_msg, get_string,
*                          disp_msg,  pause
* called by:             mmatrix
* author:                Susan K. Mashiko, Capt, USAF
*                          Gary C. Tarczynski, Capt, USAF
*****
)

```

```
procedure get_matrx_name( var mat_name : msg_line; row : integer;
col : integer; abort_command : boolean );
```

```
label again:
begin
  again:
    gotoxy( row, col );
  highlight;
  out_string( copy( blanks, 1, 4), crt_only);
  nohighlight;
  gotoxy( 20, 0 );
  out_string( blanks, crt_only );
  gotoxy( 20, 10 );
  highlight;
  out_string( ' Your choice... ', as_assigned);
  nohighlight;
  get_strng( mat_name, abort_command, as_assigned, ' ', '-' );
  ucasc( mat_name );
  if (( mat_name = 'MATA' ) or ( mat_name = 'MATB' ) or
( mat_name = 'MATC' ) or ( mat_name = 'MATD' ) or
( mat_name = 'MATE' )) then
```

```
begin
  gotoxy( row,col);
  out_string( mat_name, as_assigned );
end
```

```
else
begin
  gotoxy( 18, 5 );
  disp_msg( 9 );
  pause;
  gotoxy( 18, 0 );
  clear_msg( 9 );
  goto again;
end;
```

```
end;
```

```
(*****
*
* procedure: mmatrix
* version: 1.0
* date: 19 Sep 85
* description: This procedure will decode the command string
* from procedure define and call the appropriate
* procedures.
* global variables used: cmdbuffer, abort_command
* global variables changed: abort_command
* global constants used: as_assigned
* passed variables: cmdbuffer, wordnumber
*
*****)
```

```
FILE: MATRIX.PAS
```



```

* procedures called:      trim, clear, disp_matrx, pause,
*                        gotoxy, disp_msg, out_string,
*                        get_matrx_name, matrx_manip1,
*                        matrx_manip2, get_r_num
*
* called by:  disp      Susan K. Mashiko, Capt, USAF
*            author:   Gary C. Tarczynski, Capt, USAF
*
* *****
procedure mmatrix( var cmdbuffer : buffer;
                  var wordnumber : integer );
var
  mat_obj   : cmdword;
  first    : cmdword;
  second   : cmdword;
  third    : cmdword;
  mat_name : msg_line;
  number   : real;
begin
  abort_command := false;
  mat_obj := cmdbuffer[ 3 ];
  trim( mat_obj );
  clear;
  (* this is the catch code for the display of a matrix *)
  if ( ( mat_obj = 'MATA' ) or ( mat_obj = 'MATB' ) or ( mat_obj = 'MATC' )
    or ( mat_obj = 'MATD' ) or ( mat_obj = 'MATE' ) ) then
    begin
      disp_matrx( mat_obj );
      pause;
    end
  else
    if ( ( mat_obj = 'ADD' ) or ( mat_obj = 'SUBTRACT' ) or
      ( mat_obj = 'MATXMULT' ) ) then
      begin
        clear;
        gotoxy( 3, 5 );
        disp_msg( 48 );
        gotoxy( 5, 20 );
        out_string( mat_obj, as_assigned );
        gotoxy( 15, 27 );
        out_string( mat_obj, as_assigned );
        gotoxy( 15, 48 );
        out_string( '=', as_assigned );
        get_matrx_name( mat_name, 15, 15, abort_command );
      end
    end
  end
end

```

FILE: MATRIX.PAS

```

first := mat_name;
get_matrix_name( mat_name, 15, 37, abort_command );
second := mat_name;
get_matrix_name( mat_name, 15, 52, abort_command );
third := mat_name;
matrix_manip1( first, second, third, mat_obj );
end
else
if ( ( mat_obj = 'TRANPOSE' ) or ( mat_obj = 'INVERSE' ) ) then
begin
clear;
gotoxy( 7, 0 );
disp_msg( 54 );
get_matrix_name( mat_name, 13, 35, abort_command );
first := mat_name;
get_matrix_name( mat_name, 15, 35, abort_command );
second := mat_name;
matrix_manip2( number, first, second, mat_obj );
end
else
if mat_obj = 'SCLRMULT' then
begin
clear;
gotoxy( 5, 0 );
disp_msg( 55 );
get_matrix_name( mat_name, 12, 44, abort_command );
first := mat_name;
get_num( number, 14, 44, abort_command );
get_matrix_name( mat_name, 16, 44, abort_command );
second := mat_name;
matrix_manip2( number, first, second, mat_obj );
end
else
if mat_obj = 'HELP' then
begin
clear;
disp_msg( 49 );
pause;
clear;
end;
end;

```

FILE: MATRIX.PAS

```

*****
**
** file: MATRXMAN.PAS
** procedures contained: matradd, mmatrxmlt, matrsub
**                    smatrxmlt, matrtran, matrxfnv
**
** version: 2.0
** date: 6 November 85
** description: This file contains the matrix manipulation
**              procedures.
** author: Susan K. Mashiko, Capt, USAF
**         Gary C. Tarczynski, Capt, USAF
**
**
**
*****

```

```

*****
**
** procedure: matradd
** version: 1.0
** date: 18 September 85
** description: The procedure adds the first two matrices
**              passed to it and places the result in the third.
**
** Global variables used: abort_command
** Global variables changed: abort_command
** Passed variables: amat, bmat, cmat, abort_command
** Returned variables: cmat, abort_command
** Procedures called: clear, disp_msg,
**                  pause, gotoxy
**
** Called by: matrsub, matr_manipl
** Author: Susan K. Mashiko, Capt, USAF
**         Gary C. Tarczynski, Capt, USAF
**
**
*****

```

```

procedure matradd( var amat : matrix; var bmat : matrix;
var cmat : matrix; var abort_command : boolean );
(* in this procedure the matrix amat and bmat are added *)
(* together to form the third matrix, cmat

```

```

var
  i : integer;
  j : integer;
begin
  abort_command := false;
  (* check to see if the two matrices may be added together *)
  if ( ( amat.num_rows <> bmat.num_rows ) or
      ( amat.num_cols <> bmat.num_cols ) ) then

```

FILE: MATRXMAN.PAS

```

begin
  clear;
  gotoxy( 7, 1 );
  disp_msg( 51 );
  pause;
  clear;
  abort_command := true;
  exit;
end

else
  (* the dimension of both matrices are the same add the elements *)
  begin
    for j := 1 to amat.num_cols do
      begin
        for i := 1 to amat.num_rows do
          cmat.element[ i, j ] := amat.element[ i, j ] +
            bmat.element[ i, j ];
        end;
      end;
    end;

    (* store the dimensions of the new matrix *)
    cmat.num_rows := amat.num_rows;
    cmat.num_cols := amat.num_cols;
  end;

  (* ***** *)
  procedure:   matrixsub
  *
  * version:   1.0
  * date:      20 September 85
  * description: This procedure subtracts the second matrix
  *              passed from the first and places the result into
  *              the third.
  *
  * global variables used:   abort_command
  *
  * passed variables changed: abort_command
  *
  * returned variables:      amat, bmat, cmat, abort_command
  *
  * procedures called:       matrixadd
  *
  * called by:               matrix_manip
  *
  * author:                  Susan K. Mashiko, Capt, USAF
  *                          Gary C. Tarczynski, Capt, USAF
  *
  * ***** *)
  procedure matrixsub( var amat : matrix; var bmat : matrix;
    var cmat : matrix; var abort_command : boolean );
  var

```

FILE: MATRXMAN.PAS

```

i : integer;
j : integer;
nmat : matrix;

begin
  abort_command := false;
  (* negate all of the elements of the second matrix *)
  (* then call the matradd routine *)
  for j := 1 to amat.num_cols do
    begin
      for i := 1 to amat.num_rows do
        nmat.element[ i, j ] := - bmat.element[ i, j ];
      end;
    end;
  nmat.num_rows := bmat.num_rows;
  nmat.num_cols := bmat.num_cols;
  matradd( amat, nmat, cmat, abort_command );
  if abort_command then exit;
end;

(*****
* procedure: mmatrxmlt
* version: 1.0
* date: 22 September 85
* description: This procedure multiplies the first two matrices
* passed to it and places the result in the
* third.
* global variables used: abort_command
* global variables changed: abort_command
* passed variables: amat, bmat, cmat, abort_command
* returned variables:
* cmatrix, abort_command
* procedures called:
* clear, gotoxy, pause,
* disp_msg
* called by: Susan K. Mashiko, Capt, USAF
* author: Gary C. Tarczynski, Capt, USAF
* *****)

procedure mmatrxmlt( var amat : matrix; var bmat : matrix;
var cmat : matrix; var abort_command : boolean );

var
  i : integer;
  j : integer;
  l : integer;

begin
  abort_command := false;

```

FILE: MATRXMAN.PAS

```

if amat.num_cols <> bmat.num_rows then
begin
clear;
gotoxy( 7, 0 );
disp_msg( 52 );
pause;
clear;
abort_command := true;
exit;
end
else
begin
for j := 1 to bmat.num_cols do
begin
for i := 1 to amat.num_rows do
begin
cmat.element[ i, j ] := 0.0;
for l := 1 to amat.num_cols do
cmat.element[ i, j ] := cmat.element[ i, j ] +
amat.element[ i, l ] * bmat.element[ l, j ];
end;
end;
end;
end;
cmat.num_rows := amat.num_rows;
cmat.num_cols := bmat.num_cols;
end;
end;

```

```

(*****
* procedure: smatrxmlt
* version: 1.0
* date: 20 September 85
* description: This procedure multiplies a matrix by a scalar
* and places the result in the in the second matrix
* passed variables: number, amat, bmat
* returned variables: bmat
* called by: matrix_manip2
* author: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
*****
)

```

```

procedure smatrxmlt( var number : real; var amat : matrix;
var bmat : matrix );

```

```

var
i : integer;
j : integer;

```

FILE: MATRXMAN.PAS

```

begin
  for j := 1 to amat.num_cols do
    begin
      for i := 1 to amat.num_rows do
        bmat.element[ i, j ] := amat.element[ i, j ] * number;
      end;
    end;
  bmat.num_rows := amat.num_rows;
  bmat.num_cols := amat.num_cols;
end;

(*****
*
* procedure:   matrixtran
* version:    1.0
* date:      20 September 85
* description: This procedure transposes a matrix and places the
*              result in the second matrix
* passed variables:   amat, bmat
* returned variables: bmat
* called by:         Susan K. Mashiko, Capt, USAF
*                   Gary C. Tarczynski, Capt, USAF
*
* *****)

procedure matrixtran( var amat : matrix; var bmat : matrix );
var
  i : integer;
  j : integer;
begin
  for j := 1 to amat.num_cols do
    begin
      for i := 1 to amat.num_rows do
        bmat.element[ j, i ] := amat.element[ i, j ];
      end;
    end;
  bmat.num_rows := amat.num_cols;
  bmat.num_cols := amat.num_rows;
end;

(*****
*
* procedure:   matrixinv
* version:    2.0
* date:      6 November 85
* description: This procedure invert the first matrix and store
*              the result in the second.
* global variables used:  abort_command
*
* *****)

```

FILE: MATRIXMAN.PAS

```

* global variables changed: abort_command
* passed variables: amat, bmat, abort_command
* returned variables: amat, bmat, abort_command
* procedures called: clear, gotoxy, disp_msg, pause
* called by: matrx_manip2
* author: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
* modification: replaced the algorithm for inverting a matrix
* modified by: author
* *****

```

```

procedure matrxinv( var amat : matrix; var bmat : matrix;
var abort_command : boolean );

```

```

label single;

```

```

var
dimension : integer;
i, j      : integer;
I1, I2, I3 : integer;
J1        : integer;
k         : array[ 1..50 ] of integer;
p         : array[ 1..50 ] of real;
PE, TPE  : real;

```

```

begin
abort_command := false;

```

```

(* set up primary counter for inversion *)
(* NN in fortran code *)
dimension := amat.num_rows;

```

```

(* test the matrix for singularity *)
for i := 1 to dimension do
begin

```

```

if amat.element( i, i ) = 0 then
begin

```

```

single;
clear;
gotoxy( 7, 1 );
disp_msg( 56 );
pause;
clear;
abort_command := true;
exit;
end;
end;

```



```

(* the matrix must be square for inversion *)
(* if it is not square display error message *)
if amat.num_rows <> amat.num_cols then
begin
  clear;
  gotoxy( 7, 1 );
  disp_msg( 53 );
  pause;
  clear;
  abort_command := true;
  exit;
end
else
begin
  for i := 1 to dimension do
  begin
    k[ i ] := 1;
    for j := 1 to dimension do
      bmat.element[ i, j ] := amat.element[ i, j ];
    end;
  end;
  for i := 1 to dimension do
  begin
    I2 := dimension - i + 1;
    PE := 0.0;
    for I1 := 1 to I2 do
    begin
      TPE := bmat.element[ I1, 1 ];
      if ( abs(PE) - abs(TPE) ) <= 0 then
      begin
        PE := TPE;
        IP := I1;
      end;
    end;
  end;
  if PE <> 0 then
  begin
    for j := 2 to dimension do
      p[ j - 1 ] := bmat.element[ IP, j ]/PE;
    p[ dimension ] := 1.0/PE;
    IP := k[ IP ];
    I2 := 0;
    for j := 1 to dimension do
    begin
      I1 := j - I2;
      k[ I1 ] := k[ j ];
      if ( k[ j ] - IP ) = 0 then
        I2 := 1
      else

```

FILE: MATRXMAN.PAS

```

begin
  TPE := -bmat.element[ j, 1 ];
  for J1 := 2 to dimension do
    bmat.element[ 1, J1 - 1 ] :=
      bmat.element[ j, J1 ] + TPE * p[ J1 - 1 ];
    bmat.element[ 1, dimension ] := TPE * p[ dimension ];
  end;
end;

for j := 1 to dimension do
  bmat.element[ dimension, j ] := p[ j ];
end
else
  goto single;
k[ dimension ] := IP;
end;

for i := 1 to dimension do
  begin
  for j := 1 to dimension do
    begin
      I1 := k[ j ];
      p[ I1 ] := bmat.element[ i, j ];
    end;
    for j := 1 to dimension do
      bmat.element[ i, j ] := p[ j ];
    end;
  end;
  bmat.num_rows := dimension;
  bmat.num_cols := dimension;
end;

```

FILE: MATRXMAN.PAS

```
(*****  
** file: MODIFY.PAS  
** procedures contained: chgmat, modify  
** version: 1.0  
** date: 22 September 85  
** description: This file contains the procedures that handle  
** the logic to modify or change polynomials and  
** matrices.  
** authors: Susan K. Mashiko, Capt, USAF  
** Gary C. Tarczynski, Capt, USAF  
**  
*****)  
(*****  
**  
*****)
```

```
(*****  
** procedure: chgmat  
** version: 1.0  
** date: 22 September 1985  
** description: This procedure will display the requested matrix  
** on the screen and ask the user which row and col  
** location should be modified and will store the  
** result in the original location.  
** global variables used: cmbuffer, abort_command  
** global constants used: as_assigned, crt_only  
** passed variables: cmbuffer, wordnumber  
** files read: MATRIX.DAT  
** files written:  
** procedures called:  
** trim, gotoxy, out_string,  
** disp_msg, out_real,  
** clear_msg, pause,  
** get_r_num, get_string,  
** ucase, disp_matrix,  
** get_int, make_pretty_small_matrix,  
** make_pretty_large_matrix_one,  
** make_pretty_large_matrix_two  
** called by: modify  
** authors: Susan K. Mashiko, Capt, USAF  
** Gary C. Tarczynski, Capt, USAF  
**  
*****)  
overlay procedure chgmat( var cmbuffer : buffer; wordnumber : integer );
```

```
label second_page.  
again.
```

```
FILE: MODIFY.PAS
```

```

repeat_again,
first_page_only;

var
stor_loc      : integer;
mats          : file of matrix;
col_element  : integer;
i             : integer;
j             : integer;
number       : real;
mat          : matrix;
num_row     : integer;
num_col     : integer;
chg_row     : integer;
chg_col     : integer;
row         : integer;
col         : integer;
choice      : cmdword;
select      : msg_line;
result      : integer;

begin
clear;
choice := cdbuffer[ 3 ];
trim( choice );
(* get the selected matrix from memory *)
if choice = 'MATA' then stor_loc := 0
else
if choice = 'MATB' then stor_loc := 1
else
if choice = 'MATC' then stor_loc := 2
else
if choice = 'MATD' then stor_loc := 3
else
if choice = 'MATE' then stor_loc := 4;

assign( mats, 'matrix.dat' );
reset( mats );
seek( mats, stor_loc );
read( mats, mat );
close( mats );

(* put the title on the screen *)
clear;
gotoxy( 1, 32 );
disp_msg( 50 );
gotoxy( 2, 37 );
out_string( choice, as_assigned );
num_col := mat.num_cols;

```

FILE: MODIFY.PAS

```

num_row := mat.num_rows;
row := 5;
col := 10;

(* if the matrix is small (i.e. 5 columns or less) display *)
(* and change the entry
if num_col <= 5 then
begin
make_pretty_small_matrix( num_row, num_col );
for j := 0 to ( num_col - 1 ) do
begin
col_element := j + 1;
for i := 1 to num_row do
begin
gotoxy( ( row + i ), ( col + ( j * 13 ) ) );
out_real( mat.element[ i, col_element ], 12, as_assigned );
end;
end;
end;

(* request the location of the change *)
gotoxy( 18, 0 );
disp_msg( 57 );

(* get the row of the change *)
repeat
begin
gotoxy( 18, 46 );
out_string( ' ', crt_only );
gotoxy( 18, 46 );
get_int( chg_row, abort_command );
if abort_command then exit;
if ( ( chg_row > mat.num_rows ) or ( chg_row < 1 ) ) then
begin
gotoxy( 21, 5 );
disp_msg( 9 );
pause;
gotoxy( 21, 5 );
clear_msg( 9 );
end;
end;
until ( ( chg_row > 0 ) and ( chg_row <= mat.num_rows ) );

(* get the column of the change *)
repeat
begin
gotoxy( 19, 46 );
out_string( ' ', crt_only );
gotoxy( 19, 46 );
get_int( chg_col, abort_command );
if abort_command then exit;

```

FILE: MODIFY.PAS

