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       Gary C. Tarczynski
Captain, USAF           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.
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, getff,
get_unfact, poly, get_mean,
matrixmanipl, matrixmanip2, get_matrix_entries,
matrixadd, mmatrixmlt, get_mat,
matrixsub, inroot, delroot,
matrixinv, 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: matrixadd, matrixsub, mmatrixmlt,
smatrixmlt, 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, get_real, recover, make_pretty, update, get_fact, ccopyy, get_r_num, get_poly_name, ppoly, make_pretty_small_matrix, get_mat, disp_poly, make_pretty_large_matrix_one, inroot, delroot, disp_matrix, mmatrix

Name: b
Aliases: None
Type Of Entry: Variable
Description: This variable is used as a temporary storage area the polynomial.
Make up: bl
Source: Destination: Used In: roots

Name: bl
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, pppoly, delroot

Name: bld_stat_line
Type: Procedure
Description: This procedure builds the status line from initialization data from disk storage. 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

FILE: DATA DICTIONARY D-5
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, matrxmlt, smatrxmlt, matrxtran

Name: bpoly
Aliases: None
Type Of Entry: Variable
Description: This is a polynomial.
Make_up: polynomial
Source: Destination:
Used In: spolymlt, pulymlt, 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 for the polynomial.
Make_up: cl
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: cl
Aliases: None
Type Of Entry: Variable
Description: This variable is used as a temporary storage area for 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: ccopyy
Type: Procedure
Description: This procedure manages the copy function. It gets the
source and destination location then performs the
copy operation. (called ccopyy 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,

FILE: DATA DICTIONARY
Called By: disp_msg
Called By: 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: form
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: get_chr, get_int, ck_chr, get_real
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: help
Used In: help

*****************************************************************************
Name: check

FILE: DATA DICTIONARY D-9
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

FILE: DATA DICTIONARY D-10
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 column 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_string, ucase, disp_matrix, 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

Type Of Entry: Variable

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

Make up: cmdword

Source: Destination:

Used In: chgmat, inroot, delroot, poly_into_storage, form, disp_matrix, poly_from_storage, disp_poly
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. It 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

FILE: DATA D ICTIONARY D-12
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_matrix,
ppoly, mmatrix, polymlt,
define, get_poly, mmatrixmlt

Version: 2.0
Date: 21 Oct 83
Author: Vincent M. Parisi II, Capt, USAF

Name: clsir 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

FILE: DATA DICTIONARY
Files Written: None
Aliases: None
Procedures Called: clear, out_string
Called By: disp_msg, procEs_error, recover, update, get_real, get_mat, get_matrxmlt, chgmat, 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

******************************************************************************

FILE: DATA DICTIONARY   D-14
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmd</td>
<td>This is a string of char.</td>
</tr>
<tr>
<td>cmdbuffer</td>
<td>This is a buffer of individual commandwords input by the user.</td>
</tr>
<tr>
<td>cmd_col</td>
<td>This is the column that is used as the beginning of user entered commands.</td>
</tr>
<tr>
<td>cmd_len</td>
<td>This variable is the length of the command.</td>
</tr>
</tbody>
</table>

**FILE: DATA DICTIONARY D-15**
**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, disp_poly, modify, disp_matrix, poly_into_storage, matrixmanip2, 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 Alias: None Type Of Entry: variable Description: The variable is a temporary storage area for the polynomial.
Make_up: array[1..maxdeg1] of real
Source: Destination: Used In: roots

Name: col Alias: 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: promptcmd, get_r_num, make_pretty_large_matrix_one, disp_matrix, chgmat, get_poly_name, make_pretty_small_matrix, get_matrix_entries, gotoxy

Name: col_element Alias: 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_matrix, chgmat, get_matrix_entries

Name: column Alias: None Type Of Entry: Variable

FILE: DATA DICTIONARY D-17
Description: This variable is the column location for the left side of the rectangle.

Make up: Integer
Source:
Destination: rectangle
Used In:

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: spolymlt
Used In:

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: crtonly
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:
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 msdwtpe
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
******************************************************************************

FILE: DATA DICTIONARY  D-21
**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

FILE: DATA DICTIONARY  D-22
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: selectroutine

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

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

FILE: DATA DICTIONARY D-23
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: degree
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: ck_chr
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, disp_poly, 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

FILE: DATA DICTIONARY D-25
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

FILE: DATA DICTIONARY           D-26
Name: denom_deg
Aliases: None
Type Of Entry: Variable
Description: This variable is the degree of the transfer function's denominator.
Make up: Integer
Source:
Destination: gettf
Used In: gettf

Name: denominator
Aliases: None
Type Of Entry: Variable
Description: This variable is the polynomial that is the denominator of the transfer function.
Make up: Polynomial
Source:
Destination: gettf, disptf
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:** ccopyy

---

**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, ccopyy

---

**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:**

```plaintext
dict_buffer = record
    ptrs : array[1..num_ptrs] of ptr_recs;
    words : array[1..num_words] of string[word_length];
    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:**

```plaintext
dictionary = record
    dictword : string[wordsize];
    matchp : integer;
    nomatchp : integer;
end;
```

**Source:** Declared in icecappc

**Destination:** N/A

**Used In:** icecappc, check_word, get_line

---

**FILE:** DATA DICTIONARY . D-28
Name: dimension
Aliases: None
Type Of Entry: Variable
Description: This variable is the dimension of the matrix.
Make_up: Integer
Source:
Destination:
Used In: matrixinv

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_matrix
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: matrixmanip1, matrixmanip2, 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

FILE: DATA DICTIONARY  D-31
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, clear_msg, proces_error, recover, update, ccopyy, help, make_pretty, get_real, get_fact, roots, gettf, define, mmatrix, getmat, mattrxmlt, matrxadd, modify, matrxinv, disptf, delroot, form, disp, ppoly, get_poly, disppoly, disp, get_poly_name, get_matrix_name

Called By: proces_error, recover, update, ccopyy, help, make_pretty, get_real, get_fact, roots, gettf, define, mmatrix, getmat, mattrxmlt, modify, matrxadd, matrxinv, disptf, delroot, form, disp, ppoly, get_poly, disppoly, 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: gettf
Used In: gettf

******************************************************************************

******************************************************************************

Name: DISP.PAS

FILE: DATA DICTIONARY D-32
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: polymannip, polymannip2, ppol, 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

FILE: DATA DICTIONARY D-33
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: matrixinv
**********************************************************************

**********************************************************************
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:
**********************************************************************

**********************************************************************
Name: d_len

FILE: DATA DICTIONARY    D-34
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

FILE: DATA DICTIONARY D-35
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: 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 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

FILE: DATA DICTIONARY   D-36
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, polymlt, 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: maxdegI
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

FILE: DATA DICTIONARY D-38
Type:  File
Description: This file will form OLTIF'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:  form
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:  form
Used In:  form

Name:  getch
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:  

FILE:  DATA DICTIONARY  D-39
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, val_n_dec, prompt_help, displa_commandword, prompt_cmd, instruction, proces_error, clear

Called By: icecappc

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

FILE: DATA DICTIONARY  D-40
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, transact.ion, 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 title slide.

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

FILE: DATA DICTIONARY  D-41
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_1st_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_string
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
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

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

FILE: DATA DICTIONARY  D-44
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

Name: get_matrix_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, as_assigned, crt_only
Global Constants Used: None
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_string, disp_msg, pause
Called By: mmatrix

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

Version: 1.0
Date: 11 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
Contained In File: GETMAT.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_matrix_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, disp_poly, trim, pause
Called By: define
Version: 1.0
Date: 28 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

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 an inadvertent exit to the operating system.

Global Variables Used: abort_command, strng
Global Variables Changed: abort_command, strng
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

******************************************************************************

******************************************************************************

FILE: DATA DICTIONARY D-47
Name: get_num
Type: Procedure
Description: This procedure gets a real number from the user. The procedure checks the validity of the input number. This prevents 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_matrix_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. Tarzynski, Capt, USAF
Contained In File: GETTF.PAS

******************************************************************
******************************************************************

Name: GETSTRING.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_string
Version: 2.0
Date: 28 Aug 83
Author: Vincent M Parisi II, Capt, USAF

******************************************************************

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

FILE: DATA DICTIONARY D-48
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_matrix_name

Version: 2.0
Date: 28 Aug 83
Author: Vincent M. Parisi II, Capt, USAF

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

FILE: DATA DICTIONARY D-49
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

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

FILE: DATA DICTIONARY D-50
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,
title_slide,
prompt_cmd,
get_real,
update,
roots,
get_unfact,
getmat,
matrix,
form,
polymlt,
get_poly,
make_pretty

Files Read: None
Files Written: None
Aliases: None
Procedures Called: ttype
Called By:
clear,
title_slide,
prompt_cmd,
get_real,
update,
roots,
get_unfact,
getmat,
matrix,
form,
polymlt,
get_poly,
make_pretty

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
********************************************************************************

FILE: DATA DICTIONARY D-51
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: 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
Type Of Entry: Variable
Aliases: None
Description: Variable is a polynomial, denominator of the feedback transfer function.
Make_up: Polynomial
Source:
Destination:
Used In: form

Name: height
Type Of Entry: Variable
Aliases: None
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

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

FILE: DATA DICTIONARY D-53
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 initialization.
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:
highlight, title_slide, prompt_cmd
process_error, get_r_num, ccopyy,
get_real, roots, delroot,
polymlt, form, get_matrix_name

Called By: hnpol

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, video low,
svideo low, vvideo bold, svideobold,
rectangle, disp_msg, clear_msg,
trim, readCom, check_word,
get_cmd, get_data, recover,
ccopyy, make_pretty, update,
movetf, move_poly, move_matrix,
get_real, gettf, get_fact,
get_unfact, roots, left_bracket,
matrxsub, right_bracket, get_matrix_entries,
getmat, matrxadd, mmatrxmul,
polymanip, polymanip2, dispmatrx,
matrixmanipl, polysub, smatrxmul,
matrxmanip2, matrxtran, matrxinv,
disppoly, get_poly, disptf,
chgmat, inroot, delroot,
spolymlt, polymlt, polyadd,
make_pretty_small_matrix, select_routine,
make_pretty_large_matrix_one, process_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

*****************************************************************************

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: 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, disp_poly, 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

Got to 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

******************************************************************************

FILE: DATA DICTIONARY D-57
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: instring
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, svideo-low

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

This file issues the appropriate instructions for entering a command based on the number of command words already entered.

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: 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: roots
Used In: roots

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

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: mmatrxmlt

Name: Ll
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.

FILE: DATA DICTIONARY
Make_up: Integer
Source: 
Destination: 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
Procedures 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

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

FILE: DATA DICTIONARY D-61
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: get_cmd, instruction, val_n_dec
Used In: 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: retangle

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

FILE: DATA DICTIONARY D-62
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: macroerror
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.

FILE: DATA DICTIONARY D-63
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
**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

**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:** getfact, get_unfact, disp_tf, disp_poly

---

**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

**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_matrix, chgmat, get_matrix_entries

---

FILE: DATA DICTIONARY D-65
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_matrix, 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_matrix, 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:

FILE: DATA DICTIONARY D-66
Used In:  recover,  update,  chgmat,  disp_matrix

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:  matrxmanipl,  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:  matrxmanipl,  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:  matrxmanipl

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

FILE:  DATA DICTIONARY  D-67
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: matrixmanip2, 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

FILE: DATA DICTIONARY    D-68
**Description:** This is the record definition for a file of matrix.

**Make-up:**

```pascal
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, matrixmanipl, matrixmanip2, disp_matrix, matrixtran, matrixinv, chgmat, get_matrix_entries, matrixsub, smatrixmlt, getmat, matrixadd, mmmatrixmlt

---

**Name:** MATRIX.PAS
**Type:** File
**Description:** This file will decode the command string from define.pas and call the appropriate procedures.
**Procedures Contained:** disp_matrix, matrix_manipl, matrix_manip2, get_matrix_name, mmatrix

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

---

**Name:** matrx
**Aliases:** None
**Type Of Entry:** Variable
**Description:** This is a matrix.

**Make-up:** matrix

**Source:**

**Destination:** move_matrix

**Used In:**

---

**Name:** MATRXMAN.PAS
**Type:** File
**Description:** This file contains the matrix manipulation procedures.
**Procedures Contained:** matrixadd, mmmatrixmlt, matrixsub, smatrixsub, matrixtran, matrixinv

**Version:** 1.0
**Date:** 22 Sep 85
**Author:** Susan K. Mashiko, 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: matrx manip
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_matrix, matrxadd, mmatrixmlt, matrxsub
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: matrx 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_matrix, matrxinv, smatrixmlt, matrxtran
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: matrxsub

FILE: DATA DICTIONARY D-71
Type: Procedure
Description: This procedure will subtract the second matrix from 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: matrxadd
Called By: matrxmanipl

Version: 1.0
Date: 20 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

-contained In File: MATRXMAN.PAS

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

FILE: DATA DICTIONARY D-72
Makeup: file of matrix
Source:
Destination:
Used In: recover, update, matrxmanipl, matrxmanip2, disp_matrix, 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_degl
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.