```

if ( ( chg_col > mat.num_cols ) or ( chg_col < 1 ) ) then
begin
  gotoxy( 21, 5 );
  disp_msg( 9 );
  pause;
  gotoxy( 21, 5 );
  clear_msg( 9 );
end;
end;
until ( ( chg_col > 0 ) and ( chg_col <= mat.num_cols ) );
(* ask the user for the new number *)
get_f_num( number, ( row + chg_row ), ( col + (( chg_col - 1 ) * 13 ) ),
  abort_command );
(* update the matrix with new entry *)
mat.element[ chg_row, chg_col ] := number;
end
else
(* this is the code for the modification of a large matrix *)
begin
(* display the first page of a large matrix *)
make_pretty_large_matrix_one( num_row, num_col );
for j := 0 to 4 do
begin
  col_element := j + 1;
  for i := 1 to num_row do
  begin
    gotoxy( ( row + i ), ( col + ( j * 13 ) ) );
    out_real( mat.element[ i, col_element ], 12, as_assigned );
  end;
end;
(* request the location of the change or to proceed to next page *)
gotoxy( 17, 0 );
disp_msg( 58 );
repeat again:
begin
  gotoxy( 18, 46 );
  out_string( ' ', crt_only );
  gotoxy( 18, 46 );
  get_string( select, abort_command, as_assigned, ' ', '~' );
  if abort_command then exit;
  ucasc( select );
  if select = 'NEXT' then
    goto second_page
end;

```

FILE: MODIFY.PAS

AD-A164 844

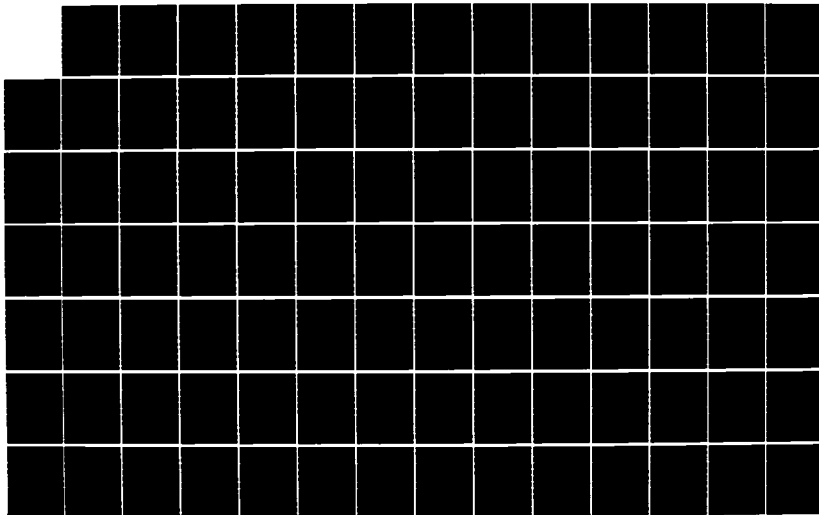
DEVELOPMENT OF A COMPUTER AIDED DESIGN PACKAGE FOR
CONTROL SYSTEM DESIGN A. (U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..
S K NASHIKO ET AL. DEC 85

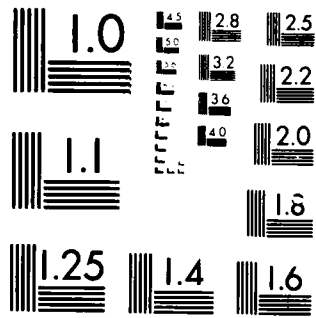
9/5

UNCLASSIFIED

F/8 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A


```

else
  val( select, chg_row, result );
  if result <> 0 then goto again;
  if ( ( chg_row > mat.num_rows ) or ( chg_row < 1 ) ) then
    begin
      again:
        gotoxy( 21, 5 );
        disp_msg( 9 );
        pause;
        gotoxy( 21, 5 );
        clear_msg( 9 );
        goto repeat_again;
      end;
    end;

  (* get the column of the change *)
  repeat
    begin
      gotoxy( 19, 46 );
      out_string( ' ', crt_only );
      gotoxy( 19, 46 );
      get_int( chg_col, abort_command );
      if abort_command then exit;
      if ( ( chg_col > 5 ) or ( chg_col < 1 ) ) then
        begin
          gotoxy( 21, 5 );
          disp_msg( 9 );
          pause;
          gotoxy( 21, 5 );
          clear_msg( 9 );
          end;
        end;
      until ( ( chg_col > 0 ) and ( chg_col <= 5 ) );
      (* ask the user for the new number *)
      get_r_num( number, ( row + chg_row ), ( col + ( chg_col - 1 ) * 13 ) );
      abort_command;
      (* update the matrix with new entry *)
      mat.element[ chg_row, chg_col ] := number;
      goto first_page_only;
    end;
  second_page:
  clear;
  (* display the second page of a large matrix *)
  make_pretty_large_matrix_two( num_row, num_col );
  for j := 0 to num_col - 6 do
    begin
      col_element := j + 6;

```

FILE: MODIFY.PAS

```

for i := 1 to num_row do
  begin
    gotoxy( ( row + i ), ( col + ( i * 13 ) ) );
    out_real( mat_element[ i, col_element ], 12, as_assigned );
  end;
end;

(* request the location of the change *)
gotoxy( 18, 0 );
disp_msg( 57 );

(* get the row of the change *)
repeat
  begin
    gotoxy( 18, 46 );
    out_string( ' ', crt_only );
    gotoxy( 18, 46 );
    get_int( chg_row, abort_command );
    if abort_command then exit;
    if ( ( chg_row > mat.num_rows ) or ( chg_row < 1 ) ) then
      begin
        gotoxy( 21, 5 );
        disp_msg( 9 );
        pause;
        gotoxy( 21, 5 );
        clear_msg( 9 );
      end;
    end;
  until ( ( chg_row > 0 ) and ( chg_row <= mat.num_rows ) );

(* get the column of the change *)
repeat
  begin
    gotoxy( 19, 46 );
    out_string( ' ', crt_only );
    gotoxy( 19, 46 );
    get_int( chg_col, abort_command );
    if abort_command then exit;
    if ( ( chg_col > mat.num_cols ) or ( chg_col < 6 ) ) then
      begin
        gotoxy( 21, 5 );
        disp_msg( 9 );
        pause;
        gotoxy( 21, 5 );
        clear_msg( 9 );
      end;
    end;
  until ( ( chg_col > 5 ) and ( chg_col <= mat.num_cols ) );

```

```

(* ask the user for the new number *)
get_r_num( number, ( row + chg_row ), ( col + ( ( chg_col - 6 ) * 13 ) ),
          abort_command );
(* update the matrix with new entry *)
mat.element[ chg_row, chg_col ] := number;
end;

```

```

first_page_only:

```

```

assign( mats, 'matrix.dat' );
reset( mats );
seek( mats, stor_loc );
write( mats, mat );
close( mats );

```

```

(* display the modified matrix to the user *)
disp_matrix( choice );
pause;

```

```

end;

```

```

(*****
*
* procedure: modify
* version: 1.0
* date: 22 September 85
* description: This procedure contains the logic to decide which
* modification procedure should be called and calls it
* global variables called: cmdbuffer
* passed variables: cmdbuffer, wordnumber
* procedures called: inroot, delroot, chgmat, clear,
* trim, disp_msg, pause
* called by: select
* authors: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
*
*****
)

```

```

procedure modify( var cmdbuffer : buffer;
                 var wordnumber : integer);

```

```

var
  mod_obj : cmdword;

```

```

begin
  mod_obj := cmdbuffer[ 2 ];
  trim( mod_obj );
  clear;

```

```

FILE: MODIFY.PAS

```

```
if mod_obj = 'ADDR00T' then
  inroot( cmdbuffer, number_of_commands )
else
  if mod_obj = 'DELROOT' then
    delroot( cmdbuffer, number_of_commands )
  else
    if mod_obj = 'CHANGE' then
      chgmat( cmdbuffer, number_of_commands )
    else
      if mod_obj = 'HELP' then
        begin
          clear;
          disp_msg( 25 );
          pause;
          clear;
        end;
      end;
    end;
  end;
end;
```

FILE: MODIFY.PAS

```
(.....
**   MSDWCONS.PAS
**   version: 4.0
**   date: 19 September 85
**   description: This file contains the constant
**               definitions for the MICROSDW
**               routines.
**   author: Paul A. Moore, Capt, USAF
**   modified by: Susan K. Mashiko, Capt, USAF
**               Gary C. Tarczynski, Capt, USAF
**   mod description: Changed num_words from 75 to 80
**   mod date: 8 Aug 85
**   mod description: Changed num_ptrs from 200 to 250,
**               and changed num_words from 80 to 100.
**   mod date: 4 September 1985
**   mod description: Changed num_ptrs from 250 to 350.
**   mod date: 19 September 1985
**   (.....)

const
  term_length = 95; ( length of array for terminal control data )
  printer_length = 50; ( length of array for printer control data )
  ff = 12; ( decimal for form feed char )
  wordlength = 9; ( length of word in storage )
  num_param_group = 1;
  num_bools = 10;
  num_ints = 10;
  num_reals = 10;
  num_strings = 10;
  num_ptr_recs = 3;
  num_ptrs = 350;
  num_words = 100;
  num_msg_dir = 70;
  num_msg_line = 250;
  screen_width = 79;

  ENDCODE = 9999;
  DONEWORD = '$$$$$$$$$$';
```

FILE: MSDWCONS.PAS

```

*****
**
**      file:      MSDWTYPE.PAS
**      version:   1.2
**      date:      28 November 84
**      description: This file contains the
**                  type definitions for the MICROSDW
**                  routines.
**      author:    Paul A. Moore, Capt. USAF
**
**
*****

```

```

type
  ptr_recs   = array[ 1..num_ptr_recs ] of integer;
  paramstring = string[14];

  msg      = record
    loc_rec : integer;
    length  : byte;
  end;

  dict_buffer = record
    ptrs : array[ 1..num_ptrs ] of ptr_recs;
    words : array[ 0..num_words ] of string[ wordlength ];
    abbrev : array[ 0..num_words ] of integer;
  end;

  param_group = record
    bools : array[ 1..num_bools ] of boolean;
    ints  : array[ 1..num_ints ] of integer;
    reals : array[ 1..num_reals ] of real;
    strings : array[ 1..num_strings ] of paramstring;
  end;

  msg_dat   = array[1..num_msg_line ] of string[screen_width];
  data      = record
    param : array[ 1..num_param_group ] of param_group;
    term  : array[ 1..term_length ] of byte;
    printr : array[ 1..printer_length ] of byte;
    msg_dir : array[ 1..num_msg_dir ] of msg;
    decode_dict : dict_buffer;
  end;

  wordtype = string[wordlength];

```

```

*****
**
** file: MSG.PAS disp_line,
** procedures contained: clear_msg,
** disp_msg
**
** version: 1.5
** date: 23 August 85
** description: This file contains the procedures to
** display and clear messages.
** author: vincent m. parisi ii, capt., usaf
** Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt. USAF
**
*****

```

```

*****
**
** procedure: disp_line
** version: 1.2
** date: 18 oct 83
** description: This procedure reads one line of text from
** the file HELP.SYS and displays it on the
** assigned device.
**
** global variables used: msg_txt
** global constants used: as_assigned, screen_width
** passed variables: rec_num
** files read: HELP.SYS
** procedure called: out_string
** called by: disp_msg
** author: vincent m. parisi ii, capt., usaf
**
*****

```

```

procedure disp_line( rec_num : integer );
var print_line : string[screen_width];
begin
seek( msg_txt, rec_num );
read( msg_txt, print_line );
out_string( print_line, as_assigned );
writeln;
end;

```

```

*****

```

```

* procedure: clear_msg
* version: 1.3
* date: 18 oct 83
* description: This procedure clears the message indicated
* by msg_num, from the screen. It is the
* programmer's responsibility to position the
* cursor prior to calling this routine. The
* cursor should be placed at the beginning of
* the line where you wish the message erased.
*
* global variables used: msg_dir, blanks
* global constants used: crt_only
* passed variables: msg_num
* procedure called: clear, out_string
* called by: disp_msg
* author: vincent m. parisi ii, capt., usaf
*
*****

```

```

procedure clear_msg( msg_num : integer );

```

```

var
  i      : integer;
  length : integer;

```

```

begin

```

```

  length := msg_dir[ msg_num ].length;

```

```

  if length > 23 then

```

```

    clear

```

```

  else

```

```

    for i := 1 to length do

```

```

      begin

```

```

        out_string( blanks, crt_only );

```

```

        writein;

```

```

      end;

```

```

end;

```

```

(*****
*
* procedure: disp_msg
* version: 3.1
* date: 23 Aug 85
* description: This procedure displays the message pointed
* to by the parameter passed in, msg_num.
* The message is displayed at the current
* cursor position. If the message length is
* longer than 23 lines, the display stops
* after showing 22 lines and waits for the
* user to indicate 'continue' with a <CR>.
* If a '$' is received, the procedure is
*
*****

```

```

FILE: MSG.PAS

```



```

* * * * * exited and returns to the calling routine.
* * * * *
* * * * * global variables used:      msg_dir
* * * * * passed variables:         msg_num
* * * * * procedures called:        disp_line, disp_msg,
* * * * *                             gotoxy, clear, clear_msg
* * * * *
* * * * * called by:               help, proces_error
* * * * * author:                  vincent m. parisi ii, capt., usaf
* * * * * modified by:             Susan K. Mashiko, Capt, USAF
* * * * *                             Gary C. Tarczynski, Capt, USAF
* * * * *
* * * * * mod description: Original code used one method for the
* * * * * display of the first and second pages of the
* * * * * message and a second method for the re-
* * * * * maining pages. This mod changed the code so
* * * * * that the same method is used for the display
* * * * * of all pages.
* * * * * mod date:                 23 Aug 85
* * * * *
* * * * * *****

```

```

procedure disp_msg( msg_num : integer );

```

```

var
  i          : integer;
  length    : integer;
  rec_num   : integer;
  remain_lines : integer;
  resp      : char;

```

```

begin

```

```

  length := msg_dir[ msg_num ].length;
  rec_num := msg_dir[ msg_num ].loc_rec - 1;

```

```

  if length < 23 then
    for i := 0 to ( length - 1 ) do
      disp_line(( rec_num + i ))
  else

```

```

    begin
      remain_lines := length;
      while remain_lines > 21 do
        begin

```

```

          clear;
          gotoxy( 0, 0 );
          for i := 0 to 20 do
            disp_line( (rec_num + i) );
          remain_lines := remain_lines - 21;
          rec_num := rec_num + 20;
          gotoxy( 22, 0 );
          disp_msg( 13 );
          read( resp );

```

```

FILE: MSG.PAS

```

```
gotoxy( 22, 0 );
clear_msg( 13 );
if resp = '$' then exit;
gotoxy( 22, 0 );
end;
clear;
for i := 0 to remain_lines do
  disp_line( (rec_num + i) );
end;
end;
end; (* records on disk begin at 0 *)
```

FILE: MSG.PAS

```

*****
**
** file: OUTPUT.PAS
** procedure contained: out_string
** version: 1.0
** date: 01 august 1983
** description: This module contains the procedure that
** handles all output to the user.
** author: vincent m. parisi ii, capt., usaf
**
*****
(*****
**
** procedure: out_string
** version: 1.0
** date: 01 august 1983
** description: This procedure handles all string
** output for the program. Whenever
** system output is required, this
** module is called. The output is direc-
** ted to the appropriate device.
** The devices that can accept output are:
** crt-- generally all interaction
** printer-- for hard copy either dyn-
** amically or selectively.
** transaction file--file which contains
** all user interactions for session
** tracing.
** temporary file--so user can review
** work just accomplished.
** Output can be directed specifically
** to the crt only, the printer only, both
** or as indicated by the boolean switches
** discussed below.
**
** global variables used: crt, temp, trans, printer, list_dev,
** trans_file, temp_file
**
** passed variables: ostring, dest
** files written: PRINTER.OUT, TRANSACTION, TEMP.OUT
** called by: many
** author: vincent m. parisi ii, capt., usaf
**
*****
)

```

```

procedure out_string( ostring : msg_line; dest : char );
begin

```

```

FILE: OUTPUT.PAS

```

```
case dest of
'C', 'c' :
'P', 'p' :
'B', 'b' :
    write( ostring );
    writeln( list_dev, ostring );
begin
    write( ostring );
    writeln( list_dev, ostring );
end;
'A', 'a' :
begin
    if crt then write( ostring );
    if trans then writeln( trans_file, ostring );
    if printer then writeln( list_dev, ostring );
    if temp then writeln( temp_file, ostring );
end;
end;
end;
```

FILE: OUTPUT.PAS

```
(*****
**
**          file:          PAUSE.PAS
** procedure contained: pause
** version:             1.3
** date:               29 October 1984
** description:       This module contains the procedure
**                   that waits for user response to continue
**                   anytime there is a stop in the program.
**                   vincent m. parisi ii, capt., usaf
**
** *****
)
```

```
(*****
*
* procedure:          pause
* version:           1.1
* date:             29 October 1984
* description:       This procedure waits for user response
*                   to continue anytime there is a stop in
*                   the program.  If the user has selected
*                   the status line on, then it is displayed,
*                   otherwise it is not.
* global variables used:  blanks, status_line, stat_on
* global constants used:  screen_width, crt_only,
*                   stat_line_width
* procedures called:  gotoxy, highlight, nohighlight,
*                   out_string
* called by:         many
* author:            vincent m. parisi ii, capt., usaf
* modifier:          Paul A. Moore, Capt, USAF
*
* *****
)
```

```
procedure pause;
var resp : char;

begin
gotoxy( 22, 0);
out_string(blanks, crt_only);
gotoxy( 22, 20);
highlight;
out_string(' >>> Press <CR> Key to continue... <<<< ', crt_only );
nohighlight;
read( resp );
if stat_on then
```

FILE: PAUSE.PAS

```
begin
  gotoxy(22,0);
  out_string(status_line, crt_only);
end
else
  out_string( blanks, crt_only );
end;
```

FILE: PAUSE.PAS

```

*****
** file: POLYMAN.PAS
** procedures contained: polyadd, polylmt, polysub
**                      spolylmt
** version: 4.0
** date: 7 November 85
** description: This file contains the polynomial manipulation
**              procedures.
** author: Susan K. Mashiko, Capt, USAF
**         Gary C. Tarczynski, Capt, USAF
**
*****

```

```

*****
** procedure: polyadd
** version: 2.0
** date: 18 September 85
** description: The procedure adds the first two polynomials
**              passed to it and places the result in the third.
** passed variables: apoly, bpoly, cpoly
** returned variables: apoly, bpoly, cpoly
** procedures called: roots
** called by: polysub, polymanip
** author: Susan K. Mashiko, Capt, USAF
**         Gary C. Tarczynski, Capt, USAF
** mod description: Modified the code to correct coefficient
**                  handling.
** modified by: author
** mod date: 18 Sep 85
**
*****

```