FILE: DATA DICTIONARY D-73
Make up: Integer
Source: Declared in concons
Destination: matrix, getmat
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: 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, matrixmanip2, get_matrix_name, get_r_num, gotoxy, out_string, disp_matrix, trim, disp_msg, matrixmanipl
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: mmatrixmlt
Type: Procedure
Description: This procedure will multiply the second matrix to 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

FILE: DATA DICTIONARY D-75
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: 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
Modified 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: 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
Modified 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: 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
Modified 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 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: 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

FILE: DATA DICTIONARY D-78
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: 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_string, recover, update, ck_chr, get_poly_name, svideobold, svideo low

******************************

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

FILE: DATA DICTIONARY D-80
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:

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: 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, ccopyy, get_r_num, roots, delroot, form, polymlt, get_matrix_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_r_num, get_fact,  
- get_unfack, spolymlt, matrixmanip2,  
- ppolyn, dispipmx, polymanip2,  
- smatrixmlt, get_matrix_entries, disptf  
- mmatrix, getmat, chgmat,  
- disp_poly, inroot

---

**FILE: DATA DICTIONARY**  
D-84
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_matrix, chgmat, getmat, make_pretty_large_matrix_one, get_matrix_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

FILE: DATA DICTIONARY D-85
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 msdwcs
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 msdwcs
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

FILE: DATA DICTIONARY D-87
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_matrix, chgmat, getmat,
make_pretty_large_matrix_one, get_matrix_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

FILE: DATA DICTIONARY    D-89
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, transact.ion, printer.out
Aliases: None
Procedures Called:
Called By: make_pretty, inroot, make_pretty_large_matrix_one,
make_prettysmall_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, transact.ion
Aliases: None
Procedures Called:
get_fact, get_unfact, get_r_num,
disptf, chgmat, inroot,
disppoly, disp_matrix
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 program. 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, transact.ion, temp.out
Aliases: None
Procedures Called:
Called By:

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

*****************************
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 throughout the program.
Make_up: paramstring = string[ 14 ]
Source: Declared in msdwtype
Destination: 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, copyy, help, get_real, get_fact, matrixadd, get_matrix_entries, matrxinv, define, modify, chgmat, dispmatrx, inroot, delroot, mmatrix, polymanip, polymanip2, get_matrix_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:

FILE: DATA DICTIONARY D-94
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.

UNCLASSIFIED  S K MASHIKO ET AL. DEC 85

F/G 9/2  ML
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: recover, update

Name: pol_deg
Aliases: None
Type Of Entry: Variable
Description: This is the degree of the polynomial.
Make_up: Integer
Source: 
Destination: get_poly
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: recover, update
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

FILE: DATA DICTIONARY... D-96
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
Passed Variables: choice, pol
Returned: None
Files Read: TF&POLS.DAT
Files Written: None
Aliases: None
Procedures Called: trim
Called By: form

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, polymlt, 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

FILE: DATA DICTIONARY
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

FILE: DATA DICTIONARY       D-99
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..maxdeg1] of real;
end;
Source: Declared in concons
Destination: 
Used In: poly, roots, get_unfact, get_poly,
form_poly, get_fact, move_tf, polyman:p,
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

FILE & DATA DICTIONARY - D-100
**********************************************************************
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: 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
Containing In File: POLYMAN.PAS
**********************************************************************

FILE: DATA DICTIONARY D-101
**Name:** pos  
**Aliases:** None  
**Type Of Entry:** Variable  
**Description:** This a the passed position.  
**Make_up:** Integer  
**Source:***  
**Destination:**  
**Used In:** svideo1ow, svideo1obold

**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, dispoly, 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 the column location that the logo appears at.  
**Make_up:** Integer  
**Source:** get_cmd  
**Destination:**  
**Used In:** get_cmd

---

FILE: DATA DICTIONARY   D-102
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

FILE: DATA DICTIONARY D-103
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

FILE: DATA DICTIONARY  D-104
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 is 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

FILE: DATA DICTIONARY .. D-105
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

FILE: DATA DICTIONARY D-106
**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:** PROMPTH.E.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, getline, 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:** PROMPTH.E.PAS

---

**FILE:** DATA DICTIONARY  
**D-107**
Name: ptr_rec
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[ l..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

FILE: DATA DICTIONARY D-108
Aliases: None
Procedures Called: get_string, 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: S
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.

FILE: DATA DICTIONARY D-109
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,

FILE: DATA DICTIONARY  D-110
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: 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&pol.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

******************************************************************************

FILE: DATA DICTIONARY       D-111
**remain_lines**

- **Type Of Entry:** Variable
- **Description:** Counter of the total number of lines remaining in the selected message.
- **Make-up:** Integer
- **Used In:** disp_msg

**repeat1**

- **Type Of Entry:** Label
- **Description:** Used for goto statement.
- **Make-up:** Char
- **Used In:** get_unfact, get_fact, recover, update

**repeat2**

- **Type Of Entry:** Label
- **Description:** Used for goto statement.
- **Make-up:** Char
- **Used In:** recover, update

**repeatagain**

- **Type Of Entry:** Label
- **Description:** Used for goto statement.
- **Make-up:** Char
- **Source:**
- **Destination:**

---

**FILE: DATA DICTIONARY**

D-112
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

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, polymlt, polyadd, spolymlt

Version: 2.0
Date: 6 Sep 85
Author: Susan K. Mashiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF

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_matrix,
get_poly_name, disp_poly, get_matrix_entries,
disp_tf, 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, process_error, make_pretty,
gttf, 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.

FILE: DATA DICTIONARY
**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

***************************************************************************

***************************************************************************

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

FILE: DATA DICTIONARY   D-116
**Show Abbreviation**

*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:* icecappc, get_data

**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:* matrxmanip2

*Version:* 1.0

*Date:* 20 Sep 85

*Author:* Susan K. Mashiko, Capt, USAF

Gary C. Tarczynski, Capt, USAF

*Contained In File:* MATRXMAN.PAS

**Source Loc**

*Type Of Entry:* Variable

*Description:* This variable is integer value of the storage location of the source for a copy.

*Make up:* Integer

*Source:* **FILE: DATA DICTIONARY** D-117
**Source**

**Destination:**

**Used In:** ccopyy, 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:** ccopyy

**Spolymlt**

**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

**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
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: None
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: N/A
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 Icecappc
Destination:  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:  matrixinv

*************************************************************************************

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

*************************************************************************************

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

FILE:  DATA DICTIONARY  D-120
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: Integer
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, matrixmanipl, matrixmanip2, get_poly, getmat, inroot, delroot, disppoly, disp_matrix, dispft, 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

FILE: DATA DICTIONARY D-122
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 temporary 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
*******************************************************************************

FILE: DATA DICTIONARY   D-124
Name: temppol
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

FILE: DATA DICTIONARY D-125
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, clearscren, 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, nohighlight, gotoxy, clear, clearscren, videolow, svideolow, videobold,

FILE: DATA DICTIONARY D-126
FILE: DATA DICTIONARY D-127

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: icecappc, out_string, bld_stat_line, out_int, get_data, out_real

Name: TRANSACTION
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: 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: 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: wordsiz-
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, matrixmanip1, matrixmanip2, define, disp_matrix, ppolypoly, chgmat, inroot, delroot, polymanip, polymanip2, get_poly_name, dispnpoly, 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

FILE: DATA DICTIONARY     D-129
**Name:** tword
**Aliases:** None
**Type Of Entry:** Variable
**Description:** This variable is used as 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, ccopyy

**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

**FILE:** DATA DICTIONARY  D-130
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_matrix_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

FILE: DATA DICTIONARY D-131
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

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, getline, 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

FILE: DATA DICTIONARY D-133
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

FILE: DATA DICTIONARY D-135
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:       displaya_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:
Used In:       icecappc, get_line, trim, val_n_dec, displaya_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:
**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
builddat

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

MNUCONS.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
  readme
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
ccopyy

** 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_strng  

** 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_matrix  
   matrix_manipl  
   matrix_manip2  
   get_matrix_name  
   mmatrix

** MATRIXMAN.PAS 
   matrxadd  
   matrxsub  
   mmatrixxmlt  
   smatrixxmlt  
   matrxtran  
   matrxinv

** MODIFY.PAS 
   chgmat  
   modify

MSDWCONS.PAS - MICROSDW.SYS Constant Def. File

MSDWTYPE.PAS - MICROSDW.SYS Type Def. File

MSG.PAS 
   disp_line  
   clear_msg  
   disp_msg

OUTPUT.PAS 
   out_string

PAUSE.PAS 
   pause

E-6
** POLYMAN.PAS
polyadd
polysub
polymlt
spolymlt

** POLY.PAS
disppoly
polmanip
polmanip2
get_poly_name
ppoly

PROCESER.PAS
proces_error

PROMPTCM.PAS
prompt_cmd

PROMPTHE.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

E-7
SVideobold
Rectangle

TRIM.PAS
trim

UCASE.PAS
ucase

** UPDATE.PAS
update

VALNDEC.PAS
check_word
val_n_dec

E-8
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: select routine
author: Gary C. Tarczynski, Capt. USAF
         Susan K. Mashiko, Capt. USAF

procedure frequency_response;
var
  i : integer;

begin
  ClearScreen;

FILE: BODE.PAS
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: DOJE.PAS
**file**: CONCONS.PAS

**version**: 1.0

**date**: 26 August 85

**description**: Contains the constants, type, and var declarations for the control procedures and functions.

**author**: Susan K. Mashiko, Capt, USAF

Gary C. Tarczynski, Capt, USAF

---

```pascal
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, copy
**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.
**author:** vincent m. parisi ii, capt, usaf
Gary C. Tarczynski, Capt, USAF
Susan K. Mashiko, Capt, USAF