```

procedure polyadd( var apoly : polynomial; var bpoly : polynomial;
                  var cpoly : polynomial );
(* in this procedure the polynomials apoly and bpoly are added *)
(* together to form the third polynomial, cpoly *)

```

```

label
50, 55;

```

```

var
n : integer;
nc : integer;
ncc : integer;
i : integer;

```

FILE: POLYMAN.PAS

```

nn : integer;
begin
  (* find the degree of cpoly *)
  if apoly.polydeg > bpoly.polydeg then
    cpoly.polydeg := apoly.polydeg;
  if apoly.polydeg <= bpoly.polydeg then
    cpoly.polydeg := bpoly.polydeg;
  (* insure the constant/gain is one *)
  if apoly.coefficient <> 1 then
    for i := 1 to ( apoly.polydeg + 1 ) do
      apoly.poly[ i ] := apoly.poly[ i ] *
        apoly.coefficient;
    apoly.coefficient := 1;
  (* insure the constant/gain is one *)
  if bpoly.coefficient <> 1 then
    for i := 1 to ( bpoly.polydeg + 1 ) do
      bpoly.poly[ i ] := bpoly.poly[ i ] *
        bpoly.coefficient;
    bpoly.coefficient := 1;
  (* establish counters for do loops *)
  ncc := ( cpoly.polydeg + 1 );
  nn := abs( apoly.polydeg - bpoly.polydeg );
  if ( apoly.polydeg - bpoly.polydeg ) < 0 then
    begin
      for i := 1 to n do
        cpoly.poly[ i ] := bpoly.poly[ i ];
      nn := n + 1;
      for i := nn to ncc do
        cpoly.poly[ i ] := bpoly.poly[ i ] +
          apoly.poly[ ( i - n ) ];
      goto 50
    end
  else
    if ( apoly.polydeg - bpoly.polydeg ) = 0 then
      begin
        for i := 1 to ncc do
          cpoly.poly[ i ] := apoly.poly[ i ] + bpoly.poly[ i ];
        goto 50
      end
    else
      if ( apoly.polydeg - bpoly.polydeg ) > 0 then
        begin
          for i := 1 to n do

```

FILE: POLYMAN.PAS


```

      cpoly.polypoly[ i ] := apoly.polypoly[ i ];
      nn := n + 1;
      for i := nn to ncc do
        cpoly.polypoly[ i ] := apoly.polypoly[ i ] +
          bpoly.polypoly[ ( i - n ) ];
      end;
50: if cpoly.polypoly[ 1 ] <> 0 then goto 55;
   nc := nc - 1;
   if nc = 0 then goto 55;
   goto 50;
55: cpoly.coefficient := cpoly.polypoly[ 1 ];

(* standardization code for poly storage *)
if cpoly.polypoly[ 1 ] <> 1 then
  for i := 2 to (cpoly.polydeg + 1) do
    cpoly.polypoly[ i ] := cpoly.polypoly[ i ] /
      cpoly.polypoly[ 1 ];
  end;
  roots( cpoly );
end;

```

```

(*****
*
* procedure: polysub
* version: 2.0
* date: 7 November 85
* description: This procedure subtracts the second polynomial
* passed from the first and places the result into
* the third.
* passed variables: apoly, bpoly, cpoly
* procedures called: polyadd
* called by: polymanip
* author: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
* mod description: added error detection code
* mod date: 7 Nov 85
*
*****
)

```

```

procedure polysub( var apoly : polynomial; var bpoly : polynomial;
var cpoly : polynomial; var abort_command : boolean );

```

```

label continue;

```

```

var
  nbb : integer;
  i : integer;

```

FILE: POLYMAN.PAS

```

nbpoly : polynomial;

begin
nbb := bpoly.polydeg + 1;
nbpoly.polydeg := bpoly.polydeg;
nbpoly.coefficient := - bpoly.coefficient;

for i := 1 to nbb do
  nbpoly.polypoly[ i ] := bpoly.polypoly[ i ];
if ( apoly.coefficient = bpoly.coefficient ) and ( apoly.polydeg =
  bpoly.polydeg ) then
begin
  for i := 1 to apoly.polydeg + 1 do
    if apoly.polypoly[ i ] <> bpoly.polypoly[ i ] then
      goto continue
    else
      begin
        clear;
        gotoxy( 10, 5 );
        writeln( 'ERROR: you may not subtract equal polynomials from' );
        gotoxy( 11, 5 );
        writeln(
          '         pause;
          abort_command := true;
          exit;
        );
      end;
end;

continue;
polyadd( apoly, nbpoly, cpoly );

end;

(*****
* procedure: polymlt
* version: 1.0
* date: 4 September 85
* description: This procedure multiplies the first two poly-
*             nomials passed to it and places the result in the
*             third.
* passed variables: apoly, bpoly, cpoly
* returned variables: apoly, bpoly, cpoly
* procedures used: clear, gotoxy,
*                 highlight, nonhighlight,
*                 roots
* called by: polymanlp
*****

```

FILE: POLYMAN.PAS

```

* author: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
*
*****

```

```

procedure polymlt( var apoly : polynomial; var bpoly : polynomial;
var cpoly : polynomial );

```

```

var
i,j : integer;
naa : integer;
nbb : integer;
begin
(* insure the constant/gain is one *)
if apoly.coefficient <> 1 then
for i := 1 to ( apoly.polydeg + 1 ) do
apoly.polypoly[ i ] := apoly.polypoly[ i ] *
apoly.coefficient;
apoly.coefficient := 1;
(* insure the constant/gain is one *)
if bpoly.coefficient <> 1 then
for i := 1 to ( bpoly.polydeg + 1 ) do
bpoly.polypoly[ i ] := bpoly.polypoly[ i ] *
bpoly.coefficient;
bpoly.coefficient := 1;
(* check to see if the resulting polynomial will be too large *)
cpoly.polydeg := apoly.polydeg + bpoly.polydeg;
if cpoly.polydeg > 10 then
begin
clear;
gotoxy( 8, 0 );
highlight;
writeln( ' Degree of result greater than 10, option aborted.' );
nohighlight;
writeln( ' Due to the storage space limitations your resulting ');
writeln( ' polynomial is limited to 10 th order.' );
exit;
end
else
begin
for i := 1 to 11 do
cpoly.polypoly[ i ] := 0.0;
naa := apoly.polydeg + 1;
nbb := bpoly.polydeg + 1;
for i := 1 to naa do
for j := 1 to nbb do

```

FILE: POLYMAN.PAS

```

    cpoly.polypoly[ i + j - 1 ] := cpoly.polypoly[ i + j - 1 ] +
        apoly.polypoly[ i ] * bpoly.polypoly[ j ];
    cpoly.coefficient := apoly.coefficient * bpoly.coefficient;
end;

```

```

cpoly.coefficient := cpoly.polypoly[ 1 ];

```

```

(* standardization code for poly storage *)

```

```

if cpoly.polypoly[ 1 ] <> 1 then
    for i := 2 to (cpoly.polydeg + 1) do
        cpoly.polypoly[ i ] := cpoly.polypoly[ i ] /
            cpoly.polypoly[ 1 ];
    cpoly.polypoly[ 1 ] := 1;
end;

```

```

roots( cpoly );
end;

```

```

(*****
*
* procedure:  spolymlt
* version:   1.0
* date:      7 October 85
* description: This procedure multiplies the polynomial by the
* scalar (real) value and stores the result in the
* second polynomial
* passed variables:  apoly, bpoly, number
* returned variables:  apoly, bpoly
* procedures called:  roots
* called by:          polymanip, form
* author:            Susan K. Mashiko, Capt, USAF
*                  Gary C. Tarczynski, Capt, USAF
*
*****
*)

```

```

procedure spolymlt( var apoly : polynomial; var bpoly : polynomial;
    var number : real );

```

```

var
    counter : integer;
    i : integer;

```

```

begin
    counter := apoly.polydeg + 1;
    if apoly.polypoly[ 1 ] <> 1 then
        begin
            for i := 2 to ( apoly.polydeg + 1 ) do
                apoly.polypoly[ i ] := apoly.polypoly[ i ] / apoly.polypoly[ 1 ];
            apoly.polypoly[ 1 ] := 1;
            apoly.coefficient := apoly.coefficient * apoly.polypoly[ 1 ];
        end;
    end;

```

```

FILE: POLYMAN.PAS

```

```
end;  
bpoly.coefficient := apoly.coefficient * number;  
bpoly.polydeg := apoly.polydeg;  
for i := 1 to counter do  
  bpoly.polypoly[ i ] := apoly.polypoly[ i ];  
roots( bpoly );  
end;
```

FILE: POLYMAN.PAS

```

*****
** file: POLY.PAS
** procedures contained: disppoly, polymanip,
** ppoly, get_poly_name,
** polymanip2
** version: 3.0
** date: 8 October 85
** description: This file contains the procedures to display
** and manipulate polynomials.
** author: Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt, USAF
**
*****

```

```

*****
* procedure: disppoly
* version: 1.0
* date: 6 September5
* description: This procedure will display a polynomial from a
* record in TF&POL.S.DAT The user should place a pause
* in his/her code after the calling subroutine to
* keep the display on the screen.
* global constants used: as_assigned
* passed variables: choice
* files read: TF&POL.S.DAT
* procedures called: clear, gotoxy,
* out_string, disp_msg,
* make_pretty, out_real,
* trim
* called by: polymanip, ppoly, inroot, delroot
* author: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
*
*****

```

```

procedure disppoly( var choice : cmdword );

```

```

var
stor_loc : integer;
polys : file of polynomial;
i : integer;
row : integer;
number : real;
pol : polynomial;

```

FILE: POLY.PAS

```

begin
  trim( choice );
  if choice = 'POLVA' then stor_loc := 18
  else
  if choice = 'POLVB' then stor_loc := 19
  else
  if choice = 'POLVC' then stor_loc := 20
  else
  if choice = 'POLVD' then stor_loc := 21
  else
  if choice = 'POLVE' then stor_loc := 22
  else
  if choice = 'ONPOLY' then stor_loc := 0
  else
  if choice = 'ODPOLY' then stor_loc := 1
  else
  if choice = 'CNPOLY' then stor_loc := 2
  else
  if choice = 'COPOLY' then stor_loc := 3
  else
  if choice = 'GNPOLY' then stor_loc := 4
  else
  if choice = 'GDPOLY' then stor_loc := 5
  else
  if choice = 'HNPOLY' then stor_loc := 6
  else
  if choice = 'HDPOLY' then stor_loc := 7;

  assign( polys, 'tf&polys.dat' );
  reset( polys );
  seek( polys, stor_loc);
  read( polys, pol );

  (* put the title on the screen *)
  clear;
  gotoxy( 0, 27 );
  disp_msg( 34 );
  gotoxy( 1, 35 );
  out_string( choice, as_assigned );
  gotoxy( 2, 34 );
  row := 3;

  (* draw the form on the screen *)
  make_pretty( row, pol.polydeg );
  i := 1;

  (* get the coefficient and display it *)
  number := pol.coefficient;
  gotoxy( ( row + 2 ), 19 );

```

FILE: POLY.PAS

```

out_real( number, 12, as_assigned );
gotoxy( row + 2, 57 );
out_real( number, 12, as_assigned );

(* display the factored form *)
while i <= pol.polydeg do
begin
gotoxy( ( row + 3 + i ), 43 );
out_real( pol.polyfact[ i ].realpart, 12, as_assigned );
gotoxy( ( row + 3 + i ), 59 );
out_real( pol.polyfact[ i ].imagpart, 12, as_assigned );
i := i + 1;
end;

(* now display the polynomial form *)
i := 1;
while i <= ( pol.polydeg + 1 ) do
begin
gotoxy( ( row + 3 + i ), 7 );
out_real( pol.poly[ i ], 12, as_assigned );
i := i + 1;
end;
end;

(*****
*
* procedure: polymanip
* version: 1.0
* date: 6 September 85
* description: This procedure will add, subtract, or multiply two
* polynomials.
* passed variables: first, second, result, poly_obj
* files read: TF&POLS.DAT
* files written: TF&POLS.DAT
* procedures called: trim, disppoly,
* polyadd, polymlt,
* polysub, pause
* called by: ppoly
* author: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
*****
)

procedure polmanip( var first : cmdword; var second : cmdword;
var result : cmdword; var poly_obj : cmdword );
var
stor_loc : integer;
poly5 : file of polynomial;

```

FILE: POLY.PAS


```

pol1,pol2 : polynomial;
pol3      : polynomial;
i         : integer;

begin
  trim( first );
  if first = 'POLVA' then stor_loc := 18
  else
  if first = 'POLVB' then stor_loc := 19
  else
  if first = 'POLYC' then stor_loc := 20
  else
  if first = 'POLYD' then stor_loc := 21
  else
  if first = 'POLYE' then stor_loc := 22
  else
  if first = 'UNPOLY' then stor_loc := 0
  else
  if first = 'ODPOLY' then stor_loc := 1
  else
  if first = 'CNPOLY' then stor_loc := 2
  else
  if first = 'CDPOLY' then stor_loc := 3
  else
  if first = 'UNPOLY' then stor_loc := 4
  else
  if first = 'GDPOLY' then stor_loc := 5
  else
  if first = 'HMPOLY' then stor_loc := 6
  else
  if first = 'HDPOLY' then stor_loc := 7;
  assign( polys, 'f&polys.dat' );
  reset( polys );
  seek( polys, stor_loc);
  read( polys, pol1 );
  close( polys );

  trim( second );
  if second = 'POLVA' then stor_loc := 18
  else
  if second = 'POLVB' then stor_loc := 19
  else
  if second = 'POLYC' then stor_loc := 20
  else
  if second = 'POLYD' then stor_loc := 21
  else
  if second = 'POLYE' then stor_loc := 22

```

FILE: POLY.PAS

```

else
  if second = 'ONPOLY' then stor_loc := 0
  else
  if second = 'ODPOLY' then stor_loc := 1
  else
  if second = 'CNPOLY' then stor_loc := 2
  else
  if second = 'CDPOLY' then stor_loc := 3
  else
  if second = 'GNPOLY' then stor_loc := 4
  else
  if second = 'GDPOLY' then stor_loc := 5
  else
  if second = 'HNPOLY' then stor_loc := 6
  else
  if second = 'HDPOLY' then stor_loc := 7;

assign( polys, 'f&pol's:dat' );
reset( polys );
seek( polys, stor_loc);
read( polys, pol2 );
close( polys );

if poly_obj = 'ADD' then
  polyadd( pol1, pol2, pol3)
else
  if poly_obj = 'POLYMLT' then
    polymt( pol1, pol2, pol3 )
  else
    if poly_obj = 'SUBTRACT' then
      begin
        polysub( pol1, pol2, pol3, abort_command );
        if abort_command then exit;
      end;

  trim( result );

  if result = 'POLVA' then stor_loc := 18
  else
  if result = 'POLVB' then stor_loc := 19
  else
  if result = 'POLVC' then stor_loc := 20
  else
  if result = 'POLVD' then stor_loc := 21
  else
  if result = 'POLVE' then stor_loc := 22
  else
  if result = 'ONPOLY' then stor_loc := 0

```

FILE: POLY.PAS

```

if result = 'ODPOLY' then stor_loc := 1
else
if result = 'CNPOLY' then stor_loc := 2
else
if result = 'CDPOLY' then stor_loc := 3
else
if result = 'GNPOLY' then stor_loc := 4
else
if result = 'GDPOLY' then stor_loc := 5
else
if result = 'HNPOLY' then stor_loc := 6
else
if result = 'HQPOLY' then stor_loc := 7;

assign( polys, 'tf&polys.dat' );
reset( polys );
seek( polys, stor_loc);
write( polys, pol3 );
close( polys );

disppoly( result );
pause;

end;
(*****
* procedure: polymanip2
* version: 1.0
* date: 8 October 85
* description: This procedure will multiply a polynomial by a
* scalar, store the result in the desired location,
* and display the result
* passed variables: first, result, number
* procedures called: trim, pause, disppoly, spolymit
* files read: TF&POLS.DAT
* files written: TF&POLS.DAT
* called by: ppoly
* author: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
*****
)
procedure polmanip2( var first : cmdword; var result : cmdword;
var number : real );
var
stor_loc : integer;
polys : file of polynomial;

```

FILE: POLY.PAS

```

pol1,pol2 : polynomial;
i          : integer;

begin
  trim( first );
  if first = 'POLVA' then stor_loc := 18
  else
  if first = 'POLVB' then stor_loc := 19
  else
  if first = 'POLVC' then stor_loc := 20
  else
  if first = 'POLVD' then stor_loc := 21
  else
  if first = 'POLVE' then stor_loc := 22
  else
  if first = 'ONPOLY' then stor_loc := 0
  else
  if first = 'ODPOLY' then stor_loc := 1
  else
  if first = 'CNPOLY' then stor_loc := 2
  else
  if first = 'CDPOLY' then stor_loc := 3
  else
  if first = 'GNPOLY' then stor_loc := 4
  else
  if first = 'GDPOLY' then stor_loc := 5
  else
  if first = 'HNPOLY' then stor_loc := 6
  else
  if first = 'HDPOLY' then stor_loc := 7;

  assign( polys, 'tf&pol.s.dat' );
  reset( polys );
  seek( polys, stor_loc);
  read( polys, pol1);
  close( polys );

  polymlt( pol1, pol2, number );

  trim( result );

  if result = 'POLVA' then stor_loc := 18
  else
  if result = 'POLVB' then stor_loc := 19
  else
  if result = 'POLVC' then stor_loc := 20
  else
  if result = 'POLVD' then stor_loc := 21

```

FILE: POL1.PAS

```

else
if result = 'POLYE' then stor_loc := 22
else
if result = 'ONPOLY' then stor_loc := 0
else
if result = 'ODPOLY' then stor_loc := 1
else
if result = 'CNPOLY' then stor_loc := 2
else
if result = 'CDPOLY' then stor_loc := 3
else
if result = 'GNPOLY' then stor_loc := 4
else
if result = 'GDPOLY' then stor_loc := 5
else
if result = 'HNPOLY' then stor_loc := 6
else
if result = 'HDPOLY' then stor_loc := 7;

assign( polys, 'tf&polys.dat' );
reset( polys );
seek( polys, stor_loc );
write( polys, pol2 );
close( polys );

```