```pascal
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: copy
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;
```

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;

******************************************************************************

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
write( tf_file, poly );
seek( tf_file, ( sorce_loc + 1 ) );
read( tf_file, poly );
seek( tf_file, ( dest_loc + 1 ) );
write( tf_file, poly );
close( tf_file );
end;

*******************************************************************************

procedure: move_poly
version: 2.0
date: 04 September 1985
description: This procedure receives the source and destination polynomial
locations and performs the copy.
passed variables: sorce_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 MF to TURBO Pascal
mod date: 04 September 1985
*******************************************************************************

overlay procedure move_poly( sorce_loc: integer; dest_loc: integer );
var	poly_file : file of polynomial;
	poly : polynomial;
	l : integer;

begin
assign( poly_file, 'TF&POLS.DAT' );
reset( poly_file );
seek( poly_file, sorce_loc );
read( poly_file, poly );
seek( poly_file, dest_loc );
write( poly_file, poly );

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
called by: ccopy
author: vincent m. parisi ll, capt. usaf
modified by: Gary C. Targzynski, Capt., USAF
            Susan K. Masniko, 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: ccopyy (ie copy, called ccopyy due to pascal compiler having internal function called copy)
version: 2.0

date: 22 September 1985
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

procedures called: clear, gotoxy, trim, out_string,
get_location, move_matrix,
move_poly, move_tf, highlight,
onhighlight, pause, disp_msg
called by: select_routine
author: vincent m. partisi ii, capt, usaf
modified by: Gary C. Tarczynski, Capt, USAF
Susan K. Mashiko, Capt, USAF

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 ccopyy( var cmdbuffer : buffer );

var

1 : integer;
source : cmdword;
destination : cmdword;
type_move : char;
dest_loc : integer;
sorce_loc : integer;

(****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** ***)

begin

gotoxy(15,30);
source := cmdbuf[2];
trim( source );
if source = 'HELP' then
  begin
    clear;
    disp_msg( 17 );
pause;
clear;
    exit;
  end
else
destination := cmdbuf[3];
out_string( 'Copying ', as_assigned );
out_string( source, as_assigned );
out_string( destination, as_assigned );
(**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, source_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 *)
if type_move1 <> type_move2 then
  begin
    gotoxy(16,20);
    highlight;
    out_string( 'Mismatched types. Copy not performed.', as_assigned );
    pause;
    exit;
  end
else

FILE: COPY.PAS
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 );
pol.polydeg := pol_deg;
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 = 'POLYB' then stor_loc := 19
  else
    if def_obj = 'POLVC' then stor_loc := 20
    else
      if def_obj = 'POLYD' then stor_loc := 21
      else
        if def_obj = 'POLYE' then stor_loc := 22
        else
          if def_obj = 'ONPOLY' then stor_loc := 0
          else
            if def_obj = 'DOPOLY' then stor_loc := 1
          end
        end
      end
    end
  end
end

FILE: DEFINE.PAS
else
if def_obj = 'CNPOLY' then stor_loc := 2
else
if def_obj = 'DNPOLY' then stor_loc := 3
else
if def_obj = 'GNPOLY' then stor_loc := 4
else
if def_obj = 'DNPOLY' then stor_loc := 5
else
if def_obj = 'HNPOLY' then stor_loc := 6
else
if def_obj = 'HNPOLY' then stor_loc := 7;
(* save polynomial to the file *)
assign( polys, 'tf&pols.dat' );
reset( polys );
seek( polys, stor_loc );
write( polys, poly );
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, pause,
get_tf, get_mat,
get_poly, disp_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
var
def_obj : cmdword;
check : cmdword;
begin
def_obj := cmdbuffer[2];
trim( def_obj );
check := cmdbuffer[3];
trim( check );
if check = 'HELP' then begin
  clear;
disp_msg( 28 );
pause;
end;
end;

(* The following code calls the transfer function input procedure *)
if ( (def_obj = 'OLTF') or (def_obj = 'CLTF') or (def_obj = 'GTF') or
    (def_obj = 'MTF') or (def_obj = 'TF1') or (def_obj = 'TF2') or
    (def_obj = 'TF3') or (def_obj = 'TF4') or (def_obj = 'TF5')) then
  get_tf( def_obj, cmdbuffer[3] );

(* Code for the help function *)
if def_obj = 'HELP' then begin
  clear;
disp_msg( 18 );
pause;
end;
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, cmdbuffer[ 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, cmdbuffer[ 3 ] );

(*DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***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 handled 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, cmdbuffer[ 3 ] );

*DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE*)

end;

FILE: DEFINE.PAS
overlay procedure delroot( var cmdbuffer : buffer;
                        var wordnumber : integer );

label once_more;

FILE: DELROOT.PAS
var
cm
"polys " : file of polynomial;
i : integer;
onpolynomials : polynomial;
newpoly : polynomial;
stor_loc : integer;
root_num : integer;
begin

FILE: DELROOT.PAS
gotoxy(20,0);
out_string( blanks, crt_only);
gotoxy(20,5);
nhighlight;
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[1].realpart :=
    oldpoly.polyfact[1].realpart;
    newpoly.polyfact[1].imagpart :=
    oldpoly.polyfact[1].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;
    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[1].realpart := oldpoly.polyfact[1].realpart;
  newpoly.polyfact[1].imagpart :=
  oldpoly.polyfact[1].imagpart;
end;
if root_num >= (oldpoly.polydeg - 2) then
begin
  for i := (root_num - 1) to (oldpoly.polydeg - 2) do
  begin
    newpoly.polyfact[1].realpart :=
    oldpoly.polyfact[i + 2].realpart;
    newpoly.polyfact[1].imagpart :=
    oldpoly.polyfact[i + 2].imagpart;
  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;
  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;
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
procedure displayf
version: 1.0
date: 25 September 1985
description: This procedure will display the requested transfer function on the screen. The displayed function 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 displayf; (* forward referenced from gettf *)
(* ( var disp_obj : cmdword ); *)

var
stor_loc : integer;
numerat : polynomial;
denominat : polynomial;
polys : file of polynomial;
l : 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&pols.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 );
1 := 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], i, realpart, 12, as_assigned);
   gotoxy( (row + 3 + i), 59);
   out_real( numerator.polyfact[i], i, imagpart, 12, as_assigned);
   i := i + 1;
end; (* end of while 'loop *)

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

FILE: DISP.PAS
gotoxy( (row + 3 + 1), 43);
out_real( denominator.polyfact[1].realpart, 12, as_assigned);
gotoxy( (row + 3 + 1), 59);
out_real( denominator.polyfact[1].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 + 1), 7);
  out_real( denominator.polypoly[1].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: dispft, 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 *)

FILE: DISP.PAS
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
    poly( 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;

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 display_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
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 = 'POLYC' then stor_loc := 20
         else
            if choice = 'POLYD' then stor_loc := 21
   end

FILE: FORM.PAS
else
  if choice = 'POLYE' then stor_loc := 22
else
  if choice = 'ONPOLY' then stor_loc := 0
else
  if choice = 'QOPOLY' then stor_loc := 1
else
  if choice = 'CNPOLY' then stor_loc := 2
else
  if choice = 'COPOLY' then stor_loc := 3
else
  if choice = 'ONPOLY' then stor_loc := 4
else
  if choice = 'GOPOLY' then stor_loc := 5
else
  if choice = 'HNPOLY' then stor_loc := 6
else
  if choice = 'HDPOLY' then stor_loc := 7;
assign( polys, 'tf&polis.dat' );
reset( polys );
seek( polys, stor_loc );
read( polys, pol );
close( polys );
end;

procedure poly_into_storage
version: 1.0
date: 10 October 85
description: This file contains the procedures to place a polynomial into storage
passed variables: choice, pol
returned variables: pol
files written: "TF&POLIS.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: FORM.PAS
file of polynomial;
integer;

if choice = 'POLY A' then stor_loc := 18
else if choice = 'POLY B' then stor_loc := 19
else if choice = 'POLY C' then stor_loc := 20
else if choice = 'POLY D' then stor_loc := 21
else if choice = 'POLY E' then stor_loc := 22
else if choice = 'ON POLY' then stor_loc := 0
else if choice = 'OD POLY' then stor_loc := 1
else if choice = 'CN POLY' then stor_loc := 2
else if choice = 'CD POLY' then stor_loc := 3
else if choice = 'CN POLY' then stor_loc := 4
else if choice = 'CD POLY' then stor_loc := 5
else if choice = 'CN POLY' then stor_loc := 6
else if choice = 'HD POLY' then stor_loc := 7;

assign( polys, 'tf&pols.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 OLF's and DLF's
global variables used: abort_command
global constants used: crt_only

FILE: FORM.PAS
procedure form:

label
abort, abort2, repeat_again;

var
  selection : integer;
  gnpol, hnpol : polynomial;
  cnpol, cdpol : polynomial;
  templo, templo : 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
      ...
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, gpol);
    choice := 'HNPOLY';
    poly_from_storage(choice, hpol);
    choice := 'DPOLY';
    poly_from_storage(choice, dpol);
    choice := 'HPOLY';
    poly_from_storage(choice, hpol);

    (* check to see if the resulting tf will have a degree *)
    (* greater than 10 *)
    if (gpol.polydeg + hpol.polydeg) > 10 then
      goto abort;
    if (gpol.polydeg + hpol.polydeg) > 10 then
      begin
        abort;
        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 10th order');
        exit;
      end;
    else
      begin
        polymit(gpol, hpol, onpol);
        polymit(dpol, hpol, 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');
      end;

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.coef := gain * ( onpol.coef / odpol.coef
odpol.coef := 1.0; )
choice := 'ONPOLY';
poly_into_storage( 'choice, onpol );
choice := 'OPPOLY';
poly_into_storage( 'choice, odpol );
choice := 'OLTP';
clear;
disptf( 'choice ' );
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 := 'OPPOLY';
poly_from_storage( 'choice, oppol ' );
    choice := 'MPOLY';
poly_from_storage( 'choice, mpol ' );
    (* check to see if the resulting tf will have *)
    (* a degree greater than 10 *)
    if ( gnpol.polydeg + hnpol.polydeg ) > 10 then
go to abort;
    if ( gnpol.polydeg + hnpol.polydeg ) > 10 then
begin
    abort;
clear;
gotoxy( 8, 10 );
highlight:
writeln(' Degree of result greater than 10, option aborted ');
highlight:
writeln(' Due to the storage space limitations your
resulting ');

FILE: FORM.PAS
writeln(' polynomial is limited to 10th order');
exit;
end
else
begin
polymtl( 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;
spolymtl( cnpol, temppol, gain );
polymtl( gdpol, hdpol, cnpol );
polyadd( cnpol, temppol, cdpol );
polymtl( gnpol, hnpol, temppol );
spolymtl( temppol, cnpol, gain );
choice := 'CNPOLY';
poly_into_storage( choice, cnpol );
choice := 'CDPOLY';
poly_into_storage( choice, cdpol );
choice := 'CLTF';
clear;
disptf( choice );
end;

(* selection 3 forms the CLTF from the QLTF *)
if selection = 3 then
begin
choice := 'CNPOLY';
poly_from_storage( choice, onpol );
choice := 'ODPOLY';
poly_from_storage( choice, odpol );
end;
(* 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( cnpol, cnpol, gain );
polyadd( cnpol, dnpol, cdpol );
choice := 'CNPOLY';
poly_into_storage( choice, cnpol );
choice := 'CDPOLY';
poly_into_storage( choice, cdpol );
choice := 'CLTF';
clear;
disptf( 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, temppol );

(* 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;
cnpoly.coefficient := gain * ( cnpoly.coefficient / cdpoly.coefficient );
cdpoly.coefficient := 1.0;

gain := 1;
polyadd( gdpoly, hnpoly, cdpoly );
polyadd( temppoly, cdpoly, cnpoly );
polyadd( gdpoly, hdpoly, cdpoly );

choice := 'CNPOLY';
poly_into_storage( choice, cnpoly );
choice := 'CDPOLY';
poly_into_storage( choice, cnpoly );
choice := 'CLTF';
clear;
dispif( choice );

end; (* end of selection 4 *)
end; (* end of procedure *)

FILE: FORM.PAS
FILE: GETCOM.PAS

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 remains here until a valid command is entered.
global variables used: help_level, cmdbuffer, call_routine, abort_command

author: vincent m. parisil ii, capt., usaf
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
                         words are displayed. *)
  error_code := 'a';

  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_line( decode, rec_num ); (* get first dictionary entry *)
        level := 1;
        (* initialize for instruction *)
        (* and validec *)
      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 (* display the appropriate command *)
      prompt_hlp( rec_num, pr_hlp_row ); (* 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
    display_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 vali *)

(* 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

(* help_level > 1 then *)

begin

help_level := 1;

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
procedure title_slide;  
begin  
  clear;  
  gotoxy(2,34);  
  write( ' WELCOME TO ' );  
  rectangle(4,19,41,7);  
  (*------- begin writing "ICECAP" in big letters -------*)  
  highlight;  
end;
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(' ');}

(*------------ 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, On 45433');
gotoxy(22,21);
hightlight;
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. Tarcynski');
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 It, capt., usaf

FILE: GETDAT.PAS
procedure bid_stat_line
  (help_level : integer; temp : boolean; printer : boolean;
   trans : boolean);
begin
  help := char;
  on_off := string[3];
  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 bid_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 1960 A
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_dev_name : paramstring;
var trans_file_name : paramstring;

FILE: GETDAT.PAS
var macro_file_name : paramstring;

type datarecord = record
data : data;
end;
datat = ^datarecord;

var data_file : file of data;
data_rec = datat;

begin
writein('Initializing the MICROSDW Menu Structure',
'and Hardware configuration');
assign(data_file, 'MICROSDW.SYS');
reset(data_file);
new(data_rec);
read(data_file, data_rec.data);
close(data_file);
move(data_rec.data, term, term_dat, (term_length + term_length));
blanks := ' '; call_routine := ' '; 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_rec.data.print[i];
for i := 1 to num_msg_dir do
begin
msg_dir i.loc_rec := data_rec.data.msg_dir[i].loc_rec;
msg_dir[i].length := data_rec.data.msg_dir[i].length;
end;
for i := 1 to num_ptr 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[1] := data_recs^.tdata.decode_dict.words[1];
    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 *)
    build_status_line( help_level, temp, printer, trans );

(""""""
(* 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**DELETE**DELETE**DELETE**DELETE**DELETE**DELETE**DELETE**DELETE
(*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
 end;

FILE: GETDAT.PAS
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);
(*-------- begin writing "ICECAP" in big letters --------*)
  highlight:

FILE: GETDAT.PAS *** IBM ONLY ***
gotoy(5,21); write(' '); gotoy(5,26); write(' '); gotoy(5,31); write(' '); gotoy(5,36); write(' '); gotoy(5,41); write(' '); gotoy(5,46); write(' '); gotoy(5,51); write(' '); gotoy(5,56); write(''); gotoy(6,22); write(' '); gotoy(6,26); write(' '); gotoy(6,31); write(' '); gotoy(6,36); write(' '); gotoy(6,41); write(' '); gotoy(6,46); write(' '); gotoy(6,51); write(' '); gotoy(6,56); write(''); gotoy(7,22); write(' '); gotoy(7,26); write(' '); gotoy(7,31); write(' '); gotoy(7,36); write(' '); gotoy(7,41); write(' '); gotoy(7,46); write(' '); gotoy(7,51); write(' '); gotoy(7,56); write(''); gotoy(8,22); write(' '); gotoy(8,26); write(' '); gotoy(8,31); write(' '); gotoy(8,36); write(' '); gotoy(8,41); write(' '); gotoy(8,46); write(' '); gotoy(8,51); write(' '); gotoy(8,56); write(''); gotoy(9,21); write(' '); gotoy(9,26); write(' '); gotoy(9,31); write(' '); gotoy(9,36); write(' '); gotoy(9,41); write(' '); gotoy(9,46); write(' '); gotoy(9,50); write(' '); gotoy(9,53); write(''); gotoy(12,11); write(' INTERACTIVE CONTROL ENGINEERING COMPUTER ANALYSIS PACKAGE '); gotoy(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... <<< ');
nonhighlight;
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(10,20);
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  
help_level, temp, printer,  
trans  
called by:  get_dct  
author:  vincent m. parish ii, capt., usaf

FILE: GETDAT.PAS  *** IBM ONLY ***
procedure bid_stat_line
   ( help_level : integer; temp : boolean; printer : boolean;
     trans : boolean );
var help : char;
on_off : string[ 3 ];
begina
    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.src and
bid_stat_line.
global variables used: blanks, call_routine,
status_line, msg_dir,
deck, strike, printer, trans,
temp, CRT, show_abbreviation,

FILE: GETDAT.PAS  *** IBM ONLY ***
procedure get_data(var term_dat : term_array;
var print_dat : print_array;
var msg_dir : msg_array;
var decode_dic : 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_dev_name : paramstring;
var trans_file_name : paramstring;

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; 
{ 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) 

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[1];
  decode_dict.abbrev[i] := data_recs.tdata.decode_dict.abbrev[1];
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 *)
bld_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***DELETE*

This section of Moore's original code was commented out by Mashiko and Tarcynski 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' );
*DELETE**DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DELETE***DEL
file: GETINT.PAS

procedures contained: del_list_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.

author: vincent.m. parisi 11, 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: int
global variables changed: str
returned variables: getch

called by: get_int, get_real

author: vincent.m. parisi 11, capt.. usaf

function getch : char;
begin
  if length( str ) > 0 then
    begin
      getch := str[1];
      delete( str, 1, 1 );
      end
    else
      getch := ' ';
  end;

FILE: GETINT.PAS
procedure del_lst_ch;
begin
  write(chr(backspace));
  write('');
  write(chr(backspace));
end;

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_list_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
files written: TEMP.OUT, TRANSACTION, PRINTER.OUT
called by: make_pretty
author: vincent m. parisi ff, capt., usaaf

procedure out_int( number : integer; field : integer; dest : char );
begin
    case dest of
    'C', 'c' :
        write( number:field );
        writeln( number:field );
    'P', 'p' :
        begin
            write( number:field );
            writeln( number:field );
        end;
    'B', 'b' :
        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;
    '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;

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: GETINT.PAS
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.abbrevs[ 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
file: 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

passed variables: num_rows

procedures called: graphics, gotoxy, nographics
called by: make_pretty_small_matrix,
make_pretty_large_matrix_one,
make_pretty_large_matrix_two

author: Gary C. Tarczynski, Capt, USAF

Susan K. Mashiko, Capt, USAF

procedure left_bracket( var num_rows : integer );

var

column_length : integer;
1 : 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  (* draw column *)
begin
  gotoxy(1,7);
  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
passed variables:  num_rows, num_cols
procedures called:  graphics, gotoxy, nographics
called by:  make_pretty_small_matrix,
           make_pretty_large_matrix_two
author:  Gary C. Tarczynski, Capt., USAF
         Susan K. Moshiko, 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;
  graphics;
  gotoxy(5, column_location);  (* draw upper right corner *)
  write( chr( term[61] ));

FILE: GETMAT.PAS
for i := 6 to column_length do  (* draw column *)  
begin  
gotoxy(i, column_location);  
write( chr( term[54] ));  
end;  
gotoxy( column_length + 1, column_location);  
write( chr( term[62] ));  (* draw lower right corner *)  
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:  num_row, num_col  
passed variables:  as_assigned  
procedures called:  gotoxy, out_string,  
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 1 := 1 to num_row do  
begin  
gotoxy( row + 1, 2 );  
out_string( 'ROW', as_assigned );  
gotoxy( row + 1, 5 );  
out_int( 1, 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 R. Mashiko, Capt, USAF
                       Gary C. Tarzynski, 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;

FILE: GETMAT.PAS
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: matrix
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[1, 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[1, col_element] := number;
          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[1, col_element] := number;
          end;
      end;
  end;
end;

FILE: GETMAT.PAS
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 variables changed: 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( i, 2);
            out_string('ROW', as_assigned);
            gotoxy( i, 5);
            out_int( i, 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;

FILE: GETMAT.PAS
gotoxy(4, l 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,  
gotoxy, get_int, get_matrix_entries,  
disp_msg, clear_msg,  
make_matrix, make_matrix_on,  
make_matrix_on,  
called by: Susan K. Mashiko, Capt, USAF  
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
procedure get_string;
(* arguments are specified in an earlier "forward" reference
 ( var strng : msg_line; var abort_command : boolean;
   in_dev : ch; chr1, chr2 : char ); *)

begin
  strng := ''; (* clear the string *)
  abort_command := false;

FILE: GETSTRIN.PAS
case in_dev 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 *)
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
- get_fact, form_poly,
- disp_poly, roots,
- disp_fact, get_unfact,
- poly, get_tf

**version:** 1.2
**date:** 20 August 1986
**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 (i.e. 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:
- crl only, as assigned

passed variables:
- number, row, col, abort_command

procedures called:
- gotoxy, highlight,
  - nonhighlight, out_string,
  - get_real, out_real

called by:
- get_fact

**author:**
- vincent 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 1986

FILE: GETTF.PAS
procedure get_r_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 );
  nhighlight;
  gotoxy( 20, 0 );
  out_string( blanks, crt_only );
  gotoxy( 20, 10 );
  highlight;
  out_string( 'Your number....', as_assigned );
  nhighlight;
  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: crt_only, screen_width, as_assigned
passed variables: row, degree
procedures called: gotoxy, disp_msg,
graphics, out_string,
nofigraphics, out_int,
called by: get_fact, get_unfact
author: vincent m. parisi, 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 + 1 + i ), 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
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 ii, capt., usaf
mod description: Changed the order of conjugate storage
modified by: Susan K. Mashiko, Capt., USAF
              Gary C. Tarczynski, 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
```pascal
getxy( 20, 0 );
disp_msg( 47 );
pause;
getxy( 20, 0 );
clear_msg( 47 );
goto repeat1;
end;

getxy( 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 ].impart := 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 impart 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 ), 99 );
    end
end
```

FILE: GETTF.PAS
if number < 0.0 then
  begin
    poly.polyfact[ 1 ].imagpart := number;
    out_real( number, 12, as_assigned );
    poly.polyfact[ 1 + 1 ].imagpart := ( number * (-1.0) );
  end
else
  begin
    poly.polyfact[ 1 ].imagpart := ( number * (-1.0) );
    out_real( poly.polyfact[ 1 ].imagpart, 12, as_assigned );
    poly.polyfact[ 1 + 1 ].imagpart := number;
    i := i + 1;
    gotoxy( ( row + i = 3 ), 59 );
    out_real( poly.polyfact[ 1 ].imagpart, 12, as_assigned );
  end;
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: maxdeg!
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_coef : array[1..maxdeg] of real;
  imag_coef : array[1..maxdeg] of real;
  i, j, k : integer;
  number_of_roots : integer;
  number_of_coef : integer;

begin
  number_of_roots := poly.polydeg;
  number_of_coef := 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 k do
begin
real_coeff[k] := -real_coeff[k-1] * poly.polyfact[i].realpart
+ imag_coeff[k-1] * poly.polyfact[i].imagpart
real_coeff[k] := -real_coeff[k];
imag_coeff[k] := -imag_coeff[k-1] * poly.polyfact[i].realpart
+ imag_coeff[k-1] * poly.polyfact[i].imagpart
imag_coeff[k] := -imag_coeff[k];
k := k - 1;
end;
end;
for i := 1 to number_of_coeff do
begin
poly.poly[i] := real_coeff[i];
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
returned variables: poly, row, abort_command
procedures called: make_pretty, get_r_num, gotoxy,
out_real, disp_msg, pause, clear_msg
called by: poly
author: Susan K. Mashiko, Capt, USAF
        Gary C. Tarczynski, Capt, USAF
modified by: author
mod description: Corrected the gain handling procedures
mod date: 25 Sep 85
------------------------------------------------------------------------------------
procedure get_intact( var poly : polynomial; row : integer; 
var abort_command : boolean );

label repeat1, 
repeat2;

var i : integer;
number : real;

begin
make_pretty( row, poly, poly_deg );
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.poly_deg + 1 ) do
begin
  (* get the coefficients for the polynomial, *)
  (* display them and save them *)
  get_r_num( number, ( row + 3 + 1 ), 7, abort_command );
  if abort_command then exit;
poly.poly[ i ] := number;
i := i + 1;
end; (* end while *)

(* the leading coefficient of the polynomial cannot be zero *)
if poly.poly[ 1 ] = 0 then
begin
  gotoxy( 20, 0 );
disp_msg( 62 );
pause;
end;

FILE: GETTF.WAS
procedure: roots
version: 2.0
date: 6 September 85
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: maxdeg
passed variables: poly
returned variables: poly
procedures called: gatxy, 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
  bi = array[1..30] of real;

FILE: GETTF.PAS
Var
ul : real;
v1 : real;
eps : real;
l, 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 *)
ul := 0.0;
v1 := 0.0;
eps := 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.polypoly[ i ];
degree := poly.polydeg;
(* Brixton's method expects the highest coefficient to be 1 *)
if coeff_poly[ 1 ] <> 0 then
  for i := 1 to degree do
    begin
      coeff_poly[ i + 1 ] := ( coeff_poly[ i ] + 1 ) /
        ( coeff_poly[ i ] );
    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;
else
   if degree = 2 then goto 8;
   if degree > 2 then goto 13;
8: u := coeff_poly[1];
   v := coeff_poly[2];
   iteration := 1;
   real_root := -(u / 2);
   rad := sqr(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;
9: 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[1] := 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 := u1;
v := v1;
iteration := 1;

(* calculate the b values *)
for k := 3 to degree do
b[k] := coeff_poly[k] - b[k-1] * u - b[k-2] * v;

(* calculate the c values *)
m := degree - 1;
for k := 3 to m do
b[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] - sqr( 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 18;
17: denom := c[degree - 1] * c[degree - 3] - sqr( c[degree - 2] );
if denom = 0 then goto 30;
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;

FILE: GETTF.PAS
19: store := sum;  
goto 21;  
20: if iteration = 50 then goto 28;  
   if iteration >= 5000 then  
      begin  
         gotoxy( 21, 1 );  
         highlight;  
         writeln(' ITERATING STOPPED AFTER 5000 ITERATIONS ');  
         nohighlight;  
         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 );  
         nohighlight;  
      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 ');  
writeln(' Enter correct polynomial ');  
nohighlight;  
pause;  
gotoxy( 21, 1 );  
writeln(' ');  
writeln(' ');  
exit;  
30: gotoxy( 21, 1 );  
highlight;  
writeln(' ERROR - Calculated denominator is zero ');  
writeln(' Enter correct polynomial ');  
nohighlight;  
pause;  
FILE: GETTF.PAS
```pascal
* ----------------------------------------------------------------------- *
* 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 used: method, poly, disp_row, abort_command        *
* procedures called: get_unfact, roots,                              *
*                    get_fact, form_poly,                              *
* called by: get_unfact                                               *
* author: vincent m. parti I, Capt., USAF                            *
* modified by: Susan K. Mashiko, Capt., USAF                         *
* 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
procedure dispf( 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;
umerator : 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 = 'TF5' then stor_loc := 16;

(* Now save the transfer function in the file *)
assign( polys, 'tf&pols.dat' );
reset( polys );
seek( polys, stor_loc );
write( polys, numerator );
seek( polys, ( stor_loc + 1 ) );
write( polys, denominator );

(* commented out by Maschio 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
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
program ICECAPPC:

(constant)

const
words = 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..buffersize] of string[wordsize];
cmdword = string[wordsize];
msg_line = string[screen_width];
dictionary = record
dictword : cmdword;
matchp : integer;
nomatchp : integer; (* minimum length of abbreviation *)
abbrev : byte;
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;
trt : 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;
dedecode_dict : dict_buffer;
msg_txd : file of msg_line;
string : msg_line;
deencode : dictionary;
list_dev_name : paramstring;
trans_file_name : paramstring;