```

dispoly( result );
pause;

```

```

end;

```

```

*****
*
* procedure:  get_poly_name
* version:    1.0
* date:      6 September 85
* description: This procedure will get the name of a polynomial
*             from the screen
*
* Global variables used:  abort_command, blanks
* Global constants used:  as_assigned, crt_only
* Passed variables:      poly_name, row, col, abort_command
* Procedures called:      highlight, nohighlight, gotoxy,
*                       out_string, ucase, trim, pause,
*                       get_string, disp_msg, clear_msg
*
* called by:  ppoly
* author:     Susan K. Mashiko, Capt, USAF
*            Gary C. Tarczynski, Capt, USAF
*
*****

```

```

FILE: POLY.PAS

```

```
procedure get_poly_name( var poly_name : msg_line; row : integer;
  col : integer; abort_command : boolean );
```

```
label again;
begin
again:
  gotoxy( row, col );
  highlight;
  out_string( copy( blanks, 1, 5), crt_only);
  nohighlight;
  gotoxy( 20, 0 );
  out_string( blanks, crt_only );
  gotoxy( 20, 10 );
  highlight;
  out_string( ' Your choice... ', as_assigned);
  nohighlight;
  get_string( poly_name, abort_command, as_assigned, ' ', '-' );
  ucase( poly_name );
  if ( ( poly_name = 'POLYA' ) or ( poly_name = 'POLVB' ) or
    ( poly_name = 'POLVC' ) or ( poly_name = 'POLVD' ) or
    ( poly_name = 'POLVE' ) or ( poly_name = 'ONPOLY' ) or
    ( poly_name = 'ODPOLY' ) or ( poly_name = 'GNPOLY' ) or
    ( poly_name = 'CDPOLY' ) or ( poly_name = 'GNPOLY' ) or
    ( poly_name = 'GDPOLY' ) or ( poly_name = 'HNPOLY' ) or
    ( poly_name = 'HDPOLY' ) ) then
  begin
    gotoxy( row, col);
    out_string( poly_name, as_assigned );
  end
else
begin
  gotoxy( 21, 5 );
  disp_msg( 9 );
  pause;
  gotoxy( 21, 0 );
  clear_msg( 9 );
  goto again;
end;
end;
(*****
* procedure:  ppoly
* version:    2.0
* date:      19 September 85
* description: This procedure will get the name of a polynomial
*              from the screen
* global variables used:  cmdbuffer, abort_command
*****)
```

FILE: POLY.PAS

```

* Global variables changed: none
* Global constants used: as_assigned, wordnumber
* Passed variables: cmdbuffer, disp_msg,
* Procedures called: clear, disppoly,
* trim, gotoxy, out_string,
* polymanip, get_poly_name,
* get_r_num, polmanip2,
* pause
*
* Called by: disp
* Author: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
*
* Mod description: Added a help option for the DISPLAY POLY and
* rearranged this menu's call structure
*
* Mod date: 19 September 85
*
*****

```

```

procedure ppoly( var cmdbuffer : buffer;
var wordnumber : integer);

```

```

var
poly_obj : cmdword;
first : cmdword;
second : cmdword;
third : cmdword;
poly_name : msg_line;
number : real;

```

```

begin
poly_obj := cmdbuffer[ 3 ];
trim( poly_obj );
clear;

```

```

(* this is the catch code for the display of polynomials *)
if ( ( poly_obj = 'POLYA' ) or ( poly_obj = 'POLYB' ) or
( poly_obj = 'POLYC' ) or ( poly_obj = 'POLYD' ) or
( poly_obj = 'POLVE' ) or ( poly_obj = 'ONPOLY' ) or
( poly_obj = 'ODPOLY' ) or ( poly_obj = 'CNPOLY' ) or
( poly_obj = 'CDPOLY' ) or ( poly_obj = 'GNPOLY' ) or
( poly_obj = 'GDPOLY' ) or ( poly_obj = 'HNPOLY' ) or
( poly_obj = 'POLY' ) ) then
begin
disppoly( poly_obj );
pause;
end
else
if ( ( poly_obj = 'ADD' ) or ( poly_obj = 'SUBTRACT' ) or
( poly_obj = 'POLVMLT' ) ) then

```

FILE: POLY.PAS

```

begin
  clear;
  gotoxy( 3, 5 );
  disp_msg( 35 );
  gotoxy( 5, 20 );
  out_string( poly_obj, as_assigned );
  gotoxy( 15, 27 );
  out_string( poly_obj, as_assigned );
  gotoxy( 15, 48 );
  out_string( '.', as_assigned );
  get_poly_name( poly_name, 15, 15, abort_command );
  first := poly_name;
  get_poly_name( poly_name, 15, 37, abort_command );
  second := poly_name;
  get_poly_name( poly_name, 15, 52, abort_command );
  third := poly_name;
  polmanip( first, second, third, poly_obj );
end
else
if poly_obj = 'SPOLVMLT' then
  begin
  clear;
  gotoxy( 4, 0 );
  disp_msg( 60 );
  get_poly_name( poly_name, 14, 52, abort_command );
  first := poly_name;
  get_r_num( number, 16, 52, abort_command );
  get_poly_name( poly_name, 18, 52, abort_command );
  second := poly_name;
  polmanip2( first, second, number );
  end
else
if poly_obj = 'HELP' then
  begin
  clear;
  disp_msg( 21 );
  pause;
  clear;
  end;
end;

```

FILE: POLY.PAS


```

*****
**
**      PROCESER.PAS
**
** procedure contained: proces_error
** version: 1.1
** date: 16 august 1983
** description: This file contains the procedure that
**             handles command decoding errors. It
**             prompts the user for proper action to
**             take for error correction.
**             vincent m. parisi ii, capt., usaf
**
**
**
*****

```

```

*****
**
** procedure: proces_error
** version: 1.1
** date: 16 august 1983
** description: This procedure handles command decoding
**             errors. It prompts the user for proper
**             action to take for error correction.
**
** global variables used: help_level, cmdbuffer,
** passed variables: error_code, level, cmdbuffer,
**                   bufferpointer
**
** procedures called:
**                   gotoxy,
**                   pause,
**                   displa_commandword,
**                   highlight,
**                   nohighlight,
**                   disp_msg
**                   get_cmd
**
** called by:
** author: vincent m. parisi ii, capt., usaf
**
**
*****

```

```

procedure proces_error( error_code : char; level : integer;
cmdbuffer : buffer; bufferpointer : integer );

var i : integer;

begin
  if help_level > 1 then
    begin
      case error_code of
        'B', 'b' : begin

```

```

for i := 1 to ( level - 1 ) do
  displa_commandword( cmdbuffer, i );
highlight;
displa_commandword( cmdbuffer, level );
nohighlight;
for i := ( level + 1 ) to bufferpointer do
  displa_commandword( cmdbuffer, i );
end;
begin
  highlight;
  for i := 1 to ( level - 1 ) do
    displa_commandword( cmdbuffer, i );
  nohighlight;
  for i := level to bufferpointer do
    displa_commandword( cmdbuffer, i );
  end;
end;

```

```

'C', 'c' :

```

```

end;
end;

```

```

if help_level > 2 then
  begin
    gotoxy( 20, 0 );
    case error_code of

```

```

'B', 'b' :

```

```

  begin
    disp_msg( 4 );
    pause;
  end;
  begin;
    disp_msg( 5 );
    pause;
  end;

```

```

'C', 'c' :

```

```

end;
end;

```

```

end;

```

FILE: PROCESER.PAS

```

*****
**
** file: PROMPTCM.PAS
** procedure contained: prompt_cmd
** version: 1.2
** date: 30 October 1984
** description: This file contains the procedure that
** places the command line prompt at the
** row and column input.
** author: vincent m. parisi ii, capt., usaf
**
*****
*****
*****
*****
** procedure: prompt_cmd
** version: 1.2
** date: 30 October 1984
** description: This procedure places the command line
** prompt at the row and column input.
** global variables used: blanks
** global constants used: as_assigned, crt_only
** passed variables: row, col
** procedures called: highlight, gotoxy, nohighlight,
** out_string
** called by: get_cmd
** author: vincent m. parisi ii, capt., usaf
**
*****
*****
procedure prompt_cmd( row : integer; col : integer );
const logo = 'Enter Option >';
begin
  gotoxy( row, 0 );
  out_string( blanks, crt_only );
  gotoxy( row, col );
  highlight;
  out_string( logo, as_assigned );
  nohighlight;
end;

```

FILE: PROMPTCM.PAS

```

*****
**
** file: PROMPTHE.PAS
** procedure contained: prompt_help
** version: 2.2
** date: 27 September 84
** description: This module contains the procedures that
** display acceptable command words based
** upon the words already entered
** author: vincent m. paris |i, capt., usaf
**
*****

```

```

*****
**
** procedure: prompt_help
** version: 2.1
** date: 20 October 1983
** description: This procedure displays the acceptable
** command words based upon the words
** already entered.
**
** global constants used: crt_only, ENDCODE, DONEWORD
** passed variables: rec_num, row
** returned variables: rec_num
** procedures called: get_line, gotoxy, out_string
** SVideoLow, SVideoBold
** called by: get_cmd
** author: vincent m. paris |i, capt., usaf
**
*****

```

```

procedure prompt_help( rec_num : integer; row : integer );

```

```

const prompt_col_offset = 5;

```

```

var
  row_count : integer;
  j          : integer;
  decode    : dictionary;
  displayword : msg_line;

```

```

begin
  row_count := 0;

```

```

  repeat
    gotoxy( ( row + row_count ), prompt_col_offset );
    out_string( ' ', crt_only );
  until false;

```

```

FILE: PROMPTHE.PAS

```

```
j := 1;
repeat
  get_line( decode, rec_num );
  if decode.dictword <> DONEWORD then
    begin
      displayword := decode.dictword;
      SVideoLow( displayword, decode.abbrev+1 );
      SVideoBold( displayword, 1 );
      out_string( displayword, crt_only );
      j := j + 1;
    end;
  rec_num := decode.nomatchp;
until ( ( j = 6 ) or ( decode.nomatchp = ENDCODE ) );
row_count := row_count + 1;
until ( ( row_count = 6 ) or ( decode.nomatchp = ENDCODE ) );
end;
```

FILE: PROMPTE.PAS

```

*****
**                               READCOM.PAS                               **
**                               readcom                                **
** procedure contained:         1.1                                     **
** version:                    1.1                                     **
** date:                        28 august 1983                       **
** description:                 This module contains the procedure    **
**                               that gets a command line from the user. **
** author:                      vincent m. parisi ii, capt., usaf   **
**                               *****                               **
**                               *****                               **

```

```

*****
**                               *****                               **
** procedure:                  readcom                                **
** version:                    1.1                                     **
** date:                       28 august 1983                       **
** description:                 Reads command from user and splits it  **
**                               into individual words in the command  **
**                               buffer.                               **
** global variables used:      cmdbuffer, abort_command, blanks,    **
**                               string, macro_error                 **
** global variables changed:  cmdbuffer, string                     **
** global constants used:     buffersize, wordsize, terminal_only,  **
**                               as_assigned                           **
** passed variables:          cmdbuffer, bufferpointer,             **
**                               abort_command                       **
** returned variables:        cmdbuffer, bufferpointer             **
** procedures called:         get_string, ucase                     **
** called by:                  get_cmd                               **
** author:                     vincent m. parisi ii, capt., usaf   **
**                               *****                               **
**                               *****                               **

```

```

procedure readcom(var cmdbuffer : buffer; var bufferpointer : integer;
var abort_command : boolean );
var i : integer;
    j : integer;
    tword : msg_line;
    lencmd : integer;
begin
    (* main body of procedure readcom *)
    for i := bufferpointer to buffersize do      (* clear cmdbuffer not used to *)
        cmdbuffer[ i ] := copy( blanks, 1, wordsize ); (* spaces

```

```

*)
(* get the command from either the macro file or the terminal as assigned
   if there is an error while in macro, then get input from terminal *)
if macro_error then
  get_string( string, abort_command, terminal_only, ' ', '~' )
else
  get_string( string, abort_command, as_assigned, ' ', '~' );
(* if there is no abort command in the string, process the command string.
   * break up the command string into individual words and put each word in
   * the command buffer, left justified with right filled spaces. return with
   * bufferpointer at next free command buffer position.
if abort_command = no then
  begin
  ucase ( string ); (* change entire command to upper case *)
  lencmd := length( string );
  i := 1;
  repeat
    while ((string[i] = ' ') and ( i <= lencmd )) do
      i := i + 1;
    if (( string[i] > ' ') and ( i <= lencmd )) then
      begin
        j := 0;
        while (( string[j + i] <> ' ') and ( ( i + j ) <= lencmd )) do
          j := j + 1;
        tword := copy( string, i, j );
        tword := concat(tword, ' ');
        cmdbuffer[bufferpointer] := copy(tword,1,12);
        bufferpointer := bufferpointer + 1;
        i := i + j;
      end;
  until i >= lencmd;
  end;
end;
(* end of procedure readcom *)

```

FILE: READCOM.PAS

```

*****
**
** file: reals out_real
** procedures included: get_reals, out_real
** version: 2.4
** date: 19 August 1985
** description: contains the procedures to input and output
** real numbers.
** author: vincent m. parisi, capt, usaf
** Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt, USAF
**
*****

```

```

*****
**
** procedure: out_real
** version: 1.4
** date: 2 sep 83
** description: outputs real numbers to the crt or any of
** the required files
** global variables used: crt, trans, printer, temp, list_dev,
** trans_file
** global variables changed: none
** passed variables: number, field_width, dest
** files written: PRINTER.OUT, TRANSACTION, TEMP.OUT
** called by: many
** author: vincent m. parisi ii, capt., usaf
**
*****

```

```

procedure out_real( number : real; field_width : integer; dest : char);
begin
  case dest of
    'C', 'c' : write( number:field_width );
    'P', 'p' : writeln( list_dev, number:field_width );
    'B', 'b' : begin
                  write( number:field_width );
                  writeln( list_dev, number:field_width );
                end;
    'A', 'a' : begin
                  if crt then write( number:field_width );
                  if trans then writeln( trans_file, number:field_width );
                  if printer then writeln( list_dev, number:field_width );
                  if temp then writeln( temp_file, number:field_width );
                end;
  end;
end;

```



```

end;
end;
(*****
*
* procedure: get_reals
* version: 1.0
* date: 19 August 1985
* description: Handles the input of real numbers. Normal system
* real number input has error checking after the
* <CR> has been entered. This is unsatisfactory
* when using a particular layout on the screen as
* the error message will more than likely occur at
* an inappropriate place. This routine uses redirect-
* ed IO in that the input is read into a string
* through filters which only accept valid real
* number characters. The string is null or with just
* one space, the operator just entered a <CR> so
* 0.0 is returned.
*
* Global variables used: abort_command, string
* Global variables changed: string
* Global constants used: as_assigned
* Passed variables: number, abort_command
* Procedures called:
* get_string, gotoxy, pause,
* disp_msg, clear_msg, highlight,
* out_string, nohighlight
*
* called by: many
* author: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
*
* *****
*)
procedure get_real( var number : real; var abort_command : boolean );
var
  ch : char;
  result : integer;
  i : integer;
begin
  string := '';
  number := 0.00;
  i := 1;
  while i <= 1 do
  begin
    get_string( string, abort_command, as_assigned, '0', '9' );
    if abort_command then
      exit;
    val( string, number, result);
  end;
end;

```

FILE: REALS.PAS

```
if result <> 0 then
begin
  gotoxy( 20, 5 );
  disp_msg( 9 );
  pause;
  gotoxy( 20, 0 );
  clear_msg( 9 );
  gotoxy( 20, 10 );
  highlight;
  out_string( 'Your number...', as_assigned );
  nohighlight;
end
else
  i := i + 1;
end; (* end of while loop *)
end; (* end of procedure *)
```

FILE: REALS.PAS

(*****
**
** file: RECOVER.PAS
** procedure contained: recover
** version: 2.0
** date: 19 Sep 85
** description: This file contains the procedure to copy the
** user specified files into the ICECAP TF&POLS.DAT
** file and the matrix.dat file
** author: Susan K. Mashiko, Capt, USAF
** Gary C. Tarczynski, Capt, USAF
**
**
**
**
*****)

(*****
*
* procedure: recover
* version: 2.0
* date: 19 Sep 85
* description: This file contains the procedure to copy the
* user specified files into the ICECAP TF&POLS.DAT
* file and the MATRIX.DAT file
* global variables used: abort_command, blanks
* global constants used: as_assigned, crt_only
* files read: user specified transfer function file
* and matrix file
* files written: TF&POLS.DAT, MATRIX.DAT
* procedures called: clear, gotoxy,
* disp_msg, get_string,
* pause, clear_msg
* out_string
*
* called by: select
* author: Susan K. Mashiko, Capt, USAF
* Gary C. Tarczynski, Capt, USAF
* mod description: Code was added to limit the length of the file
* name.
* modifier: Author
* mod date: 19 Sep 85
*
*
*****)

overlay procedure recover;

label
repeat1;
repeat2;

FILE: RECOVER.PAS

```

var
  polys      : file of polynomial;
  polya      : file of polynomial;
  pol        : polynomial;
  i          : integer;
  your_name  : msg_line;
  mats       : file of matrix;
  mata       : file of matrix;
  mat        : matrix;

begin
  clear;

  (* copy user file into tf&polys.dat *)
  repeat;
    gotoxy( 4, 0 );
    disp_msg( 38 );
    gotoxy( 10, 24 );

    (* get the user specified file name from the user *)
    get_string( your_name, abort_command, as_assigned, ' ', '-' );
    if abort_command then exit;
    if length( your_name ) > 8 then
      begin
        gotoxy( 20, 10 );
        disp_msg( 44 );
        pause;
        clear_msg( 44 );
        gotoxy( 10, 0 );
        out_string( blanks, crt_only );
        goto repeat;
      end;

    assign( polys, 'tf&polys.dat' );
    rewrite( polys );

    assign( polya, your_name );
    reset( polya );

    for i := 0 to 22 do
      begin
        seek( polya, f );
        read( polya, pol );
        seek( polys, f );
        write( polys, pol );
      end;
    close( polys );
    close( polya );

```

FILE: RECOVER.PAS

```
(* copy user file into matrix.dat *)
repeat2;
gotoxy( 12, 0 );
disp_msg( 39 );
gotoxy( 18, 24 );