FILE: ICECAPP.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 MICROSWD *
*******************************************************************************

($I ucase.pas )
($I terminal.pas )
($I output.pas )
($I pause.pas )
($I getdat.pas )
($I msg.pas )
($I instruct.pas )
($I getline.pas )
($I promptme.pas )
($I promptcm.pas )
($I trim.pas )
($I displayc.pas )
($I getint.pas )
($I getstr.pas )
($I readcom.pas )
($I processer.pas )
($I valdec.pas )
($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:polym.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 *)
($I b:dis.pas ) (* added 4 Sep 85 *)
($I b:bode.pas ) (* added 11 Oct 85 *)
($I b:select.pas ) (* modified Sep 85 *)

FILE: ICECAPPC.PAS *** Z100 floppy disk version ***
(*****************************************************************************
  * 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_t 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  *** Z100 floppy disk version ***
program ICECAPPC;

(FILE: ICECAPPC.PAS)

(* ******************************************************
** file: ICECAPPC.PAS                          ** hard disk version **
** program contained: ICECAPPC **
** version: 9.00 **
** date: 11 Oct 1985 **
** description: This file contains the main program for **
** the MICROSW 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                          ** hard disk version **
** 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, **
** selectRoutine **
** authors: Gary C. Tarczynski, Capt. USAF **
** Susan K. Mashiko, Capt. USAF **
* ******************************************************)

program ICECAPPC;

(*$1 msdwcons.pas) *(\$1 msdwtype.pas)

const
wordsiz e  = 12;
buffersiz e  = 6;
stat_line_width  = 77;
crt_only = 'c';
terminal_only = 't';
as_assigned = 'a';
backspace  = 8;
del  = 127;
yes  = true;

FILE: ICECAPPC.PAS  ** Z100 hard disk version **
literal = float;
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[words];
cmdword = string[words];
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_Outline : 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;
lst_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;
lst_dev_name : paramstring;
trans_file_name : paramstring;

FILE: ICECAPPC.PAS *** Z100 hard disk version ***
macro_file_name : string;
number_of_commands : integer;

($L concons.pas) (* added 14 Aug 85 these declarations are *)
(* unique to the controls package ICECAP *)

*****************************************************************************
* Include the sources for the routines called by MICROSDW *
*****************************************************************************

($L ucase.pas)
($L terminal.pas)
($L output.pas)
($L pause.pas)
($L getdat.pas)
($L msg.pas)
($L instruct.pas)
($L getline.pas)
($L prompte.pas)
($L promptcm.pas)
($L trim.pas)
($L displayc.pas)
($L getint.pas)
($L getstrin.pas)
($L readcom.pas)
($L processor.pas)
($L valdec.pas)
($L getcom.pas)

($L recover.pas) (* added 9 Sep 85 *)
($L update.pas) (* added 9 Sep 85 *)
($L copy.pas) (* added 5 Sep 85 *)
($L help.pas) (* added 9 Aug 85 *)
($L realis.pas) (* added 13 Aug 85 *)
($L getft.pas) (* added 13 Aug 85 *)
($L getmat.pas) (* added 11 Sep 85 *)
($L matrixman.pas) (* added 20 Sep 85 *)
($L matrix.pas) (* added 20 Sep 85 *)
($L polym.pas) (* added 4 Sep 85 *)
($L poly.pas) (* added 5 Sep 85 *)
($L form.pas) (* added 7 Oct 85 *)
($L define.pas) (* added 12 Aug 85 *)
($L inroot.pas) (* added 8 Sep 85 *)
($L delroot.pas) (* added 7 Sep 85 *)
($L modify.pas) (* added 23 Sep 85 *)
($L disp.pas) (* added 4 Sep 85 *)
($L bode.pas) (* added 11 Oct 85 *)
($L select.pas) (* modified Sep 85 *)

FILE: ICECAPPC.PAS
*** Z100 hard disk version ***
(**************************************************************************
* 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');
Clear-Screen;
end.  (* end of main program *)

FILE: ICECAPPC.PAS    *** Z100 hard disk version ***
file: ICECAPPC.PAS  *** IBM ONLY  floppy disk version
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  *** IBM ONLY  floppy disk version
version: 9.00
date: 11 Oct 1985
description: This program provides a flexible user interface for software development or other applications. This program also contains the ICECAP-PC subroutines.
    procedures called: get_cmd, pause, ClearScreen, get_data, select RTEune
    authors:       Gary C. Tarczynski, Capt. USAF
                    Susan K. Mashiko, Capt. USAF

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[words] size ;
cmdword = string[words] size ;
msg_line = string[screen_width] ;
dictionary = record
dictword : cmdword;
matchp : integer;
nonmatchp : integer;
abbrev : byte; (* minimum length of abbreviation *)
end;

(*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, bp, sp, di, ds, es, flags : integer;
end;

(*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;
print : boolean;
temp : boolean;
crt : boolean;
macro_error : boolean;
show_abbreviation : boolean;
in_terminal : boolean;
stat_on : boolean;
macro_file : text;
trans_file : text;

FILE: ICECAPP.CAS

*** IBM ONLY floppy disk version ***
begin  (* begin main program *)

(*INITIALIZE******************************
 *
 * 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***INSERT**

This statement was added by Tarczyński and Marshiko 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 ***
get_detail( 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;

(*insert***insert***insert***insert***insert***insert***insert*)

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:

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

III 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..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***)

  This type declaration was added by Tarczynski
  and Mashiko to be able to use the MS-DOS
  function call in the procedure standard_output.*)

  regpsk = record
    al, ah, bl, bh, cl, ch, dl, dh : byte;
    ax, bx, cx, dx, bp, si, di, ds, es, flags : integer;
  end;

(*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 : parameter;
trans_file_name : parameter;
macro_file_name : parameter;
number_of_commands : integer;

(*$I concons.pas *) (* added 14 Aug 85 these declarations are *)
(* unique to the controls package ICECAP *)

(*$I stdout.pas *)

(*$I ucase.pas *)
(*$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 procsesr.pas *)

FILE: ICECAPPC.PAS *** IBM ONLY hard disk version ***
program main;

(* 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***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 := Ofa(standard_output);

FILE: ICECAPPCL.PAS *** IBM ONLY hard disk version ***
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;

(*insert ***insert ***insert ***insert ***insert ***insert ***insert*)

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  *** IBM ONLY hard disk version ***
procedure inroot
version: 2.0
date: 19 September 85
description: This procedure will insert a root from a polynomial. 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&POLS.DAT
files written: TF&POLS.DAT
procedures called: clear, gotoxy, display, trim, out_real, out_string,
get_r_num, out_int, form_poly, pause
called by: select
author: Susan K. Mashiiko, Capt, USAF
Gary C. Tarczynski, Capt, USAF
mod description: Changes were made to the 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 := cmdbuffer[3];
  trim(choice);
  if choice = 'POLYA' then stor_loc := 18
  else
    if choice = 'POLYB' then stor_loc := 19
    else
      if choice = 'POLYC' then stor_loc := 20
      else
        if choice = 'POLYO' then stor_loc := 21
        else
          if choice = 'POLYE' 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 = 'HNPOLE' then stor_loc := 6
                        else
                          if choice = 'HDPOLE' then stor_loc := 7;

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

  (* display the polynomial *)
  dispoly(choice);

  (* copy all of the existing roots into the new polynomial *)
  for i := 1 to oldpoly.polydeg do
    begin
      newpoly.polyfact[1].realpart := oldpoly.polyfact[i].realpart;
    end;

FILE: INROOT.PAS
newpoly.polyfact[ i ].imagpart := oldpoly.polyfact[ i ].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 + 1 ].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&polis.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

**called by:** getcom

**author:** vincent m. parisi ii, capt., usaf

---

```pascal
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
      '(Now enter one of these commandwords, or "$" to abort...', 'c')
end;
```

---

FILE: INSTRUCT.PAS
procedure disp_matrix( var choice : cmdword );

var
  stor_loc : integer;
  mats_ : file of matrix;
  col_element : integer;

FILE: MATRIX.PAS
i : integer;
j : integer;
number : real;
mat : matrix;
um_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 = 'MATE' then stor_loc := 3
        else if choice = 'MATD' 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;
um_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 + 1 ), ( col + ( j * 13 ) ) );
            out_real( mat.element[ i, col_element ], 12, as_assigned );
  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 + 1 ), ( 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 + 1 ), ( col + ( j * 13 ) ) );
  out_real( mat_element[ i, col_element ], 12, as_assigned );
end;
end;
end;

*****************************************************************************
procedure:  matrix_manip
version:  1.0
date:  21 September 85
description:  This procedure will add, subtract, or multiply two matrices.
global variables used:  abort_command
passed variables:  first, second, result, mat_obj
files read:  MATRIX.DAT
files written:  MATRIX.DAT
procedures called:  trim, disp_matrix,
                   matrixadd, matrixmul, matrixsub, pause
called by:  mmatrix
author:  Susan K. Mashiko, Capt, USAF

FILE:  MATRIX.PAS
procedure matrix_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;
1        : 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 = 'MATH' 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 = 'MATC' then stor_loc := 2
else if second = 'MATH' 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
  mmatrixmul( 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
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 = 'TRANSPOSE' 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
    smatrixmlt( 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_matrix_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);
  nolight;
  gotoxy(20, 0);
  out_string(blanks, crt_only);
  gotoxy(20, 10);
  highlight;
  out_string('Your choice... ', as_assigned);
  nolight;
  get_string(mat_name, abort_command, as_assigned, ' ', ' ');
ucase(mat_name);
  if ((mat_name = 'MATA') or (mat_name = 'MATB') 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
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 = 'MAT0' ) or ( mat_obj = 'MATC' ) 
or ( mat_obj = 'MATD' ) or ( mat_obj = 'MATE' ) ) then 
begin -- 
disp_matrix( mat_obj ); 
pause; 
end 
else 
if (( mat_obj = 'ADD' ) or ( mat_obj = 'SUBTRACT' ) or 
( mat_obj = 'MATHMUL' ) ) 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_matrix_name( mat_name, 15, 15, abort_command ); 

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 = 'TRANSPOSE' ) 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 = 'SCLAMULT' then
begin
  clear;
gotoxy( 5, 0 );
disp_msg( 55 );
get_matrix_name( mat_name, 12, 44, abort_command );
first := mat_name;
g_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 );
peuse;
clear;
end;
end;

FILE: MATRIX.PAS
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 *)
(* 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
```pascal
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;
(* store the dimensions of the new matrix *)
cmat.num_rows := amat.num_rows;
cmat.num_cols := amat.num_cols;
end;

***********************************************************************
* procedure: matrsub
* 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
* global variables changed: abort_command
* passed variables: amat, bmat, cmat, abort_command
* returned variables: cmat, abort_command
* procedures called: matradd
* called by: matrx_manip
* author: Susan K. Mashiko, Capt., USAF
* Gary C. Tarczynski, Capt., USAF
***********************************************************************

procedure matrsub( var amat : matrix; var bmat : matrix;
                      var cmat : matrix; var abort_command : boolean );
var

FILE: MATRXMAN.PAS
```
i : integer;
j : integer;
nbmat : matrix;

begin
abolvommand := false;
(* negate all of the elements of the second matrix *)
(* then call the matrix add routine *)
for j := 1 to amat.num_cols do
begin
for i := 1 to amat.num_rows do
  nbmat.element[ i, j ] := - bmat.element[ i, j ];
end;
nbmat.num_rows := bmat.num_rows;
nbmat.num_cols := bmat.num_cols;
matrixadd( amat, nbmat, cmat, abort_command );
if abort_command then exit;
end;

*********************************************************************************

procedure: mmatrixmit
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: cmat, abort_command
procedures called: clear, gotoxy, pause, disp_msg
called by: matrix_manip
author: Susan K. Mashiko, Capt, USAF
        Gary C. Tarczynski, Capt, USAF
*********************************************************************************

procedure mmatrixmit( var amat : matrix; var bmat : matrix;
var cmat : matrix; var abort_command : boolean );

var
  i : integer;
j : integer;
i : 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
      begin
      end;
    end;
    cmat.num_rows := amat.num_rows;
    cmat.num_cols := bmat.num_cols;
  end;
end;

******************************************************************************
procedure:  smatrixmit
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 smatrixmit( 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[j, i] * number;
    end;
  bmat.num_rows := amat.num_rows;
  bmat.num_cols := amat.num_cols;
end;

(*------------------------------------------------------------------------*)

procedure:  matrxtran
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:              matrx_manip2
author:                 Susan K. Mashiko, Capt, USAF
                        Gary C. Tarczyński, Capt, USAF
(*------------------------------------------------------------------------*)

procedure matrxtran( 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;
  bmat.num_rows := amat.num_cols;
  bmat.num_cols := amat.num_rows;
end;