(* get the user specified file name from the user *)
get_string( your_name, abort_command, as_assigned, ' ', '-');
if abort_command then exit;
if length( your_name ) > 8 then
begin
gotoxy( 20, 10 );
disp_msg( 44 );
pause;
clear_msg( 44 );
gotoxy( 18, 0 );
out_string( blanks, crt_only );
goto repeat2;
end;

assign( mats, 'matrix.dat' );
rewrite( mats );

assign( mata, your_name );
reset( mata );

for i := 0 to 4 do
begin
seek( mata, i );
read( mata, mat );
seek( mats, i );
write( mats, mat );
end;
close( mats );
close( mata );

end;
```

FILE: RECOVER.PAS


```

begin
  trim( call_routine );
  if call_routine <> 'STOP' then
    begin
      if call_routine = 'DEFINE' then      (* added 12 Aug 85 *)
        define( cmdbuffer, number_of_commands )
      else
        if call_routine = 'HELP' then
          help( cmdbuffer, number_of_commands )
        else
          if call_routine = 'DISPLAY' then  (* added 4 Sep 85 *)
            disp( cmdbuffer, number_of_commands )
          else
            if call_routine = 'COPY' then   (* added 4 Sep 85 *)
              ccopy( cmdbuffer )
            else
              if call_routine = 'MODIFY' then
                modify( cmdbuffer, number_of_commands ) (* added 22 Sep 85 *)
              else
                if call_routine = 'RECOVER' then
                  recover
                (* added 9 Sep 85 *)
              else
                if call_routine = 'UPDATE' then
                  update
                (* added 9 Sep 85 *)
              else
                if call_routine = 'FORM' then
                  form
                else
                  if call_routine = 'FREQ/RESP' then
                    frequency_response
                  (* added 11 Oct 85 *)
                else
                  begin
                    (* Print out command buffer and call routine name *)
                    writeln;
                    writeln( 'SELECT: ', call_routine );
                    i := 1;
                    while (cmdbuffer[i] <> ' ' ) and (i <= buffersize) do
                      begin
                        write( cmdbuffer[i] );
                        i := i + 1;
                      end;
                    pause;
                  end;
                end;
            end;
          end;
        end;
      end;
    end;
  end;
end;

```

FILE: SELECT.PAS


```
r.dl := ord(c); (* Character in dl is sent to the display *)
msdos(r);
end;
```

FILE: STDOOT.PAS *** IBM ONLY ***

```

*****
**
** file: TERMINAL.PAS
**
** procedures contained:
** clear, gotoxy, highlight,
** nohighlight, graphics, nographics,
** VideoLow, SVideoLow, VideoBold,
** SVideoBold, ClearScreen
**
** version:
** 3.0
** date: 12 Dec 84
** description: This file contains the procedures that
** interface with the terminal.
**
** global variables used:
** term,
** stat_on,
** stat_line
**
** global constants used:
** term_length,
** stat_line_width
**
** author: vincent m. parisi ii, capt., usaf
**
*****
(*****
*
* procedure: graphics
*
* version: 2.0
* date: 21 oct 83
* description: This procedure places the terminal in
* graphics mode.
*
* global variables used: term
*
* global constants used: term_length
*
* procedures called: none
*
* called by: many
*
* author: vincent m. parisi ii, capt., usaf
*
*****
)

procedure graphics;
var i : integer;
begin
for i := 41 to ( term[ 40 ] + 40 ) do
write( chr( term[ i ] ) );
end;
(*****
*
* procedure: nographics
*****
)

```

FILE: TERMINAL.PAS

```

*
* version: 2.0
* date: 21 oct 83
* description: This procedure removes the terminal
* from graphics mode.
*
* global variables used: term
* global constants used: term_length
* procedures called: none
* called by: many
* author: vincent m. parisi ii, capt., usaf
*
*****

```

```

procedure nographics;
var i : integer;

```

```

begin
for i := 48 to ( term[ 47 ] + 47 ) do
write( chr( term[ i ] ) );
end;

```

```

*****
*
* procedure: highlight
* version: 2.0
* date: 21 oct 83
* description: This procedure puts the screen in reverse
* video.
*
* global variables used: term
* global constants used: term_length
* procedures called: none
* called by: many
* author: vincent m. parisi ii, capt., usaf
*
*****

```

```

procedure highlight;

```

```

var i : integer;

```

```

begin
for i := 28 to ( term[ 27 ] + 27 ) do
write( chr( term[ i ] ) );
end;

```

```

*****
*
* procedure: nonhighlight
* version: 2.0
* date: 21 oct 83
*

```

FILE: TERMINAL.PAS

```

* description: This procedure takes the screen out of
* reverse video.
* global variables used: term
* global constants used: term_length
* procedures called: none
* called by: many
* author: vincent m. parisi ii, capt., usaf
*
*****

```

```

procedure nohighlight;

```

```

var i : integer;

```

```

begin
  for i := 35 to ( term[ 34 ] + 34 ) do
    write( chr( term[ i ] ) );
  end;

```

```

*****
* procedure: gotoxy
* version: 2.0
* date: 21 oct 83
* description: This procedure places the cursor at the x
* and y coordinates passed to it. It is
* capable of sending an initial string of
* characters, either the row or column
* (whichever is required first), then an
* intermediate string of characters, the
* other address, and finally a trailing string
* of characters if required. Offsets if any
* are added prior to sending the row/column.
*
* global variables used: term_length
* global constants used: row_col
* passed variables: row, col
* returned variables: row, col
* procedures called: none
* called by: many
* author: vincent m. parisi ii, capt., usaf
* modifier: Paul A. Moore, CAPT, USAF
* mod description: Modified to allow different terminal types.
* H19/29 = 0, VT100 = 1
*
*****

```

```

procedure gotoxy( row : integer; col : integer );

```

```

var i : integer;

```

```

FILE: TERMINAL.PAS

```

```

procedure ttype(switch:integer; wchar : integer); (* terminal type *)
begin
  if switch = 0 then
    write ( chr(wchar) )      (* M19/29 terminal *)
  else
    write( wchar )          (* VT100 terminal *)
  end;

begin
  row := row + term[ 7 ];
  col := col + term[ 8 ];
  (* add row and col offsets *)

  (* send out initial string *)
  for i := 2 to ( term[ 1 ] + 1 ) do
    write( chr( term[ i ] ) );
  if term[ 9 ] = 0 then
    ttype(term[90],row)
  else
    ttype(term[90],col);
  (* send out intermediate *)
  for i := 11 to ( term[ 10 ] + 10 ) do
    write( chr( term[ i ] ) );
  (* string if any *)
  if term[ 9 ] = 0 then
    ttype(term[90],col);
  (* send out remaining row or col *)
  for i := 15 to ( term[ 14 ] + 14 ) do
    write( chr( term[ i ] ) );
  end;

  (*
  procedure:      clear
  version:        2.0
  date:           21 oct 83
  description:    This procedure clears the screen and homes
                  the cursor. If status line is visible
                  (stat_on = true) then the status line is
                  displayed.
  global variables used:  term, stat_on, stat_line
  global constants used:
  procedures called:      gotoxy
  called by:              many
  *)

```

FILE: TERMINAL.PAS

```
* author: vincent m. parisi ii, capt., usaf *
* .....*)
```

```
procedure clear;
```

```
var i : integer;
```

```
begin
  for i := 1 to term[ 19 ] do
    write( chr( term[ 19 + i ] ) );
```

```
  if stat_on then
```

```
    begin
```

```
      gotoxy( 22,0 );
```

```
      write( status_line );
```

```
      gotoxy( 0, 0 );
```

```
    end;
```

```
end;
```

```
(*****
* procedure: ClearScreen
* version: 1.0
* date: 12 Dec 84
* description: This procedure clears the screen and homes
* the cursor.
* global variables used: term, stat_on, stat_line
* global constants used: term_length,
* stat_line_width
* procedures called: none
* called by: many
* author: Paul A. Moore, Capt., USAF
* .....*)
```

```
procedure ClearScreen;
```

```
var i : integer;
```

```
begin
```

```
  for i := 1 to term[ 19 ] do
```

```
    write( chr( term[ 19 + i ] ) );
```

```
end;
```

```
(*****
* procedure: VideoLow
* version: 1.0
* .....*)
```

```
FILE: TERMINAL.PAS
```

```

*
*      27 Sep 84
*
* description: This procedure puts the screen into low video.
*
* global variables used:      term
*
* global constants used:    term_length
*
* procedures called:        none
*
* called by:                many
*
* author:                   Paul A. Moore, capt., usaf
*
*****

```

```

procedure VideoLow;

```

```

var i : integer;

```

```

begin
  for i := 71 to ( term[ 70 ] + 70 ) do
    write( chr( term[ i ] ) );
end;

```

```

*****
*
* procedure:                SVideoLow
*
* version:                  1.0
*
* date:                     27 Sep 84
*
* description: This procedure inserts the character string to
*
* put the screen into low video into the input
*
* string at the given position. It then returns
*
* the modified string.
*
* global variables used:    term
*
* global constants used:    term_length
*
* passed variables:         Instring, pos
*
* procedures called:        none
*
* called by:                many
*
* author:                   Paul A. Moore, capt., usaf
*
*****

```

```

procedure SVideoLow(var Instring : msg_line; pos : integer );

```

```

var i : integer;
    tempstr : string[10];

```

```

begin
  tempstr := ''; (* null string *)
  for i := 71 to ( term[ 70 ] + 70 ) do
    tempstr := concat(tempstr, chr(term[i]) );
  Insert(tempstr,Instring,pos);
end;

```

```

FILE: TERMINAL.PAS

```

```

*****
* procedure: VideoBold
* version: 1.0
* date: 27 Sep 84
* description: This procedure writes the character string
* to put the screen into bold video.
* global variables used: term
* global constants used: term_length
* procedures called: none
* called by: many
* author: Paul A. Moore, capt., usaf
*****

```

```

procedure VideoBold;
var i : integer;
begin
  for i := 77 to ( term[ 76 ] + 76 ) do
    write( chr(term[i]) );
end;

```

```

*****
* procedure: sVideoBold
* version: 1.0
* date: 27 Sep 84
* description: This procedure inserts the character string
* to put the screen into bold video into the
* input string at the given position. It then
* returns the modified string.
* global variables used: term
* global constants used: term_length
* passed variables: instring, pos
* procedures called: none
* called by: many
* author: Paul A. Moore, capt., usaf
*****

```

```

procedure SVideoBold(var Instring : msg_line; pos : integer );
var i : integer;
    tempstr : string[10];
begin
  tempstr := ''; (* null string *)

```

FILE: TERMINAL.PAS


```

for i := 77 to ( term[ 76 ] + 76 ) do
  tempstr := concat(tempstr, chr(term[i]) );
  Insert(tempstr, Instring, pos);
end;
(*****
*
* procedure: Rectangle
* version: 1.0
* date: 27 Sep 84
* description: This procedure draws a rectangle of the given
* dimensions on the video screen.
* global variables used: term
* global constants used: term.length
* passed variables: line, column, width, height
* procedures called: graphics, gotoxy, nographics
* called by: many
* author: Paul A. Moore, capt.. usaf
*
*****
procedure Rectangle(line,column,width,height : integer);
var
  i, L1, C1 : integer;
begin
  graphics;
  L1 := line + height - 1;
  C1 := column + width - 1;
  gotoxy(line,column); (* upper left corner of Rectangle *)
  write(chr(term[64])); (* upper left corner *)
  for i := column+1 to C1-1 do
    write(chr(term[55])); (* top of Rectangle *)
  write(chr(term[61])); (* upper right corner *)
  (* columns of Rectangle *)
  for i := line+1 to L1-1 do
    begin
      gotoxy(i,column); write(chr(term[54]));
      gotoxy(i,C1); write(chr(term[54]));
    end;
  gotoxy(L1,column); (* lower left corner of Rectangle *)
  write(chr(term[63])); (* lower left corner *)
  for i := column+1 to C1-1 do
    write(chr(term[55])); (* bottom of Rectangle *)
  write(chr(term[62])); (* lower right corner *)

```

FILE: TERMINAL.PAS

nographics;

end;

FILE: TERMINAL.PAS

```

*****
**
** file: TRIM.PAS
** procedures contained: trim
** version: 1.1
** date: 30 June 1984
** description: This file contains the procedure to
** trim trailing blanks from command
** words.
** author: vincent m. parisi ii, capt., usaf
**
*****
(*****
*
* procedure: trim
* version: 1.1
* date: 30 June 1984
* description: This procedure trims trailing blanks
* from command word strings.
*
* global constants used: wordsize
* passed variables: cmdword
* returned variables: cmdword
* called by: displayc
* author: vincent m. parisi ii, capt., usaf
* modifier: Paul A. Moore, Capt, USAF
*
*****
)

procedure trim( var cmdword : cmdword );

var i : integer;

begin
  i := length(cmdword);
  while (cmdword[i] = ' ') do
    i := i - 1;
  cmdword := copy(cmdword,1,i);
end;

```

FILE: TRIM.PAS

```

*****
**
**           UCASE.PAS
**
** procedure contained:  ucase
** version:             1.1
** date:               28 august 1983
** description:        This module contains the procedure
**                    to convert lower case strings to upper.
** author:             vincent m. parisi ii, capt., usaf
**
**
*****

```

```

*****
**
** procedure:    ucase
** version:     1.1
** date:       28 august 1983
** description: Converts lower case strings to upper.
** passed variables:    instrng
** returned variables:  instrng
** called by:          readcom
** author:            vincent m. parisi ii, capt., usaf
**
*****

```

```

procedure ucase( var instrng : msg_line);
var i      : integer;
begin
  for i := 1 to length(instrng) do
    instrng[i] := UpCase(instrng[i]);
end;

```

FILE: UCASE.PAS

```
(*****  
**  
** file: UPDATE.PAS  
** procedure contained: update  
** version: 2.0  
** date: 19 Sep 85  
** description: This file contains the procedure to copy the  
** ICECAP tf&pols.dat file and the matrix.dat file  
** into user specified files.  
** author: Susan K. Mashiko, Capt, USAF  
** Gary C. Tarczynski, Capt, USAF  
**  
*****  
(*****  
**  
** procedure: update  
** version: 2.0  
** date: 19 Sep 85  
** description: This file contains the procedure to copy the  
** ICECAP TF&POLS.DAT file and the MATRIX.DAT file  
** into user specified files.  
** global variables used: abort_command, blanks  
** global constants used: as_assigned, crt_only  
** files created: user specified transfer function file  
** and matrix file  
** files written: same as files created  
** files read: TF&POLS.DAT, MATRIX.DAT  
** procedures called: clear, gotoxy,  
** disp_msg, get_string,  
** pause, clear_msg  
** out_string  
**  
** called by: select  
** author: Susan K. Mashiko, Capt, USAF  
** Gary C. Tarczynski, Capt, USAF  
** mod description: Code was added to limit the length of the file  
** name.  
** modifier: Author  
** mod date: 19 Sep 85  
**  
*****  
overlay procedure update:  
label  
repeat1,  
repeat2;
```

FILE: UPDATE.PAS

```

var
  polys      : file of polynomial;
  poly_a     : file of polynomial;
  pol        : polynomial;
  i          : integer;
  your_name  : msg_line;
  mats      : file of matrix;
  mata      : file of matrix;
  mat       : matrix;

begin
  clear;
  (* update user file with tf&polys.dat *)
  repeat;
    gotoxy( 4, 0 );
    disp_msg( 36 );
    gotoxy( 10, 25 );

    (* get the user specified file name form the user *)
    get_string( your_name, abort_command, as_assigned, ' ', '-');
    if abort_command then exit;
    if length( your_name ) > 8 then
      begin
        gotoxy( 20, 10 );
        disp_msg( 44 );
        pause;
        clear_msg( 44 );
        gotoxy( 10, 0 );
        out_string( blanks, crt_only );
        goto repeat;
      end;
    assign( polys, 'tf&polys.dat' );
    reset( polys );
    assign( poly_a, your_name );
    rewrite( poly_a );

    for i := 0 to 22 do
      begin
        seek( polys, 1 );
        read( polys, pol );
        seek( poly_a, 1 );
        write( poly_a, pol );
      end;
    close( polys );
    close( poly_a );

    (* now update user file with matrix.dat *)

```

FILE: UPDATE.PAS

```
repeat2:
gotoxy( 12, 0 );
disp_msg( 37 );
gotoxy( 18, 25 );

(* get the user specified file name from the user *)
get_string( your_name, abort_command, as_assigned, '', '-');
if abort_command then exit;
if length( your_name ) > 8 then
begin
gotoxy( 20, 10 );
disp_msg( 44 );
pause;
clear_msg( 44 );
gotoxy( 18, 0 );
out_string( blanks, crt_only );
goto repeat2;
end;

assign( mats, 'matrix.dat' );
reset( mats );

assign( mata, your_name );
rewrite( mata );

for i := 0 to 4 do
begin
seek( mats, i );
read( mats, mat );
seek( mata, i );
write( mata, mat );
end;
close( mats );
close( mata );

end;
```

FILE: UPDATE.PAS

```

*****
**
** file: VALNDEC.PAS
** procedure contained: val_n_dec
** function contained: check_word
** version: 1.7
** date: 29 October 1984
** description: This file contains the procedures that
** validate and decode the user input
** command line.
** author: vincent m. parisi ii, capt., usaf
**
*****

```

```

*****
**
** function: check_word
** version: 1.1
** date: 29 October 1984
** description: This function returns "true" if there is
** a match between the dictionary word and
** the command word. The function takes into
** account abbreviations of command words.
**
** passed variables: decode, command
** returned variables: command, check_word
** procedures called: trim
** called by: val_n_dec
** author: Paul A. Moore, capt., usaf
**
*****

```

```

function check_word(decode : dictionary; command : cmdword) : boolean;

```

```

var
  dword : cmdword;
  d_len : integer;
  cmd_len : integer;
  i : integer;
begin
  dword := decode.dictword; (* get rid of trailing blanks *)
  trim(dword);
  trim(command);
  check_word := false; (* default to no match *)
  if command = dword then
    check_word := true

```

```

FILE: VALNDEC.PAS

```



```

else
begin
  d_len := length(dword);
  cmd_len := length(command);
  (* make sure the command isn't too long or short *)
  if (cmd_len >= decode.abbrev) and (cmd_len <= d_len) then
  begin (* compare characters *)
    i := 1;
    while (i <= cmd_len) and
      ( UpCase(Command[i]) = UpCase(dword[i]) ) do
      i := i + 1;
    if ( i = cmd_len + 1 ) then check_word := true;
  end;
end;
end; (* end check_word *)
(*****
* procedure: val_n_dec
* version: 1.6
* date: 16 august 1983
* description: This procedure validates and decodes the
* command line input by the user. The
* process begins by recovering record 1
* from the syntax table, using get_line.
* A comparison is made with the first
* word in the command buffer. If there is
* no match, the next record pointed to by
* nomatchp is retrieved with get_line.
* The comparisons continue until a match
* is found and the routine goes to the
* next level. If at any level no match
* is found after exhausting all the
* possibilities for that level then the
* error code is set to 'b' and the rou-
* tine is exited. If a valid command
* is decoded but there are still some
* words present in the commandbuffer, the
* error code is set to 'c'. The error
* code is set to 'n' for a valid command.
* global variables used: cmdbuffer, call_routine
* global variables changed: cmdbuffer, call_routine
* global constants used: DONEWORD, ENDCODE
* passed variables:
* level, rec_num, error_code,
* num_of_commands, cmdbuffer,
* call_routine
* returned variables:
* level, rec_num, error_code,
* cmdbuffer, call_routine
*)

```

```

*   procedures called:   check_word, trim, get_line
*   called by:         get_com
*   author:            vincent m. parisi ii, capt., usaf
*   *****
procedure val_n_dec( var level : integer; var rec_num : integer;
var error_code : char; num_of_commands : integer;
var cmdbuffer : buffer; var call_routine : cmdword );
var   last_rec_num : integer;
      cmd           : cmdword;
begin
  (* "last_rec_num" points to the beginning of a list of options *)
  last_rec_num := rec_num;      (* save rec_num *)
  (* If the word does not match, get the next
  record which is pointed to by nomatchp. repeat this until we
  run out of words indicated by an "ENDCODE" in nomatchp or there
  is a match.
  *)
  while (( error_code = 'a' ) and ( level <= ( num_of_commands + 1 ))) do
    begin
      (* get the syntax line for entry rec_num *)
      get_line( decode, rec_num );
      cmd := decode.dictword;
      trim(cmd);
      if cmd = DONEWORD then
        begin
          if level = ( num_of_commands + 1 ) then
            error_code := 'n';
            call_routine := decode_dict.words[decode.matchp];
          else
            begin
              error_code := 'c';
              rec_num := last_rec_num;
            end;
          end
        else
          if cmdbuffer[ level ] = ' ' then
            error_code := 'd';
          else
            if check_word(decode.cmdbuffer[level]) then
              begin

```

FILE: VALNDEC.PAS

```

cmdbuffer[level] := decode.dictword; (* replace possible abbr. *)
level := level + 1;
rec_num := decode.matchp;
last_rec_num := rec_num;
end
else
if decode.nomatchp = ENDCODE then
begin
error_code := 'b';
rec_num := last_rec_num;
end
else
rec_num := decode.nomatchp;
end; (* end while *)
end;
end;

```

```

(* point to first record *)
(* of options for the next *)
(* level of commands *)
(* invalid command word *)

```

```

(* try next option *)

```

FILE: VALDEC.PAS

Appendix F: BUILDDAT Text Files

This Appendix contains the five (5) text files that are used by BUILDDAT to install the menu system. These text files describe the hardware environment (TERM.TXT and PRINT.TXT), help text (HELP.TXT), and the menu structure (MENU.TXT and PARAM.TXT). These particular test files were developed for this thesis effort. If a text file has several different versions, the particular machine and configuration is annotated in the footer of the text file.

These files are in the order which they were presented above.

TERM.TXT
ZENITH Z-100

Cursor Positioning, initial char sequence

1 2
2 27
3 89
4 0
5 0
6 0
7 32
8 32
9 0
10 0
11 0
12 0
13 0
14 0
15 0
16 0
17 0
18 0
19 2
20 27
21 69
22 0
23 0
24 0
25 0
26 0
27 2
28 27
29 112
30 0
31 0
32 0
33 0
34 2
35 27
36 113
37 0
38 0
39 0
40 2
41 27
42 70
43 0
44 0
45 0
46 0
47 2
48 27
49 71

ESC

V

row offset
column offset

intermediate character(s)

terminating character(s)

Clear Screen, Home Cursor

ESC

E

Highlight (enter reverse video)

ESC

P

Normal Video (exit reverse video)

ESC

q

Enter Graphics Mode

ESC

F

Exit Graphics Mode

ESC

G

50 0
 51 0
 52 0
 53 0
 54 96 Graphics - vertical line
 55 97 Graphics - horizontal line
 56 98 Graphics - line intersection (up and down)
 57 94 Graphics - centered dot
 58 105 Graphics - solid square (space in reverse video)
 59 119 Graphics - line intersection (diagonal)
 60 115 Graphics - top "T"
 61 99 Graphics - upper right corner
 62 100 Graphics - lower right corner
 63 101 Graphics - lower left corner
 64 102 Graphics - upper left corner
 65 0
 66 0
 67 0
 68 0
 69 0
 70 2 Normal Video (exit reverse video)
 71 27 ESC
 72 113 q
 73 0
 74 0
 75 0
 76 2 Video Bold (enter reverse video)
 77 27 ESC
 78 112 p
 79 0
 80 0
 81 0
 82 0
 83 0
 84 0
 85 0
 86 0
 87 0
 88 0
 89 0
 90 0 Cursor Addressing. 0=decimal 1=ASCII
 91 0
 92 0 Row offset for beginning of plot. (not used)
 93 0 Column offset for beginning of plot. (not used)
 94 0 Vertical resolution of plotting area. (not used)
 95 0 Horizontal resolution of plotting area. (not used)

Cursor Positioning, initial char sequence

1 2 ESC
3 91 |
4 0
5 0
6 0
7 1 row offset
8 1 column offset
9 0
10 1 intermediate character(s)
11 59 ;
12 0
13 0
14 1 terminating character(s)
15 72 H
16 0
17 0
18 0
19 4 Clear Screen, Home Cursor
20 27 ESC
21 91 |
22 50 2
23 74 J
24 0
25 0
26 0
27 4 Highlight (enter reverse video)
28 27 ESC
29 91 |
30 55 7
31 109 m
32 0
33 0
34 4 Normal Video (exit reverse video)
35 27 ESC
36 91 |
37 48 0
38 109 m
39 0
40 0 Enter Graphics Mode
41 0
42 0
43 0
44 0
45 0
46 0
47 0 Exit Graphics Mode
48 0
49 0

50 0
51 0
52 0
53 0
54 179 Graphics - vertical line
55 196 Graphics - horizontal line
56 197 Graphics - line intersection (up and down)
57 250 Graphics - centered dot
58 219 Graphics - solid square (space in reverse video)
59 88 Graphics - line intersection (diagonal)
60 194 Graphics - top "T"
61 191 Graphics - upper right corner
62 217 Graphics - lower right corner
63 192 Graphics - lower left corner
64 218 Graphics - upper left corner
65 0
66 0
67 0
68 0
69 0
70 4 Normal Video (exit video bold)
71 27 ESC
72 91 l
73 48 0
74 109 m
75 0
76 4 Video Bold
77 27 ESC
78 91 l
79 49 i
80 109 m
81 0
82 0
83 0
84 0
85 0
86 0
87 0
88 0
89 0
90 1 Cursor Addressing, 1 = decimal 0 = ASCII
91 0
92 0 Row offset for beginning of plot. (not used)
93 0 Column offset for beginning of plot. (not used)
94 0 Vertical resolution of plotting area. (not used)
95 0 Horizontal resolution of plotting area. (not used)

1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 0
9 0
10 0
11 0
12 0
13 0
14 0
15 0
16 0
17 0
18 0
19 0
20 0
21 0
22 0
23 0
24 0
25 0
26 0
27 0
28 0
29 0
30 0
31 0
32 0
33 0
34 0
35 0
36 0
37 0
38 0
39 0
40 0
41 0
42 0
43 0
44 0
45 0
46 0
47 0
48 0
49 0

FILE: PRINT.TXT

50 0

FILE: PRINT.TXT

The power of the polynomial must be > 0 and < 10.
<\$> #2
What is the degree of the numerator... (max of 10)?
<\$> #3
...the denominator (max of 10)?

<\$> #4
INVALID Keyword or Abbreviation

<\$> #5
This is a valid command, the rest is extraneous.

<\$> #6
Numerator

<\$> #7
Denominator

<\$> #8
TRANSFER FUNCTION INPUT

<\$> #9
|

<\$> #10
| This is not a valid input, reenter

<\$> #11
| You cannot input a complex number for the last
| root. There is no room for its conjugate.

<\$> #12
| Do you want computer determined boundaries <C> or
| boundaries previously saved <S>...
| Enter an <S> or <C>...

<\$> #13
| Enter an <S> or <C>...or a <\$> to abort....

<\$> #14
| To exit, enter <\$>; for more, press <CR>.

<\$> #15
| Constant/Gain = Polynomial Constant/Gain = Roots

ICECAPPC SYSTEM HELP INFORMATION

The ICECAPPC program provides an environment for control system design and analysis hosted on a microcomputer. This system is menu driven and will provide help upon request.

The following is a list of the ICECAPPC main menu options:

- CHANGE - Allows the user to change analysis plane and sampling time.
- COPY - Allows the user to copy one transfer function into another or one matrix into another.
- DEFINE - Allows the user to define a transfer function or a matrix.
- DISPLAY - Any submenu option with an (*) has not been implemented yet. Displays the results, transfer functions, or matrices on the screen.
- FORM - Any submenu option with an (*) has not been implemented yet. Forms a CLTF from an OLTF or a combination of GTF and HTF.

FILE: HELP.TXT

- HELP - Provides on-line help to the user.
- MODIFY - Add a root or delete a root from a transfer function.
- PRINT - Change a single location of a matrix.
- RECOVER - Sends screen output to the printer.
- STOP - Any submenu option with and (*) has not been implemented yet.
- SWITCHES - Used to continue continue a previous session. Will copy files into ICECAPPC memory.
- UPDATE - Normal command for leaving ICECAPPC.

COMMAND INPUT

The command input structure is made up of a hierarchy of menus containing keywords (or command words). After a keyword is entered it is validated against the valid keywords for the current menu. If the keyword is valid the next lower menu of keywords is displayed or a prompt from the selected function appears. If an invalid keyword is entered the keyword is highlighted on the "Enter Option >" line and the user is prompted to enter a valid keyword. If the user knows the valid keywords at the next lower level menu for a keyword in the current menu the user may "type ahead" keywords for lower levels in the menu structure.

Example: HELP SYSTEM
 HELP FUNC

(FUNC is an abbreviation for FUNCTION)

Abbreviations are generally the first two or three letters of the keyword. Abbreviations are shown in bold for each command word. The Abbreviation shown in bold is the minimum number of characters necessary to identify a command word, additional characters of the command word may be entered. All characters entered for a command word will be validated against the valid command words.

If a mistake is noted prior to pressing the carriage return <CR>, just use the backspace or delete key to erase backwards to the error. Correct the error and retype the remainder of the command. If the program is prompting you for a command completion, you can only erase the characters internally back to the point that began that prompt.

At any point an input is expected from the user, (either command input or subroutine process) the user may abort the process and return to the command mode. If a separate program is being executed enter the "Exit" or "Abort" command for that program.

To abort a command enter '\$' followed by a carriage return and you will be returned to the top ICECAPPC menu.
 <\$> #16

CHANGE COMMAND

The CHANGE command is used to initialize the CHANGE functions of

FILE: HELP.TXT

ICECAPPC. This function allows the user to select the plane of analysis and to change the sampling time, TSAMP. The proper format for the CHANGE command is:

CHANGE PLANE or CHANGE TSAMP

The system will provide a menu of allowable plane changes in response to the first command. This option has not been implemented yet.
<\$> #17

COPY COMMAND

The command COPY is used to move the contents of one Transfer Function, called the Source, to a second Transfer Function called the Destination. The COPY command does not destroy the contents of the Source. COPY may also be used to move the contents of matrices and polynomials.

The proper format of the COPY command is as follows:

COPY <source_name> <destination_name> <CR>

For example:

COPY GTF OLTf <CR>
or
COPY MATA MATB <CR>

ICECAPPC will supply a menu of allowable transfer functions/matrices for both the source and the destination. ICECAPPC will not allow you to copy a transfer function to a matrix or vice versa.
<\$> #18

DEFINE COMMAND

The command DEFINE is used to initialize the Transfer Functions, Polynomials, and Matrices of ICECAPPC so that further calculations using those functions and/or matrices can be executed. The input to the system is set with this command.

If you try to manipulate a variable that has not been defined the system will prompt you for the necessary information. The proper format for the DEFINE command is as follows:

DEFINE (transfer function) (fact/poly)
or
DEFINE (matrix)

The system will prompt you for the required data. If you provide incorrect data for the prompt ICECAPPC will describe the error and prompt you for input again. For example, let us assume you have entered the

FILE: HELP.TXT

command:

DEFINE GTF POLY

The system prompt will be:

What is the degree of the numerator... (max of 10) ?

You would enter the integer value of the numerator degree. ICECAPPC will then display the following prompt:

... the denominator (max of 10) ?

Again input an integer value. ICECAPPC will then draw a display screen for the transfer function. ICECAPPC will highlight the first input area and provide a prompt:

Your number ...

Input the real number you would like in the input area. If you have selected the factored input format, and have entered the first half of a complex conjugate pair, ICECAPPC will enter the second half of the pair. If you try to enter the first half of a complex conjugate pair in the last root location, ICECAPPC will tell you this is an invalid input and prompt you to enter the root again.

The matrix definition option of ICECAPPC will also prompt you for the necessary information. You will be prompted for the number of rows and the number of columns. You are limited to a max of 10 rows and a max of 10 columns. ICECAPPC will draw the display screen for the matrix and highlight the input area and provide the prompt:

Your number....

ICECAPPC will provide a menu of transfer functions, polynomials, and matrices that may be specified with the DEFINE command.

<\$> #19

DISPLAY COMMAND

The command DISPLAY is used to write information onto the terminal screen. This information generally takes the form of plots of system response, the listing of the system specification, or a root locus plot. DISPLAY may also be used to display the current contents of transfer function and matrices in the data base of ICECAPPC. The proper format of the DISPLAY command is as follows:

DISPLAY (object)

FILE: HELP.TXT

ICECAPPC will provide a menu of allowable 'objects' in response to the DISPLAY command. Any 'object' followed by an asterisk (*) has not been implemented yet.
<\$> #20

FORM COMMAND

The FORM command is used to produce the CLTF from either the OLTF or a combination of GTF and HTF. The formula for producing the CLTF is as follows:

$$\begin{aligned} \text{CLTF} &= (\text{GAIN} * \text{GTF}) / (1 + \text{GAIN} * \text{GTF} * \text{HTF}) \\ \text{CLTF} &= (\text{GAIN} * \text{OLTF}) / (1 + \text{GAIN} * \text{OLTF}) \\ \text{CLTF} &= \text{GTF} + \text{HTF} \text{ (In Parallel)} \end{aligned}$$

This option will also form the OLTF from the GTF and the HTF. The formula for that operation is:

$$\text{OLTF} = \text{GTF} * \text{HTF}$$

<\$> #21

DISPLAY POLY COMMANDS

The ADD, SUBTRACT, and MULTIPLY options will prompt you for the names of the two polynomials you wish to ADD, SUBTRACT, or MULTIPLY together. ICECAPPC will also prompt you for the name of the polynomial that you wish the new polynomial stored into.

If you input one of the thirteen polynomial names the polynomial of your choice will be displayed on the screen.
<\$> #22

PRINT COMMAND

The command PRINT is used to write information to an external file as well as to the terminal screen. The contents of the external file can be sent to a printer at the conclusion of the design session. PRINT is most often used to summarize the last iteration of a design session. If you wish to view the data before it is sent to the print the DISPLAY command may be used. Further instructions for DISPLAY may be found using the HELP command. This is a good way to ensure a 'clean copy' to the printer. The proper format for the PRINT command is as follows:

PRINT (object)

ICECAPPC will provide a menu of allowable 'objects' in response to the PRINT command. Any 'object' followed by an asterisk (*) has not been implemented yet.
<\$> #23

RECOVER COMMAND

RECOVER is used to copy user specified files into ICECAPPC memory

FILE: HELP.TXT

so that you can continue a previous session at the point where you left off. The user files were previously specified with the UPDATE command. You may only RECOVER files that exist on the disk.

CAUTION: Be sure you input the transfer function and polynomial file name in response to the 'tf&pol.dat' prompt, and the matrix file name in response to the 'matrix.dat' prompt. If you mix the two ICECAPP will think there is NO data in either of its internal files.

<\$> #24

STOP COMMAND

The STOP command is used to exit gracefully from ICECAPP. Information is always stored in the ICECAPP 'tf&pol.dat' and the 'matrix.dat' files for later use. If you desire to save this information in a user specified file use the UPDATE command. STOP is the normal mode for leaving ICECAPP.

<\$> #25

MODIFY COMMAND

The modify command is used to change polynomials and matrices without redefining the entire polynomial or matrix. If you desire to change a transfer function this may be done by modifying the numerator polynomial, the denominator polynomial or both.

ADDRROOT:

The ADDRROOT command is used to modify or insert a root into a polynomial. In response to the ADDRROOT command ICECAPP will provide a menu of available polynomials. An example of the command line is:

ADDRROOT ODPOLY

After you have correctly entered the name of the polynomial ICECAPP will display the polynomial and prompt you with:

Your number ...

In addition the real area of the additional root will be highlighted. In response to the prompt enter a real number. You will then be prompted for the imaginary part of the root. If any input other than zero is made ICECAPP will calculate the conjugate of this root. After you have correctly input the root ICECAPP will recompute the polynomial and it will be displayed on the screen.

DELROOT:

The DELROOT command is used to remove or delete a root from a polynomial. In response to the DELROOT command ICECAPP will provide a menu of available polynomials. An example of a command line is:

DELROOT ONPOLY

After you have correctly entered the name of a polynomial ICECAPPC will display the polynomial and prompt you with:

The number of the root you wish to delete is....

The numbers of the roots can be found to the left of the factored display. If the root you have chosen to delete is complex the conjugate will also be deleted. After ICECAPPC has recomputed the new polynomial it will be displayed on the screen.

CHANGE:

The CHANGE command is used to modify a single location in a matrix. In response to a CHANGE command ICECAPPC will provide a menu of matrices that may be modified. An example of the command line is:

CHANGE MATA

After you have correctly entered the name of the matrix ICECAPPC will display the matrix and prompt you with:

The row of the location you wish to modify ?

Enter a valid row number. The system will then prompt:

...the column ?

Enter a valid column number. ICECAPPC will then highlight the location and provide the prompt:

Your number...

ICECAPPC will then store the new matrix in the correct location and re-display it to you.
<\$> #26

SWITCHES COMMAND

The command SWITCHES is used to flip various control switches ON and OFF. The proper format for the SWITCHES command is as follows:

SWITCH (object) (ON/OFF)

for example:

SWITCH PRINTER ON

ICECAPPC will provide a menu of allowable 'objects' for the SWITCHES command.

The switches of ICECAPPC and the functions they control:

FILE: HELP.TXT

ANSWER ON Causes all output to go to an external file
 OFF Output is displayed at user terminal
 CANCEL ON Cancels the common roots of transfer functions
 OFF Matching poles and zeros are not cancelled
 CLOSED ON Closed Loop (CLTF) used if there is a choice
 OFF Open Loop Transfer Function is used
 DECIBELS ON Magnitudes of plots are in decibels
 OFF Actual magnitude is output
 GRID ON Draw grid lines on plots
 OFF Omit grid lines from plots
 HERTZ ON Frequency data input/output in hertz
 OFF Frequency data input/output in rad/sec
 MAINMENU ON Display initial menu of ICECAPPC command words
 OFF Suppress display of initial menu

This option has not been implemented yet.
 <\$> #27

UPDATE COMMAND

The UPDATE command may be used to periodically write all of the contents of ICECAPPC memory files, 'tf&pois.dat' and 'matrix.dat' to user specified files. ICECAPPC will provide prompts for the user name for each of the files. This command is particularly useful when there is more than one user of ICECAPPC or a single user is working on more than one design in parallel. In order to write the user specified files into the ICECAPPC files use the RECOVER command.
 <\$> #28
 POLY

This command will allow you to enter the polynomial/transfer function of your choice with a polynomial format. ICECAPPC will ask you first for the 'Constant/Gain', followed by prompts for the coefficients of the polynomial terms.

FACTORED

This command will allow you to enter the polynomial/transfer function of your choice with the factored format. ICECAPPC will ask you first for the 'Constant/Gain', followed by prompts for the real and the imaginary portions of the roots. Remember if you want the root in the left half plane the real part must be negative.
 <\$> #29
 The system is S L O W L Y converging on a root, please be patient.
 <\$> #30
 What is the power of the polynomial... (max of 10)?
 <\$> #31
 POLYNOMIAL INPUT
 <\$> #32

FILE: HELP.TXT

- 3 #11
- 4 #12
- 5 #13
- 6 #14
- 7 #15
- 8 #16
- 9 #17
- 0 #18
- 1 #19
- 2 #20
- 3 #21
- 4 #22
- 5 #23
- 6 #24
- 7 #25
- 8 #26
- 9 #27
- 0 #28
- 1 #29
- 2 #30
- 3 #31
- 4 #32
- 5 #33
- 6 #34
- 7 #35

POLYNOMIAL MANIPULATION

This function will the second polynomial to(from) the first and store the result in the third polynomial location.

Available polynomials are:

ONPOLY	CNPOLY	GNPOLY	HNPOLY
ODPOLY	CDPOLY	GDPOLY	HDPOLY
POLVA	ØOLVB	POLYC	POLYD
POLVE			

<\$> #36
 Enter the name of the file you would like ICECAPPC's 'tf&pol.s.dat' copied into. 'tf&pol.s.dat' contains all of the transfer function and polynomial data. The file name should be eight characters or less, and should not contain any blank spaces. The file need not exist on your disk. ICECAPPC will create the file.

<\$> #37
 Your file name:
 Enter the name of the file you would like ICECAPPC'S 'matrix.dat' copied into. 'matrix.dat' contains all of the matrix data. The file name should be eight characters in length or less, and should not contain any blank spaces. The file need not exist on your disk. ICECAPPC will create the file.

<\$> #38
 Your file name:
 Enter the name of the file you would like to copy into ICECAPPC'S 'tf&pol.s.dat'. 'tf&pol.s.dat' contains all of the transfer function and polynomial data. The file name should be eight characters or less, and should not contain any blank spaces. The file must exist on your disk.

<\$> #39
 Your file name:
 Enter the name of the file you would like to copy into ICECAPPC'S 'matrix.dat' file. 'matrix.dat' contains all of the matrix data. The file name should be eight characters in length or less, and should not contain any blank spaces. The file must exist on your disk.

<\$> #40
 Your file name:

FILE: HELP.TXT

The number of rows or columns must be > 0 and <= 10

<\$> #41

How many rows in your matrix (max of 10) ?

<\$> #42

...columns (max of 10) ?

<\$> #43

MATRIX INPUT

<\$> #44

The user specified filename must be 8 characters or less.

<\$> #45

The row of the location you wish to modify ?

<\$> #46

...the column ?

<\$> #47

The constant/gain cannot be equal to zero.

<\$> #48

MATRIX MANIPULATION

This function will the second matrix to(from) the first
and store the result in the third matrix location.

Available matrices are:

<\$> # 49

MATA MATB MATC MATD MATE

DISPLAY MATRIX COMMANDS

The ADD, SUBTRACT, and MATXMULT will add, subtract, or multiply two matrices together and store the result in a user selected location. You will be prompted for the three matrix names.

The INVERSE command will invert a matrix and store the matrix in the user selected location. If you desire both the inverted matrix and the storage matrix may be the same.

The TRANSPOSE command will transpose a matrix and store the matrix in a user selected location. If you desire both the transposed matrix and the storage matrix may be the same.

The SCLRMULT command will multiply a matrix by a scalar and store the result in a user specified location. If you desire both the original and the resulting matrices may be the same.

The matrix name commands will display the named matrix to the user.

<\$> # 50

MATRIX DISPLAY

<\$> # 51

The rows and the columns of the two matrices you wish to add or

FILE: HELP.TXT

subtract must be the same. This option has been aborted.
<\$> # 52
The number of rows of the first matrix must equal the number of
columns of the second matrix. This option has been aborted.
<\$> # 53
The matrix must be square for inversion i.e. the matrix must have
the same number of rows as columns. This option has been aborted.
<\$> # 54

SINGLE MATRIX MANIPULATION

The available matrices are:

MATA MATB MATC MATD MATE

The matrix to be manipulated ...

The desired storage location ...

<\$> # 55
SCALAR * MATRIX = MATRIX

This option multiplies a scalar and a matrix together and stores the
resulting matrix in the desired location. The available matrices
are:

MATA MATB MATC MATD MATE

The matrix to be multiplied (MATRIX) ...

the scalar (REAL) ...

storage location (MATRIX) ...

<\$> # 56
The matrix you wish to invert is singular. This option has been aborted.

<\$> # 57
The row of the location you wish to modify ?

the column ?

<\$> # 58

If the location you wish to modify is not on this page type 'NEXT'.
The row of the location you wish to modify ?
the column ?

<\$> # 59

Form Command

1. Form OLTf ---> OLTf = GTF * HTF
2. Form CLTf ---> CLTf = (GAIN * GTF) / (1 + GAIN * GTF * HTF)
3. Form CLTF ---> CLTF = (GAIN * OLTf) / (1 + GAIN * OLTf)
4. Form CLTF ---> CLTF = GTF + HTF (In Parallel)

Your Selection (Integer) ...

<\$> # 60

FILE: HELP.TXT

SCALAR * POLYNOMIAL = POLYNOMIAL

This option multiplies a scalar and a polynomial together and stores the resulting polynomial in the desired location. The available polynomials are:

ONPOLY	CNPOLY	GNPOLY	HNPOLY
ODPOLY	CDPOLY	GDPOLY	HDPOLY
POLVA	POLYB	POLYC	POLYD
POLYE			

The polynomial to be multiplied (POLYNOMIAL) ...

the scalar (REAL) ...

storage location (POLYNOMIAL) ...

<\$> # 61

HELP Command

This help message is included for consistency. In all lower level menus the last entry on the bottom line is a HELP option. If you desire HELP with the ICECAPPC system the command line would be:

Enter Option > HELP SYSTEM

If you wish HELP with one of the main menu options the command line is:

Enter Option > HELP (option)

<\$> # 62

The leading coefficient of the polynomial cannot be zero. If you wish to reduce the order of the polynomial about this function and begin again.

<\$> # 63

message 63

<\$> # 64

message 64

<\$> # 65

message 65

<\$> # 66

message 66

<\$> # 67

message 67

<\$> # 68

message 68

<\$> # 69

message 69

<\$> # 70

message 70

<\$\$>

FILE: HELP.TXT