(*------------------------------------------------------------------------*)

procedure:  matrxeinv
version:     2.0
date:        6 November 85
description: This procedure inverts the first matrix and stores the result in the second.

global variables used:  abort_command

FILE: MATRXMAN.PAS
global variables changed: abort_command
passed variables: amat, bmat, abort_command
return variables: amat, bmat, abort_command
procedures called: clear, gotoxy, disp_msg, pause
called by: matrix_manip
author: Susan K. Mashiko, Capt, USAF
        Gary C. Tarczyynski, Capt, USAF
modification: replaced the algorithm for inverting a matrix
modified by: author

******************************************************************************

procedure matrixinv( var amat : matrix; var bmat : matrix;
                      var abort_command : boolean );

label   single;

var    dimension : integer;
       l, j, k:
       1, 12, IP:
       J1:
       p:
       PE, TPE:

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;

FILE: MATRIXMAN.PAS
(* 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[1] := 1;
                for j := 1 to dimension do
                    bmat.element[i, j] := amat.element[i, j];
                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, I1];
                        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
                            begin
                                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
                                            begin
                                                clear;
                                                gotoxy(7,1);
                                                disp_msg(53);
                                                pause;
                                                clear;
                                                abort_command := true;
                                                exit;
                                            end;
                                    end;
                        end;
                    end;
    end;
FILE: MATRIXMAN.PAS
begin
    TPE := -bmat.element[ j, 1 ];
    for J1 := 2 to dimension do
        bmat.element[ J1, J1 - 1 ] :=
        bmat.element[ j, J1 ] + TPE * p(J1 - 1 );
        bmat.element[ J1, dimension ] := TPE * p[dimension ];
    end;
    for j := 1 to dimension do
        bmat.element[ dimension, j ] := p[j ];
    end
else
goto single;
end;
for i := 1 to dimension do
    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: MATRIXMAN.PAS
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 column location should be modified and will store the result in the original location.
global variables used: cmdbuffer, abort_command
global constants used: as_assigned, crt_only
passed variables: cmdbuffer, wordnumber
files read: MATRIX.DAT
files written: MATRIX.DAT
procedures called: trim, gotoxy, disp_msg, out_string, clear_msg, out_real, clear, pause, get_r_num, get_string, ucw, 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 cmdbuffer : buffer; wordnumber : integer );
label second_page,
again,
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 := cmdbuffer[ 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 * 1), (col + (j * 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(‘
    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;
    until ( (chg_row > 0) and (chg_row <= mat.num_rows) );
(* get the column of the change *)
repeat
  begin
    gotoxy(19, 46);
    out_string(‘,
    gotoxy(19, 46);
    get_int(chg_col, abort_command);
    if abort_command then exit;
  end;
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_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;
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 );

(* get the row of the change or the prompt for next *)
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;
  ucase( select );
  if select = 'NEXT' then
    goto second_page

FILE: MODIFY.PAS
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;

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 + 1, col + ( j * 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 get the column of the change
begin
  gotoxy( 19, 46 );
  out_string( ' ', crt_only );
  gotoxy( 19, 46 );
  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_matr(x( 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. Masniko, Capt., USAF
          Gary C. Tarzynski, 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 = 'ADDROOT' then
  inroot( cmdbuffer, number_of_commands )
else
  if mod_obj = 'DELROOT' then
    delroot( cmdbuffer, number_of_commands )
  else
  if mod_obj = 'CHANGE' then
    chgmet( cmdbuffer, number_of_commands )
  else
  if mod_obj = 'HELP' then
    begin
      clear;
      disp_msg( 25 );
      pause;
      clear;
    end;
  end;

FILE: MODIFY.PAS
FILE: MSDWCONS.PAS

const

term_length = 95;  (* length of array for terminal control data *)
printer_length = 50;  (* length of array for printer control data *)
ff = 12;  (* decimal form feed char *)
wordlength = 9;  (* length of word in storage *)
num_param_group = 1;
um_bools = 10;
um_ints = 10;
um_reals = 10;
um_strings = 10;
um_ptr_recs = 3;
um_ptrs = 350;
um_words = 100;
um_msg_dir = 70;
um_msg_line = 250;
screen_width = 79;

ENDCODE = 9999;
DONEWORD = 'SSSSSSSS';

FILE: MSDWCONS.PAS
```pascal
(* --------------------------------------------------- *)
(* 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_ptr_recs ] 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: MSDWTYPE.PAS
procedure disp_line( rec_num : integer );

var
print_line : string[screen_width];

begin
  seek( msg_txt, rec_num );
  read( msg_txt, print_line );
  writeln( print_line, as_assigned );
end;

FILE: MSG.PAS
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. parisil 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 );
            writeln;
         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
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
          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);
        end
    end
end

FILE: MSG.PAS
gotoxy( 22, 0 );
clear_msg( 13 );
if resp = 'S' then exit;
gotoxy( 22, 0 );
end;
clear;
for i := 0 to remain_lines do
    disp_line( (rec_num + 1) );  (* records on disk begin at 0 *)
end;
end;

FILE: MSG.PAS
procedure out_string( ostring : msg_line; dest : char );
begin

FILE: OUTPUT.PAS
case dest of
  'C', 'c': write( ostring );
  'P', 'p': writeln( list_dev, ostring );
  'B', 'b': begin
    writeln( ostring );
    writeln( list_dev, ostring );
  end;
  'A', 'a': begin
    if crt then writeln( 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
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 );
  nonhighlight;
  read( resp );
  if stat_on then
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, polymit, polysub

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;
t : 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.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;
  (* establish counters for do loops *)
  ncc := (cpoly.polydeg + 1);
  n := abs( apoly.polydeg - bpoly.polydeg );
  if (apoly.polydeg - bpoly.polydeg) < 0 then
    begin
      for i := 1 to n do
        cpoly.polypoly[i] := bpoly.polypoly[i];
      nn := n + 1;
      for i := nn to ncc do
        cpoly.polypoly[i] := bpoly.polypoly[i] + apoly.polypoly[1 - n];
      goto 50
    end
  else
    if (apoly.polydeg - bpoly.polydeg) = 0 then
      begin
        for i := 1 to n do
          cpoly.polypoly[i] := apoly.polypoly[i] + bpoly.polypoly[i];
        goto 50
      end
    else
      if (apoly.polydeg - bpoly.polydeg) > 0 then
        begin
          for i := 1 to n do
            cpoly.polypoly[i] := apoly.polypoly[i] * bpoly.polypoly[i];
        end
  end
end

FILE: POLYMAN.PAS
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
nub := apoly.polydeg + 1;
bpoly := bpoly.polydeg;
b.poly.coefficient := bpoly.coefficient;

for i := 1 to nub do
    nbpoly.poly[i] := bpoly.poly[i];

if (apoly.coefficient = bpoly.coefficient) and (apoly.polydeg = bpoly.polydeg) then

    begin
        for i := 1 to apoly.polydeg + 1 do
            if apoly.poly[i] <> bpoly.poly[i] then
                goto continue;
        else
            begin
                clear;
gotox(10,5);
                writeln('ERROR: you may not subtract equal polynomials from');
gotox(11,5);
                writeln('one another');
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 polynomials passed to it and places the result in the third.
passed variables: apoly, bpoly, cpoly
returned variables: clear, gotoxy
procedures used: highlight, nonhighlight, roots
called by: polymanip

FILE: POLYMAN.PAS
procedure polymerge(var apoly : polynomial; var bpoly : polynomial; var cpoly : polynomial);

var
  i, j : integer;
  naa, nbb : integer;

begin
  (* ensure the constant/gain is one *)
  if apoly.coefficient <> 1 then
    for i := 1 to (apoly.polydeg + 1) do
      apoly.polyval[i] := apoly.polyval[i] * apoly.coefficient;
    apoly.coefficient := 1;
  (* ensure the constant/gain is one *)
  if bpoly.coefficient <> 1 then
    for i := 1 to (bpoly.polydeg + 1) do
      bpoly.polyval[i] := bpoly.polyval[i] * bpoly.coefficient;
    bpoly.coefficient := 1;
  (* check to see if the resulting polynomial will be too large *)
  if cpoly.polydeg > 10 then
    begin
      clear;
      gotoxy(8, 0);
      writeln('Degree of result greater than 10, option aborted.');
      writeln('Due to the storage space limitations your resulting polynomial is limited to 10th order');
      exit;
    end
  else
    begin
      for i := 1 to 11 do
        cpoly.polyval[i] := 0.0;
      naa := apoly.polydeg + 1;
      nbb := bpoly.polydeg + 1;
      for i := 1 to naa do
        for j := 1 to nbb do
          ...

FILE: POLYMER.PAS
procedure spolymit( var apoly : polynomial; var bpoly : polynomial; var number : real );

var
  counter : integer;
  i : integer;

begin
  counter := apoly.polydeg + 1;
  if apoly.polydeg[ 1 ] <> 1 then
    begin
      for i := 2 to ( apoly.polydeg + 1 ) do
      apoly.polydeg[ 1 ] := 1;
      apoly.coefficient := apoly.coefficient * apoly.polydeg[ 1 ];
    end;

  cpoly.polydeg[ i * j - 1 ] := cpoly.polydeg[ i * j - 1 ] +
    apoly.polydeg[ i ] * bpoly.polydeg[ j ];
  cpoly.coefficient := apoly.coefficient * bpoly.coefficient;
end;

cpoly.coefficient := cpoly.polydeg[ 1 ];

(* standardization code for poly storage *)
if cpoly.polydeg[ 1 ] <> 1 then
  for i := 2 to ( cpoly.polydeg + 1 ) do
  cpoly.polydeg[ 1 ] := 1;
roots( cpoly );
end;

******************************************************************************
procedure spolymit( var apoly : polynomial; var bpoly : polynomial;
  var number : real );

var
  counter : integer;
  i : integer;

begin
  counter := apoly.polydeg + 1;
  if apoly.polydeg[ 1 ] <> 1 then
    begin
      for i := 2 to ( apoly.polydeg + 1 ) do
      apoly.polydeg[ 1 ] := 1;
      apoly.coefficient := apoly.coefficient * apoly.polydeg[ 1 ];
    end;

  cpoly.polydeg[ i * j - 1 ] := cpoly.polydeg[ i * j - 1 ] +
    apoly.polydeg[ i ] * bpoly.polydeg[ j ];
  cpoly.coefficient := apoly.coefficient * bpoly.coefficient;
end;

cpoly.coefficient := cpoly.polydeg[ 1 ];

(* standardization code for poly storage *)
if cpoly.polydeg[ 1 ] <> 1 then
  for i := 2 to ( cpoly.polydeg + 1 ) do
  cpoly.polydeg[ 1 ] := 1;
roots( cpoly );
end;

******************************************************************************
procedure spolymit( var apoly : polynomial; var bpoly : polynomial;
  var number : real );

var
  counter : integer;
  i : integer;

begin
  counter := apoly.polydeg + 1;
  if apoly.polydeg[ 1 ] <> 1 then
    begin
      for i := 2 to ( apoly.polydeg + 1 ) do
      apoly.polydeg[ 1 ] := 1;
      apoly.coefficient := apoly.coefficient * apoly.polydeg[ 1 ];
    end;

  cpoly.polydeg[ i * j - 1 ] := cpoly.polydeg[ i * j - 1 ] +
    apoly.polydeg[ i ] * bpoly.polydeg[ j ];
  cpoly.coefficient := apoly.coefficient * bpoly.coefficient;
end;

cpoly.coefficient := cpoly.polydeg[ 1 ];

(* standardization code for poly storage *)
if cpoly.polydeg[ 1 ] <> 1 then
  for i := 2 to ( cpoly.polydeg + 1 ) do
  cpoly.polydeg[ 1 ] := 1;
roots( cpoly );
end;

******************************************************************************
file: POLY.PAS

*** Procedures contained: disp_poly, poly_manip, p_poly, get_poly_name, poly_manip2 ***
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 disp_poly
version: 1.0
date: 6 September 5
description: This procedure will display a polynomial from a record in TP&POLS.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: TP&POLS.DAT
procedures called: clear, gotoxy, out_string, disp_msg, make_pretty, out_real, trim
called by: poly_manip, p_poly, inroot, delroot
author: Susan K. Mashiko, Capt, USAF
       Gary C. Tarczynski, Capt, USAF

procedure disp_poly(var choice : cmdword);

var
stor_loc : integer;
poly5 : file of polynomial;
i : integer;
row : integer;
number : real;
pol : polynomial;

FILE: POLY.PAS
begin
    trim( choice );
    if choice = 'POLYA' then stor_loc := 18
    else if choice = 'POLYB' then stor_loc := 19
    else if choice = 'POLYC' then stor_loc := 20
    else if choice = 'POLYD' then stor_loc := 21
    else if choice = 'POLYE' 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 = 'MDPOLY' then stor_loc := 7;
    assign( polys, 'tf&pols.dat' );
    reset( polys );
    seek( polys, stor_loc );
    read( polys, pol1 );
    (* 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, pol1.polydeg );
    i := 1;

    (* get the coefficient and display it *)
    number := pol1.coefficient;
    gotoxy( ( row + 2 ), 19 );

FILE: POLY.PAS
out_real( number, 12, as_assigned );
gotoxy( row + 3 * i ), 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.polypoly[i], 12, as_assigned );
    i := i + 1;
  end;
end;

***********************************************************************

procedure:  polynmanip
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,
polyobj, pause
called by:    ppol
author:      Susan K. Mashiko, Capt, USAF
              Gary C. Tarczynski, Capt, USAF
***********************************************************************

procedure polynmanip( var first : cmdword; var second : cmdword;
var result : cmdword; var poly_obj : cmdword );

var
  stor_loc : integer;
  polys    : file of polynomial;

FILE: POLY.PAS
pol1, pol2 : polynomial;
pol3 : polynomial;
JT : 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 = 'UNPOLY' then stor_loc := 0
else if first = 'DDPOLY' then stor_loc := 1
else if first = 'CNPOLY' then stor_loc := 2
else if first = 'DOPOLY' then stor_loc := 3
else if first = 'UNPOLY' then stor_loc := 4
else if first = 'GPOLY' then stor_loc := 5
else if first = 'MNPOLY' then stor_loc := 6
else if first = 'MPOLY' then stor_loc := 7;
assign( polys, 'tf\&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 = 'POLVC' then stor_loc := 20
else if second = 'POLVD' then stor_loc := 21
else if second = 'POLVE' then stor_loc := 22

FILE: POLY.PAS
else
  if second = 'UNPOLY' 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, 'tt&polis.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
    polymlt( 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 = 'POLYA' then stor_loc := 18
else
  if result = 'POLYB' then stor_loc := 19
  else
    if result = 'POLYC' then stor_loc := 20
    else
      if result = 'POLYD' then stor_loc := 21
      else
        if result = 'POLYE' then stor_loc := 22
        else
          if result = 'ONPOLY' then stor_loc := 0
          else

FILE: POLY.PAS
if result = 'GDPOLY' then stor_loc := 1
else
  if result = 'CNPOLY' then stor_loc := 2
  else
    if result = 'GDPOLY' then stor_loc := 3
    else
      if result = 'CNPOLY' then stor_loc := 4
      else
        if result = 'GDPOLY' then stor_loc := 5
        else
          if result = 'CNPOLY' then stor_loc := 6
          else
            if result = 'GDPOLY' then stor_loc := 7;
          assign( polys, 'tf&pol.dat' );
          reset( polys );
          seekl( polys, stor_loc );
          write( polys, pol3 );
          close( polys );
          dispopoly( result );
          pause;
end;

procedure: polmanip2
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, dispopoly, spolylit
files read: TF&POL.SAT
files written: TF&POL.SAT
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
poll, pol2 : polynomial;
t : integer;

begin
  trim( first );
  if first = 'POLYA' then stor_loc := 18
  else
    if first = 'POLYB' 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 = 'ONPOLY' then stor_loc := 0
            else
              if first = 'OPOLY' then stor_loc := 1
              else
                if first = 'CNPOLY' then stor_loc := 2
                else
                  if first = 'CPOLY' 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&pols.dat' );
  reset( polys );
  seek( polys, stor_loc );
  read( polys, poll );
  close( polys );
  spolymit( poll, pol2, number );
  trim( result );
  if result = 'POLYA' then stor_loc := 18
  else
    if result = 'POLYB' then stor_loc := 19
    else
      if result = 'POLYC' then stor_loc := 20
      else
        if result = 'POLYD' then stor_loc := 21

FILE: POI-.AS
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 = 'MNPOLY' then stor_loc := 6
else
  if result = 'MDPOLY' then stor_loc := 7
assign( polys, 'tfpols.dat' );
reset( polys );
seek( polys, stor_loc );
write( polys, pol );
close( polys );
disppoly( 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, nonhighlight, gotoxy,
out_string, ucase, trim, pause,
get_string, disp_msg, clear_msg
called by: ppoly
author: Susan K. Manka, 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 = 'POLYB' ) or
( poly_name = 'POLYC' ) or ( poly_name = 'POLYD' ) or
( poly_name = 'DOPOLY' ) or ( poly_name = 'CNPOLY' ) or
( poly_name = 'GDPOLY' ) 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
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 = 'POLVB') or
(poly_obj = 'POLVC') or (poly_obj = 'POLVD') or
(poly_obj = 'POLYE') or (poly_obj = 'ONPOLY') or
(poly_obj = 'ODPOLY') or (poly_obj = 'CNPOLY') or
(poly_obj = 'GDPOLY') 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 = 'POLYMLT')) 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);
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
file: PROCESER.PAS

procedure procs_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.
author: vincent m. Parisi II, capt., usaf

procedure procs_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

FILE: PROCESER.PAS
for $i := 1$ to ($\text{level} - 1$) do
  disp-commandword(cmdbuffer, 1);
highlight;
  disp-commandword(cmdbuffer, level);
nonhighlight;
  for $i := ($level + 1$) to bufferpointer do
    disp-commandword(cmdbuffer, 1);
end;

if help_level > 2 then
begin
  gotoxy(20, 0);
  case error_code of
    'B', 'b' :
    begin
      disp_msg(4);
      pause;
    end;
    'C', 'c' :
    begin
      disp_msg(5);
      pause;
    end;
  end;
end;

FILE: PROCESER.PAS
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 );
  nolight;
end;

FILE: PROMPTCM.PAS
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( ' ' );
      displayword := decode( row_count );
j := 1;
repeat
  get_line( decode, rec_num );
  if decode.d dictword <= "DONEWORD" then
    begin
      displayword := decode.d 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: PROMPTHE.PAS
file:        READCOM.PAS
procedure contained:  readcom
version:      1.1
date:        28 August 1983
description:  This module contains the procedure
              that gets a command line from the user.
author:     vincent m. paris i 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
global variables used: cmdbuffer, abort_command, blanks,
              string, macro_error
global variables changed: cmdbuffer, string
global constants used: bufsize, 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. paris i ii, capt., usaf

begin (* main body of procedure readcom *)
  for i := bufferpointer to bufsize do  (* clear cmdbuffer not used to *)
        cmdbuffer[ i ] := copy( blanks, 1, wordsize );  (* spaces

FILE: READCOM.PAS
(* 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(strng, abort_command, terminal_only, ' ', ' ')
else
  get_string(strng, 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
    ucse ( strng ); (* change entire command to upper case *)
    lencmd := length(strng);
    i := 1;
    repeat
      while (( strng[i] = ' ') and ( i <= lencmd )) do
        i := i + 1;
      if (( strng[i] > ' ') and ( i <= lencmd )) then
        begin
          j := 0;
          while (( strng[j + i] <> ' ') and ((i + j) <= lencmd)) do
            j := j + 1;
          tword := copy(strng, 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
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;

FILE: REALS.PAS
procedure get_reals
version: 1.0
date: 19 August 1985
description: Handles the input of real numbers. Normal system
real number input has no error checking after the
<br/>&lt;CR&gt; has been entered. This is unsatisfactory
<br/>when using a particular layout on the screen as
<br/>the error message will more than likely occur at
<br/>an inappropriate place. This routine uses redirect-
<br/>ed IO in that the input is read into a string
<br/>through filters which only accept valid real
<br/>number characters. The string is null or with just
<br/>one space, the operator just entered a &lt;CR&gt; so
<br/>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, nonhighlight
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 );
    nhighlight;
    out_string( 'Your number...', as_assigned );
    nhighlight;
    end
  else
    i := i + 1;
  end: (* end of while loop *)
end; (* end of procedure *)
**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

---