```
*****
; File Name: strawman menu structure for ICECAP-PC
; Creation date: 24 Jul 85
; Mod dat: 9 Aug 85 - add help structure
;         4 Sep 85 - add disp structure
;         7 Sep 85 - add insert structure
;         8 Sep 85 - add delete structure
;         18-21 Sep 85 - modified entire structure
;         7 Oct 85 - add time and freq structure
;         8 Oct 85 - add spolymlt
; *****
```

```
CHANGE
DISPLAY
MODIFY
RECOVER
SWITCHES
COPY
FORM
PRINT
STOP
UPDATE
DEFINE
HELP
!
```

```
*****
; define the menu options under CHANGE
; *****
```

```
$CHANGE
PLANE
TSAMP
HELP
!
```

```
*****
; define the call routines for CHANGE.PLANE and CHANGE.TSAMP
; *****
```

```
.CHANGE.PLANE = CHANGE
.CHANGE.TSAMP = CHANGE
.CHANGE.HELP = CHANGE
; *****
; define the menu options under COPY
; *****
```

```
$COPY
OLTF
CLTF
GTF
HTF
TF1
TF2
TF3
```

FILE: MENU.TXT

```
TF4
TF5
POLVA
POLVB
POLYC
POLVD
POLVE
ONPOLY
ODPOLY
CNPOLY
CDPOLY
GNPOLY
GDPOLY
HNPOLY
HDPOLY
MATA
MATB
MATC
MATD
MATE
HELP
! *****
: define menu for .COPY.OLTF option, this will be used by
: the other options under COPY
: *****
$COPY.OLTF
OLTF
CLTF
GTF
HTF
TF1
TF2
TF3
TF4
TF5
!
$COPY.POLYA
POLVA
POLVB
POLYC
POLVD
POLVE
ONPOLY
ODPOLY
CNPOLY
CDPOLY
GNPOLY
GDPOLY
```

FILE: MENU.TXT


```

HNPOLY
HDPOLY
:
: *****
: define the call routine for the COPY option
: *****
.COPY.HELP = COPY
.COPY.OLTF.OLTF = COPY
.COPY.OLTF.CLTF = COPY
.COPY.OLTF.GTF = COPY
.COPY.OLTF.HTF = COPY
.COPY.OLTF.TF1 = COPY
.COPY.OLTF.TF2 = COPY
.COPY.OLTF.TF3 = COPY
.COPY.OLTF.TF4 = COPY
.COPY.OLTF.TF5 = COPY
.COPY.POLYA.POLYA = COPY
.COPY.POLYA.POLVB = COPY
.COPY.POLYA.POLYC = COPY
.COPY.POLYA.POLYD = COPY
.COPY.POLYA.POLYE = COPY
.COPY.POLYA.ONPOLY = COPY
.COPY.POLYA.ODPOLY = COPY
.COPY.POLYA.CNPOLY = COPY
.COPY.POLYA.CDPOLY = COPY
.COPY.POLYA.GNPOLY = COPY
.COPY.POLYA.GDPOLY = COPY
.COPY.POLYA.HNPOLY = COPY
.COPY.POLYA.HDPOLY = COPY
: *****
: define submenus for the other options under COPY
: *****
$COPY.CLTF = COPY.OLTF
$COPY.GTF = COPY.OLTF
$COPY.HTF = COPY.OLTF
$COPY.TF1 = COPY.OLTF
$COPY.TF2 = COPY.OLTF
$COPY.TF3 = COPY.OLTF
$COPY.TF4 = COPY.OLTF
$COPY.TF5 = COPY.OLTF
$COPY.POLYA = COPY.POLYA
$COPY.POLVB = COPY.POLYA
$COPY.POLYC = COPY.POLYA
$COPY.POLYD = COPY.POLYA
$COPY.POLYE = COPY.POLYA
$COPY.ONPOLY = COPY.POLYA
$COPY.ODPOLY = COPY.POLYA
$COPY.CNPOLY = COPY.POLYA
$COPY.CDPOLY = COPY.POLYA

```

FILE: MENU.TXT

```

$COPY.GNPOLY = COPY.POLVA
$COPY.GDPOLY = COPY.POLVA
$COPY.HNPOLY = COPY.POLVA
$COPY.HDPOLY = COPY.POLVA
$COPY.MATA
MATA
MATB
MATC
MATD
MATE
!
; *****
; define the call routine for the matrix options
; *****
; .COPY.MATA.MATA = COPY
; .COPY.MATA.MATB = COPY
; .COPY.MATA.MATC = COPY
; .COPY.MATA.MATD = COPY
; .COPY.MATA.MATE = COPY
; *****
; define the submenus for the matrix options
; *****
$COPY.MATB = COPY.MATA
$COPY.MATC = COPY.MATA
$COPY.MATD = COPY.MATA
$COPY.MATE = COPY.MATA
; *****
; define the menu options under DEFINE
; *****
$DEFINE
OLTF
CLTF
GTF
HTF
TF1
TF2
TF3
TF4
TF5
POLVA
POLVB
POLYC
POLVD
POLVE
ONPOLY
ODPOLY
CNPOLY
CDPOLY
GNPOLY

```

FILE: MENU.TXT

```

GDPOLY
HNPOLY
HDPOLY
MATA
MATB
MATC
MATD
MATE
HELP
: *****
: define the menu options for DEFINE OLTF, these options are used by
: the other polynomial definition options
: *****
$DEFINE.OLTF
POLY
FACTORED
HELP
:
: define the call routines for POLY and FACTORED
.DEFINE.HELP = DEFINE
.DEFINE.OLTF.HELP = DEFINE
.DEFINE.OLTF.POLY = DEFINE
.DEFINE.OLTF.FACTORED = DEFINE
$DEFINE.OLTF = DEFINE.OLTF
$DEFINE.GTF = DEFINE.OLTF
$DEFINE.HTF = DEFINE.OLTF
$DEFINE.TF1 = DEFINE.OLTF
$DEFINE.TF2 = DEFINE.OLTF
$DEFINE.TF3 = DEFINE.OLTF
$DEFINE.TF4 = DEFINE.OLTF
$DEFINE.TF5 = DEFINE.OLTF
$DEFINE.POLYA = DEFINE.OLTF
$DEFINE.POLYB = DEFINE.OLTF
$DEFINE.POLYC = DEFINE.OLTF
$DEFINE.POLYD = DEFINE.OLTF
$DEFINE.POLYE = DEFINE.OLTF
$DEFINE.ONPOLY = DEFINE.OLTF
$DEFINE.ONPOLY = DEFINE.OLTF
$DEFINE.CNPOLY = DEFINE.OLTF
$DEFINE.CDPOLY = DEFINE.OLTF
$DEFINE.GNPOLY = DEFINE.OLTF
$DEFINE.GDPOLY = DEFINE.OLTF
$DEFINE.HNPOLY = DEFINE.OLTF
$DEFINE.HDPOLY = DEFINE.OLTF
: *****
: define the call routines for the matrix options under DEFINE
: *****
.DEFINE.MATA = DEFINE

```

FILE: MENU.TXT

```

.DEFINE.MATB = DEFINE
.DEFINE.MATC = DEFINE
.DEFINE.MATD = DEFINE
.DEFINE.MATE = DEFINE
; *****
; define the menu options under DISPLAY
; *****
$DISPLAY
OLTF
CLTF
GTF
HTF
GAIN*
BUTRWTH*
BESSEL*
EQUATION*
FREQ/RESP
LOCUS*
LOC/GAIN*
LOC/BRAN*
MATRIX
MODERN*
NICHOLS*
NYQUIST*
INVQUIST*
PAR/FRAC*
POLY
RICATTI*
ROUTH*
SPECS
SWITCHES*
TIME/RESP
HELP
! *****
; define the options for the DISPLAY.FREQ/RESP option
; *****
$DISPLAY.FREQ/RESP
OLTF
CLTF
GTF
HTF
TF1
TF2
TF3
TF4
TF5
HELP
!

```

FILE: MENU.TXT

```

: *****
: ; define the call routines for DISPLAY.FREQ/RESP
: *****
: DISPLAY.FREQ/RESP.OLTF = FREQ/RESP
: DISPLAY.FREQ/RESP.CLTF = FREQ/RESP
: DISPLAY.FREQ/RESP.GTF = FREQ/RESP
: DISPLAY.FREQ/RESP.HTF = FREQ/RESP
: DISPLAY.FREQ/RESP.TF1 = FREQ/RESP
: DISPLAY.FREQ/RESP.TF2 = FREQ/RESP
: DISPLAY.FREQ/RESP.TF3 = FREQ/RESP
: DISPLAY.FREQ/RESP.TF4 = FREQ/RESP
: DISPLAY.FREQ/RESP.TF5 = FREQ/RESP
: DISPLAY.FREQ/RESP.HELP = FREQ/RESP
: *****
: ; define the options for the DISPLAY.POLY option
: *****
: $DISPLAY.POLY
: ADD
: SUBTRACT
: POLYMLT
: SPOLYMLT
: POLYA
: POLYB
: POLYC
: POLYD
: POLYE
: ONPOLY
: ODPOLY
: CNPOLY
: CDPOLY
: GNPOLY
: GDPOLY
: HNPOLY
: HDPOLY
: HELP
: *****
: ; define the call routines for DISPLAY.POLY
: *****
: DISPLAY.POLY.POLYA = DISPLAY
: DISPLAY.POLY.POLYB = DISPLAY
: DISPLAY.POLY.POLYC = DISPLAY
: DISPLAY.POLY.POLYD = DISPLAY
: DISPLAY.POLY.POLYE = DISPLAY
: DISPLAY.POLY.ONPOLY = DISPLAY
: DISPLAY.POLY.ODPOLY = DISPLAY
: DISPLAY.POLY.CNPOLY = DISPLAY
: DISPLAY.POLY.CDPOLY = DISPLAY
: DISPLAY.POLY.GNPOLY = DISPLAY

```

FILE: MENU.TXT