```pascal
procedure recover;

label
repeat1,
repeat2;

FILE: RECOVER.PAS
```
```pascal
var
  polys : file of polynomial;
  polya : file of polynomial;
  pol : polynomial;
  l : integer;
  your_name : msg_line;
  mats : file of matrix;
  mata : file of matrix;
  mat : matrix;

begin
  clear;

  (* copy user file into tfpolys.dat *)
  repeat1:
    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 repeat1;
      end;

    assign( polys, 'tfpolys.dat' );
    rewrite( polys );

    assign( polya, your_name );
    reset( polya );

    for i := 0 to 22 do
      begin
        seek( polya, i );
        read( polya, pol );
        seek( polys, i );
        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 ) > 0 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
procedure select_routine(var call_routine : cmdword;
var cmdbuffer : buffer :
number_of_commands : integer);

var l : integer;
copy : file;

FILE: SELECT.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 *)
                        copy( 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;
        end;
    end;
end;

FILE: SELECT.PAS
procedure standard_output;
var
  r: regpeak;
begin
  r.ah := 2;  (* DOS function call to output to the display *)
end;

FILE: STDOUT.PAS  *** IBM ONLY ***
procedure nographics;
var i : integer;

begin
for i := 48 to (term[47]+47) do
  write(chr(term[i]));
end;

procedure highlight;
var i : integer;

begin
for i := 28 to (term[27]+27) do
  write(chr(term[i]));
end;

procedure nohighlight;

FILE: TERMINAL.PAS
procedure nohighlight;
var  i   : integer;
begin
  for i := 35 to ( term[34] * 34 ) do
    write(chr(term[i]));
end;

procedure gotoxy( row : integer; col : integer );
var  i   : integer;

FILE: TERMINAL.PAS
procedure ttypeswitch(integer: 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]; (* add row and col offsets *)
  col := col + term[8];
(* send out initial string *)
  for i := 2 to (term[1] + 1) do
    write(chr(term[1]));
  if term[9] > 0 then (* if 0 then row goes first, else *)
    ttype(term[90],row)
  else
    ttype(term[90],col);
(* send out intermediate *)
  for i := 11 to (term[10] + 10) do
    write(chr(term[1])); (* string if any *)
  if term[9] > 0 then (* send out remaining row or col *)
    ttype(term[90],col)
  else
    ttype(term[90],row);
(* now send out ending string if any *)
  for i := 15 to (term[14] + 14) do
    write(chr(term[1]));
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
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;
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
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: InStr, 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;
var i : integer;

begin
for i := 77 to ( term[76] + 76 ) do
  write( chr(term[i]) );
end;

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,Instr_str,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 (* top of Rectangle *)
    write(chr(term[55]));
    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 (* bottom of Rectangle *)
    write(chr(term[55]));
    write(chr(term[62]));  (* lower right corner *)

FILE: TERMINAL.PAS
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;
procedure ucase(var instring: msg_line);

var i: integer;
begin
  for i := 1 to length(instring) do
    instring[i] := upCase(instring[i]);
end;
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;
polya : file of polynomial;
pol : polynomial;
i : integer;
your_name : msg_line;
mats : file of matrix;
mata : file of matrix;
mats : matrix;

begin
  clear;
  /* update user file with tfpols.dat */
  repeat1:
    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 repeat1;
      end;
    assign( polys, 'tfpols.dat' );
    reset( polys );
    assign( polya, your_name );
    rewrite( polya );
    for i := 0 to 22 do
      begin
        seek( polys, i );
        read( polys, pol );
        seek( polya, i );
        write(polya, pol );
      end;
    close( polys );
    close( polya );
    /* 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
function check_word(decord : 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(dwrd);
  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(dwrd[i]) ) do
        i := i + 1;
      if (i = cmd_len+1) then check_word := true;
    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
nomatch 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 command buffer, 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

FILE: VALNDEC.PAS
procedure var n_dec( var level : integer; var rec_num : integer; 
    var error_code : char; num_of_commands : integer; 
    var cmdbuffer : buffer; var callRoutine : 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 "ENDECDE" 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 begin
          error_code := 'n';
          callRoutine := decode.dict.words[decode.matchp];
          end 
      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
cmdbuf[leve] := decode.dicrword; (* 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; (* end while *)

FILE: VALNDEC.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.
Cursor Positioning, initial char sequence

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
2 27 ESC 89 Y
3 6 0 8 0 7 32 row offset
4 3 0 8 0 32 column offset
5 10 0 intermediate character(s)
6 11 0
7 12 0
8 13 0
9 14 0 terminating character(s)
10 15 0
11 16 0
12 17 0
13 18 0
14 19 2 Clear Screen, Home Cursor
20 27 ESC
21 69 E
22 0
23 0
24 0
25 0
26 0
27 2 Highlight (enter reverse video)
28 27 ESC
29 112 p
30 0
31 0
32 0
33 0
34 2 Normal Video (exit reverse video)
35 27 ESC
36 113 q
37 0
38 0
39 0
40 2 Enter Graphics Mode
41 27 ESC
42 70 F
43 0
44 0
45 0
46 0
47 2 Exit Graphics Mode
48 27 ESC
49 71 G

FILE: TERM.TXT *** Zenith Z100 ***
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td><strong>Graphics</strong> - vertical line</td>
</tr>
<tr>
<td>51</td>
<td><strong>Graphics</strong> - horizontal line</td>
</tr>
<tr>
<td>52</td>
<td><strong>Graphics</strong> - line intersection (up and down)</td>
</tr>
<tr>
<td>53</td>
<td><strong>Graphics</strong> - centered dot</td>
</tr>
<tr>
<td>54</td>
<td><strong>Graphics</strong> - solid square (space in reverse video)</td>
</tr>
<tr>
<td>55</td>
<td><strong>Graphics</strong> - line intersection (diagonal)</td>
</tr>
<tr>
<td>56</td>
<td><strong>Graphics</strong> - top &quot;T&quot;</td>
</tr>
<tr>
<td>57</td>
<td><strong>Graphics</strong> - upper right corner</td>
</tr>
<tr>
<td>58</td>
<td><strong>Graphics</strong> - lower right corner</td>
</tr>
<tr>
<td>59</td>
<td><strong>Graphics</strong> - lower left corner</td>
</tr>
<tr>
<td>60</td>
<td><strong>Graphics</strong> - upper left corner</td>
</tr>
<tr>
<td>61</td>
<td><strong>Normal Video</strong> (exit reverse video)</td>
</tr>
<tr>
<td>70</td>
<td><strong>ESC</strong></td>
</tr>
<tr>
<td>71</td>
<td><strong>q</strong></td>
</tr>
<tr>
<td>72</td>
<td><strong>Video Bold</strong> (enter reverse video)</td>
</tr>
<tr>
<td>73</td>
<td><strong>ESC</strong></td>
</tr>
<tr>
<td>74</td>
<td><strong>p</strong></td>
</tr>
<tr>
<td>75</td>
<td><strong>Cursor Addressing. 0=decimal 1=ASCII</strong></td>
</tr>
<tr>
<td>76</td>
<td><strong>Row offset for beginning of plot. (not used)</strong></td>
</tr>
<tr>
<td>77</td>
<td><strong>Column offset for beginning of plot. (not used)</strong></td>
</tr>
<tr>
<td>78</td>
<td><strong>Vertical resolution of plotting area. (not used)</strong></td>
</tr>
<tr>
<td>79</td>
<td><strong>Horizontal resolution of plotting area. (not used)</strong></td>
</tr>
</tbody>
</table>

FILE: TERM.TXT *** Zenith 2100 ***
Cursor Positioning, initial char sequence

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>ESC</td>
<td></td>
</tr>
<tr>
<td>3-4</td>
<td>[</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6-7</td>
<td>row offset</td>
<td></td>
</tr>
<tr>
<td>8-9</td>
<td>column offset</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>intermediate character(s)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>terminating character(s)</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Clear Screen, Home Cursor</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>ESC</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>[</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>J</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Highlight (enter reverse video)</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>ESC</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>[</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>Normal Video (exit reverse video)</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>ESC</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>[</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>Enter Graphics Mode</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>Exit Graphics Mode</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

FILE: TERM.TXT  *** IBM Monochrome ***
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 [
73 48 0
74 109 m
75 0
76 4 Video Bold
77 27 ESC
78 91 [
79 49 1
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)

FILE: TERM.TXT *** IBM Monochrome ***
The power of the polynomial must be > 0 and < 10.

What is the degree of the numerator... (max of 10)?

...the denominator (max of 10)?

INVALID Keyword or Abbreviation

This is a valid command, the rest is extraneous.

TRANSFER FUNCTION INPUT

This is not a valid input, reenter

You cannot input a complex number for the last root. There is no room for its conjugate.

Do you want computer determined boundaries <C> or boundaries previously saved <S>...

Enter an <S> or <C>...

Enter an <S> or <C>... or a <S> to abort....

To exit, enter <S>; for more, press <CR>.

Polynomial Constants/Gain

Roots

Constant/Gain =

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 - Displays the results, transfer functions, or matrices on the screen.
FORM - 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 - Any submenu option with (*) has not been implemented yet.
STOP - Normal command for leaving ICECAPP.
SWITCHES - Flips the control switches ON and OFF.
UPDATE - Allows user to save current session to a user specified file.

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 '$' followed by a carriage return and you will be returned to the top ICECAPP menu.
        <$> #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 OLT <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

```
:LE: 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.

<S> #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.

<s> #20

**FORM COMMAND**

The FORM command is used to produce the CLTF from either the OLT or a combination of GTF and HTF. The formula for producing the CLTF is as follows:

\[ \text{CLTF} = \frac{(\text{GAIN} \times \text{GTF})}{(1 + \text{GAIN} \times \text{GTF} \times \text{HTF})} \]

This option will also form the OLT from the GTF and the HTF. The formula for that operation is:

\[ \text{OLT} = \text{GTF} \times \text{HTF} \]

<s> #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 thirteenth polynomial names the polynomial of your choice will be displayed on the screen.

<s> #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 printer 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.

<s> #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&pols.dat' prompt, and the matrix
file name in response to the 'matrix.dat' prompt. If you mix the two ICECAPPC will think there is NO data in either of
its internal files.

<§> #24

STOP COMMAND

The STOP command is used to exit gracefully from ICECAPPC. Information
is always stored in the ICECAPPC 'tf&pols.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 ICECAPPC.

<§> #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.

ADDOOT:

The ADDROOT command is used to modify or insert a root into a
polynomial. In response to the ADDROOT command ICECAPPC will provide a
menu of available polynomials. An example of the command line is:

ADDOOT ODPOLY

After you have correctly entered the name of the polynomial ICECAPPC
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 ICECAPPC will calculate the conjugate of this root. After you have
correctly input the root ICECAPPC 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 ICECAPPC will provide
a menu of available polynomials. An example of a command line is:

DELROOT ODPOLY
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 the 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.

<§> #6

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 Supress 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&pols.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 SLOWLY 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
POLYNOMIAL MANIPULATION

This function will store the second polynomial into the first and record the result in the third polynomial location.

Available polynomials are:

ONPOLY CNPOLY GNPOLY HNPOLY
ODPOLY CDPOLY GDPOLY HDPOLY
POLYA GLYB POLYC POLYD
POLYE

Enter the name of the file you would like ICECAPPC's 'tf&pols.dat' copied into. 'tf&pols.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.

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.

Your file name:

Enter the name of the file you would like to copy into ICECAPPC's 'tf&pols.dat'. 'tf&pols.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.

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.

Your file name:

FILE: HELP.TT
The number of rows or columns must be > 0 and <= 10

How many rows in your matrix (max of 10)?

How many columns in your matrix (max of 10)?

MATRIX INPUT

The user specified filename must be 8 characters or less.

The row of the location you wish to modify?

...the column?

The constant/gain cannot be equal to zero.

MATRIX MANIPULATION

This function will take the second matrix to/from the first and store the result in the third matrix location.

Available matrices are:

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.

FILE: HELP.TXT
subtract must be the same. This option has been aborted.

SINGLE MATRIX MANIPULATION

The available matrices are:

MATA MATB MATC MATD MATE

The matrix to be manipulated ...

The desired storage location ...

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 ) ...

The matrix you wish to invert is singular. This option has been aborted.

The row of the location you wish to modify ?

the column ?

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 ?

Form Command

1. Form OLTF --- OLF = GTF * HTF
2. Form CLTF ---> CLTF = (GAIN * GTF) / (1 + GAIN * GTF * HTF)
3. Form CLTF ---> CLTF = (GAIN * OLF) / (1 + GAIN * OLTF)
4. Form CLTF ---> CLTF = GTF + HTF ( In Parallel )

Your Selection ( Integer ) ...

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
CPOLY   DPOLY   GPOLY   HPOLY
POLYA  POLYB  POLY C  POLY D
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 ICECAPP 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 abort 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: MENU.TXT
TF4
TF5
POLYA
POLYB
POLYC
POLYD
POLYE
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
POLYA
POLYB
POLYC
POLYD
POLYE
ONPOLY
ODPOLY
CNPOLY
CDPOLY
GNPOLY
GDPOLY

FILE: MENU.TXT
MNPOLY
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.POLYB = 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.COPOLY = COPY
COPY.POLYA.GDPOLY = COPY
COPY.POLYA.HNPOLY = COPY
COPY.POLYA.HDPOLY = COPY

*****
; define submenus for the other options under COPY
*****
$COPY.OLTF = 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.POLYB = 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.COPOLY = COPY.POLYA

FILE: MENU.TXT
define the menu options for DEFINE OLF, 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.OLTF1 = DEFINE.OLTF
$DEFINE.OLTF2 = DEFINE.OLTF
$DEFINE.OLTF3 = DEFINE.OLTF
$DEFINE.OLTF4 = DEFINE.OLTF
$DEFINE.OLTF5 = DEFINE.OLTF
$DEFINE.POLYA = DEFINE.OLTF
$DEFINE.POLYB = DEFINE.OLTF
$DEFINE.POLVC = DEFINE.OLTF
$DEFINE.POLVD = DEFINE.OLTF
$DEFINE.POLVE = DEFINE.OLTF
$DEFINE.POLYD = DEFINE.OLTF
$DEFINE.COMPOLY = DEFINE.OLTF
$DEFINE.COMopoly = DEFINE.OLTF
$DEFINE.COMopoly = DEFINE.OLTF
$DEFINE.COMopoly = DEFINE.OLTF
$DEFINE.COMopoly = DEFINE.OLTF
$DEFINE.COMopoly = DEFINE.OLTF
$DEFINE.COMopoly = DEFINE.OLTF

; define the call routines for the matrix options under DEFINE
MATCH

FILE: MENU.TXT
.DEFINE.MATB = DEFINE
.DEFINET.MATC = DEFINE
.DEFINET.MATO = DEFINE
.DEFINET.MATE = DEFINE

$DISPLAY
OLTF
CLTF
GTF
HTF
GAIN*
BUTWRTH*
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_HTF = FREQ/RESP
; DISPLAY_FREQ/RESP_CLTF = FREQ/RESP
; DISPLAY_FREQ/RESP_GTP = 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
; GD POLY
; 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.ODPOLY = DISPLAY
DISPLAY.POLY.HNPOLY = DISPLAY
DISPLAY.POLY.HMULT = DISPLAY
DISPLAY.POLY.ADD = DISPLAY
DISPLAY.POLY.SUBTRACT = DISPLAY
DISPLAY.POLY.POLYMLT = DISPLAY
DISPLAY.POLY.SPOLYMLT = DISPLAY
DISPLAY.POLY.HELP = DISPLAY

*****
define the options for DISPLAY.MATRIX
*****
$DISPLAY.MATRIX
ADD
SUBTRACT
MATXMULT
SCLRMULT
INVERSE
TRANSPOSE
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.TRANSPOSE = 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
TFI

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.NTF = 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.MODE*  = DISPLAY
DISPLAY.NICOLS* = DISPLAY
DISPLAY.NYQUIST* = DISPLAY
DISPLAY.INVQUIST* = DISPLAY
DISPLAY.PAR/FRAC* = DISPLAY
DISPLAY.RICATTI* = DISPLAY
DISPLAY.ROUTH* = DISPLAY
DISPLAY.SPECs  = DISPLAY
DISPLAY.SWITCHS* = 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 ADDROOT DELROOT CHANGE HELP

*****

FILE: MENU.TXT
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..

UNCLASSIFIED S K MASHIKO ET AL. DEC 85
\#define the options under MODIFY.ADDRoot

\#define the call routines for MODIFY.ADDRoot

\#define the options under MODIFY.DELroot

FILE: MENU.TXT
**define the call routines for the PRINT options**

```c
null
```

FILE: MENU.TXT
FILE: PARAM.TXT
REFERENCES


8. Fontana, Robert C., Professor Emeritis. Personal interview. AFIT, Wright-Patterson AFB OH, 24 Jul through present.


16. Lorentz, Mark C., Graduate Student. Personal interview. AFIT, Wright-Patterson AFB OH, September 1985.


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
Title: Development of a Computer Aided Design Package for Control System Design and Analysis for a Personal Computer (ICECAP-PC)

Thesis Chairman: Dr. Gary B. Lamont

Abstract: 

Title: Development of a Computer Aided Design Package for Control System Design and Analysis for a Personal Computer (ICECAP-PC)

Thesis Chairman: Dr. Gary B. Lamont

Abstract: 

Title: Development of a Computer Aided Design Package for Control System Design and Analysis for a Personal Computer (ICECAP-PC)

Thesis Chairman: Dr. Gary B. Lamont
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 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.