```

.DISPLAY.POLY.GDPOLY = DISPLAY
.DISPLAY.POLY.HNPOLY = DISPLAY
.DISPLAY.POLY.HDPOLY = DISPLAY
.DISPLAY.POLY.ADD = DISPLAY
.DISPLAY.POLY.SUBTRACT = DISPLAY
.DISPLAY.POLY.POLYMLT = DISPLAY
.DISPLAY.POLY.SPOLYMLT = DISPLAY
.DISPLAY.POLY.HELP
: *****
; define the options for DISPLAY.MATRIX
: *****
$DISPLAY.MATRIX
ADD
SUBTRACT
MATXMULT
SCLRMULT
INVERSE
TRANPOSE
MATA
MATB
MATC
MATD
MATE
HELP
! *****
; define the call options for DISPLAY.MATRIX
: *****
.DISPLAY.MATRIX.ADD = DISPLAY
.DISPLAY.MATRIX.SUBTRACT = DISPLAY
.DISPLAY.MATRIX.MATXMULT = DISPLAY
.DISPLAY.MATRIX.SCLRMULT = DISPLAY
.DISPLAY.MATRIX.INVERSE = DISPLAY
.DISPLAY.MATRIX.TRANPOSE = DISPLAY
.DISPLAY.MATRIX.MATA = DISPLAY
.DISPLAY.MATRIX.MATB = DISPLAY
.DISPLAY.MATRIX.MATC = DISPLAY
.DISPLAY.MATRIX.MATD = DISPLAY
.DISPLAY.MATRIX.MATE = DISPLAY
.DISPLAY.MATRIX.HELP = DISPLAY
: *****
; define the options for the DISPLAY.TIME/RESP option
: *****
$DISPLAY.TIME/RESP
OLTF
CLTF
GTF
HTF
TF1

```

FILE: MENU.TXT

```

TF2
TF3
TF4
TF5
HELP
!
; *****
; define the call routines for DISPLAY.TIME/RESP
; *****
.DISPLAY.TIME/RESP.OLTF = TIME/RESP
.DISPLAY.TIME/RESP.CLTF = TIME/RESP
.DISPLAY.TIME/RESP.GTF = TIME/RESP
.DISPLAY.TIME/RESP.HTF = TIME/RESP
.DISPLAY.TIME/RESP.TF1 = TIME/RESP
.DISPLAY.TIME/RESP.TF2 = TIME/RESP
.DISPLAY.TIME/RESP.TF3 = TIME/RESP
.DISPLAY.TIME/RESP.TF4 = TIME/RESP
.DISPLAY.TIME/RESP.TF5 = TIME/RESP
.DISPLAY.TIME/RESP.HELP = TIME/RESP
; *****
; define the other call routines for DISPLAY
; *****
.DISPLAY.GTF = DISPLAY
.DISPLAY.HTF = DISPLAY
.DISPLAY.OLTF = DISPLAY
.DISPLAY.CLTF = DISPLAY
.DISPLAY.GAIN* = DISPLAY
.DISPLAY.BUTWRTH* = DISPLAY
.DISPLAY.BESSEL* = DISPLAY
.DISPLAY.EQUATION* = DISPLAY
.DISPLAY.HELP = DISPLAY
.DISPLAY.LOCUS* = DISPLAY
.DISPLAY.LOC/GAIN* = DISPLAY
.DISPLAY.LOC/BRAN* = DISPLAY
.DISPLAY.MODERN* = DISPLAY
.DISPLAY.NICHOLS* = DISPLAY
.DISPLAY.NYQUIST* = DISPLAY
.DISPLAY.PAR/FAC* = DISPLAY
.DISPLAY.RICATTI* = DISPLAY
.DISPLAY.ROUTH* = DISPLAY
.DISPLAY.SPECS = DISPLAY
.DISPLAY.SWITCHES* = DISPLAY
; *****
; define the menu options under FORM
; *****
.FORM = FORM
; *****

```

FILE: MENU.TXT

```

; define the menu options under HELP
; *****
$HELP
SYSTEM
FUNCTION
!
; *****
; define the menu option under HELP.FUNCTION
; *****
$HELP.FUNCTION
CHANGE
DISPLAY
MODIFY
RECOVER
SWITCHES
COPY
FORM
PRINT
STOP
UPDATE
DEFINE
HELP
!
; *****
; define the call routines for HELP options
; *****
.HELP.SYSTEM = HELP
.HELP.FUNCTION.CHANGE = HELP
.HELP.FUNCTION.COPY = HELP
.HELP.FUNCTION.DEFINE = HELP
.HELP.FUNCTION.DISPLAY = HELP
.HELP.FUNCTION.FORM = HELP
.HELP.FUNCTION.MODIFY = HELP
.HELP.FUNCTION.PRINT = HELP
.HELP.FUNCTION.RECOVER = HELP
.HELP.FUNCTION.STOP = HELP
.HELP.FUNCTION.SWITCHES = HELP
.HELP.FUNCTION.UPDATE = HELP
.HELP.FUNCTION.HELP = HELP
; *****
; define the menu options under MODIFY
; *****
$MODIFY
ADDRESS
DELROOT
CHANGE
HELP
!
; *****

```

FILE: MENU.TXT

AD-A164 844

DEVELOPMENT OF A COMPUTER AIDED DESIGN PACKAGE FOR
CONTROL SYSTEM DESIGN A. (U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..
S K HASHIKO ET AL. DEC 85

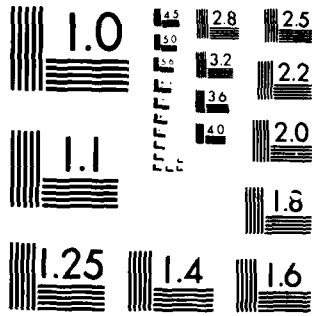
518

UNCLASSIFIED

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

```

; define the options under MODIFY.ADDROOT
; *****
$MODIFY.ADDROOT
POLYA
POLYB
POLYC
POLYD
POLYE
ONPOLY
ODPOLY
CNPOLY
GNPOLY
GDPOLY
HNPOLY
HDPOLY
!
; *****
; define the call routines for MODIFY.ADDROOT
; *****
.MODIFY.ADDROOT.POLYA = MODIFY
.MODIFY.ADDROOT.POLYB = MODIFY
.MODIFY.ADDROOT.POLYC = MODIFY
.MODIFY.ADDROOT.POLYD = MODIFY
.MODIFY.ADDROOT.POLYE = MODIFY
.MODIFY.ADDROOT.ONPOLY = MODIFY
.MODIFY.ADDROOT.ODPOLY = MODIFY
.MODIFY.ADDROOT.CNPOLY = MODIFY
.MODIFY.ADDROOT.GNPOLY = MODIFY
.MODIFY.ADDROOT.GDPOLY = MODIFY
.MODIFY.ADDROOT.HNPOLY = MODIFY
.MODIFY.ADDROOT.HDPOLY = MODIFY
; *****
; define the options under MODIFY.DELROOT
; *****
$MODIFY.DELROOT
POLYA
POLYB
POLYC
POLYD
POLYE
ONPOLY
ODPOLY
CNPOLY
GNPOLY
GDPOLY
HNPOLY

```

FILE: MENU.TXT

```
HDPOLY
!
: *****
: : define the call routines for MODIFY.DELROOT
: *****
:
: .MODIFY.DELROOT.POLYA = MODIFY
: .MODIFY.DELROOT.POLYB = MODIFY
: .MODIFY.DELROOT.POLYC = MODIFY
: .MODIFY.DELROOT.POLYD = MODIFY
: .MODIFY.DELROOT.POLYE = MODIFY
: .MODIFY.DELROOT.ONPOLY = MODIFY
: .MODIFY.DELROOT.OPPOLY = MODIFY
: .MODIFY.DELROOT.CNPOLY = MODIFY
: .MODIFY.DELROOT.CPPOLY = MODIFY
: .MODIFY.DELROOT.GNPOLY = MODIFY
: .MODIFY.DELROOT.GDPOLY = MODIFY
: .MODIFY.DELROOT.HNPOLY = MODIFY
: .MODIFY.DELROOT.HDPOLY = MODIFY
: *****
: : define the options under MODIFY.CHANGE
: *****
:
: $MODIFY.CHANGE
MATA
MATB
MATC
MATD
MATE
!
: *****
: : define the call routines for MODIFY.CHANGE
: *****
:
: .MODIFY.CHANGE.MATA = MODIFY
: .MODIFY.CHANGE.MATB = MODIFY
: .MODIFY.CHANGE.MATC = MODIFY
: .MODIFY.CHANGE.MATD = MODIFY
: .MODIFY.CHANGE.MATE = MODIFY
: *****
: : define the call for MODIFY.HELP
: *****
:
: .MODIFY.HELP = MODIFY
: *****
: : define the menu options under PRINT
: *****
:
$PRINT
OLTF*
CLTF*
GTF*
HTF*
GAIN*

```

FILE: MENU.TXT

```

BUTWRTH*
BESSEL*
EQUATION*
FRQ/RESP*
LOCUS*
LOC/GAIN*
LOC/BRAN*
MATRIX*
MODERN*
NICHOLS*
NYQUIST*
INVQUIST*
PAR/FRAC*
POLY*
RICATTI*
ROUTH*
SPECS*
SWITCHES*
TIM/RESP*
HELP
! : *****
: define the call routines for the PRINT options
: *****
.PRINT.GTF* = PRINT
.PRINT.HTF* = PRINT
.PRINT.OLTF* = PRINT
.PRINT.CLIF* = PRINT
.PRINT.GAIN* = PRINT
.PRINT.BUTWRTH* = PRINT
.PRINT.BESSEL* = PRINT
.PRINT.EQUATION* = PRINT
.PRINT.FRQ/RESP* = PRINT
.PRINT.LOCUS* = PRINT
.PRINT.LOC/GAIN* = PRINT
.PRINT.LOC/BRAN* = PRINT
.PRINT.MATRIX* = PRINT
.PRINT.MODERN* = PRINT
.PRINT.NICHOLS* = PRINT
.PRINT.NYQUIST* = PRINT
.PRINT.INVQUIST* = PRINT
.PRINT.PAR/FRAC* = PRINT
.PRINT.POLY* = PRINT
.PRINT.RICATTI* = PRINT
.PRINT.ROUTH* = PRINT
.PRINT.SPECS* = PRINT
.PRINT.SWITCHES* = PRINT
.PRINT.TIM/RESP* = PRINT
.PRINT.HELP = PRINT

```

FILE: MENU.TXT

```
.....  
; the call routines for the other options  
;.....  
-RECOVER = RECOVER  
-STOP = STOP  
-SWITCHES = SWITCHES  
-UPDATE = UPDATE  
;
```

FILE: MENU.TXT

```

1 1 Printer = false, Printer disabled
2 1 Trans = false, Transaction file disabled
3 1 Temp = false, Temporary file output disabled
4 0 Crt = true, Crt output enabled
5 1 show_abbreviation = false, not used
6 0 in_terminal = true, input from terminal enabled
7 0 stat_on = true, display status line enabled
8 1 macro_error = false
9 1 not used
10 1 not used
1 3 help_level
2 1 not used
3 0 "
4 0 "
5 0 "
6 0 "
7 0 "
8 0 "
9 0 "
10 0 "
1 1.0
2 -3.0
3 5.0
4 -1.0
5 0.0001
6 0.1
7 0.0
8 0.0
9 0.0
10 0.0
1 LST:
2 TRANSACT.ION
3 MACRO.INP
4 STRING 4
5 STRING 5
6 STRING 6
7 STRING 7
8 STRING 8
9 STRING 9
10 STRING 10
; List device name
; Transaction File name
; Macro Input File name
; description
; description
; description
; description
; description
; description
; description

```

FILE: PARAM.TXT

REFERENCES

1. Arnold, Abraham T. Further Development Of An Interactive Control Engineering Computer Analysis Package (ICECAP) For Discrete and Continuous Systems. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson Air Force Base, OH, December 1984.
2. Bullard, John. Interactive Computer Graphics for Analysis and Design of Control Systems. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson Air Force Base, OH, December 1984.
3. Cooper, Doug and Clancy, Michael. OH! PASCAL!. New York: W.W. Norton and Company., 1982.
4. CP/M-86 Version 1.1 for the IBM Personal Computer Operating System, Digital Research, Inc., Monterey, Ca., 1983.
5. D'Azzo, John J. and Constantine H. Houpis. Linear Control System Analysis and Design: Conventional and Modern (Second Edition). New York: McGraw-Hill Book Company, 1981.
6. Disk Operating System Microsoft Corporation, Boca Raton, Florida.
7. Fairly, Richard E. Software Engineering Concepts. New York: McGraw-Hill Book Company, 1985.
8. Fontana, Robert C., Professor Emeritis. Personal interview. AFIT, Wright-Patterson AFB OH, 24 Jul through present.
9. Freeman, Peter. "Fundamentals of Design," Tutorial on Software Design Techniques, Fourth Edition, edited by Peter Freeman and Anthony I. Wasserman. Silver Spring, MD: IEEE Computer Society Press, 1983.
10. Gembrowski, Charles J. Development of an Interactive Control Engineering Package (ICECAP) for Discrete and Continuous Systems. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson Air Force Base OH, December 1982.
11. IMSL Library User's Manual, IMSL, Houston, TX, 1984.
12. James, M.L., Smith, G.M., Wolford, J.C. Applied Numerical Methods For Digital Computation, New York: Harper & Row, 1977.

13. Larimer, Stanley J. An Interactive Computer-Aided Design Program for Digital and Continuous Control System Analysis and Synthesis. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson Air Force Base OH, December 1978.
14. Leong-Hong, Belkis W. and Bernard K. Plagman. Data Dictionary/Directory Systems. New York: John Wiley & Sons, 1982.
15. Logan, Glen T. Development of an Interactive Computer Aided Design Program for Digital and Continuous Control System Analysis and Synthesis. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson Air Force Base OH, December 1983.
16. Lorentz, Mark C., Graduate Student. Personal interview. AFIT, Wright-Patterson AFB OH, September 1985.
17. Moore, Paul A. Extension of the Software Development Workbench to Include Microcomputer Workstations. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson Air Force Base OH, December 1984.
18. Narathong, Chiewcharn. A Modern Control Theory Enhancement to an Interactive Control Engineering Computer Analysis Package (ICECAP). MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson Air Force Base OH, December 1984.
19. O'Brian, Frederick L. A Consolidated Computer Program for Control System Design. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson Air Force Base OH, December 1976.
20. Parisi, Vincent M. Development of a Computer Aided Design Package For Control System Design and Analysis for Use On a Personal Computer. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson Air Force Base OH, December 1983.
21. Shah, S.C. et al. MATRIXx User's Guide. Integrated Systems Inc., Palo Alto CA, 1982.
22. System Control Technology Inc. CTRL-C. Hawthorne CA, 1984.
23. Thompson, Peter M. User's Guide To Program CC, Version 3.0. Systems Technology, Inc., Hawthorne, California.

24. Travis, Mark A. Interactive Computer Graphics for System Analysis. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson Air Force Base OH, December 1983.
25. TURBO-Pascal Reference Manual Version 2.0 Borland International, Scotts Valley, California.
26. Wilson, Robert E. Continued Development of an Interactive Control Engineering Computer Analysis Package (ICECAP) for Discrete and Continuous Systems. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson Air Force Base OH, December 1983.
27. Woffinden, D.S., Professor. Personal interview. AFIT, Wright-Patterson AFB OH, 7 Nov 85.

VITA

Gary Tarczynski was born on 17 September 1957 in Chicago, Illinois. He graduated from the United States Air Force Academy in 1979 with a degree of Bachelor of Science in Electrical Engineering. Before arriving at the Air Force Institute of Technology (AFIT), he was stationed at Ellsworth AFB, SD, as a missile combat crew member and instructor.

Permanant Address: 7127 W. Lill St.
 Niles, Il 60648

Susan Mashiko was born on 22 July 1958 in Glendale, California. She graduated from the United States Air Force Academy in 1980 with a degree of Bachelor of Science in Aeronautical Engineering. Before arriving at the AFIT, she was stationed at Los Angeles AFS, CA, as a project engineer and branch chief.

Permanant Address: 6429 McAbee Rd
 San Jose, CA 95120

REPORT DOCUMENTATION PAGE

APR 16 1986

1a. NAME OF PERSON OR ORGANIZATION		5d. OFFICE SYMBOL <i>If applicable</i>		7a. NAME OF MONITORING ORGANIZATION									
School of Engineering		AFIT/ENG											
6c. ADDRESS (City, State and ZIP Code)		7b. ADDRESS (City, State and ZIP Code)											
Air Force Institute of Technology Wright-Patterson, AFB, Ohio 45433													
8a. NAME OF FUNDING SPONSORING ORGANIZATION		8b. OFFICE SYMBOL <i>If applicable</i>		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER									
8c. ADDRESS (City, State and ZIP Code)		10. SOURCE OF FUNDING NOS.											
		<table border="1"> <tr> <th>PROGRAM ELEMENT NO.</th> <th>PROJECT NO.</th> <th>TASK NO.</th> <th>WORK UNIT NO.</th> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> </table>				PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.				
PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.										
11. TITLE (Include Security Classification: See Box 19)													
12. PERSONAL AUTHOR(S): Susan K. Mashiko, B.S.A.E., Capt, USAF		Gary C. Tarczynski, B.S.E.E., Capt, USAF											
13. TYPE OF REPORT MS Thesis		13b. TIME COVERED FROM _____ TO _____		14. DATE OF REPORT (Yr., Mo., Day) 1985 December									
				15. PAGE COUNT 647									
16. SUPPLEMENTARY NOTATION													
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)										
FIELD	GROUP	SUB GR	MICROCOMPUTERS, COMPUTER AIDED DESIGN, COMPUTER PROGRAMS, CONTROL SYSTEMS, CONTROL SYSTEM DESIGN AND ANALYSIS, CONTROL THEORY										
	02												
19. ABSTRACT (Continue on reverse if necessary and identify by block number)													
<p>Title: Development of a Computer Aided Design Package for Control System Design and Analysis for a Personal Computer (ICECAP-PC)</p> <p>Thesis Chairman: Dr. Gary B. Lamont</p>													
20. DISTRIBUTION AVAILABILITY OF ABSTRACT			21. ABSTRACT SECURITY CLASSIFICATION										
UNCLASSIFIED UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS <input type="checkbox"/>			UNCLASSIFIED										
22a. NAME OF RESPONSIBLE INDIVIDUAL		22b. TELEPHONE NUMBER <i>(Include Area Code)</i>		22c. OFFICE SYMBOL									
Dr. Gary B. Lamont		513-255-3576		AFIT/ENG									

Approved for public release: LAW AFR 100-17
 E. WOLAYER 16 JAN 86
 Dean for Research and Professional Development
 Air Force Institute of Technology (AFIT)
 Wright-Patterson AFB OH 45433

This investigation developed a computer-aided design (CAD) package for control system design and analysis. The package was implemented on different varieties of small personal computers.

Structured design and other software engineering techniques were applied during the development effort. The program consists of a keyword-driven menu structure and a set of control system analysis procedures. The analysis procedures allow input of systems which are defined by transfer functions, polynomials, and matrices. Polynomials and matrices can be manipulated mathematically, and some block diagram manipulation can be performed on transfer functions as well.

The program is only part of a continuing development effort in the Information Sciences Laboratory at the Air Force Institute of Technology.

END

FILMED

4-86

DTIC