User-Friendly, Form Fillout Data Entry for Engineering Software

J.M. Clements and T.V. Hromadka II
Advanced Engineering Software, Irvine, California

Abstract

The use of user-friendly, form fillout data entry in engineering software represents the state-of-the-art in "humanized" CRT-user communications. By making the program humanized, the learning curve is essentially minimized in that all program information is prompted and scanned for acceptability and can be rejected if data is not within program specified limits. In general, the user's manuals associated with a wide variety of batch programs are eliminated because the humanized program guides the user through every possible logic path, providing the user with various checks and controls in order to further reduce parameter selection errors and an unreasonable choice of design options. The development of the humanized software may be defined as a uniform communication/presentation (C/P) to the program user. In this paper is presented several subroutines in FORTRAN for development of the humanized data entry capability.

Introduction

The use of computers to aid in engineering analysis, synthesis, and design has increased significantly during the last decade. A main motivation for computer use is that computer studies often require (1) an iterative calculation analog such as that used in the calculation of hydraulic section information, (2) solution of a convolution type integral such as is used in unit-hydrology studies, or (3) the solution of a simultaneous system of equations such as is employed in structural analysis or water distribution network analysis. Because each of these three general classifications of problems essentially involve a repetitive series of calculations, a computer code can be prepared which will offer the engineer an extremely cost effective tool.

Another motivation for the use of computers in engineering studies is the development and widespread use of digital microcomputers. For many classes of problems, the microcomputer offers the speed and capability to the single user as does a minicomputer in a multiple user system. Consequently, programming techniques which were once limited to the microcomputer of the mainframe class of computers is now available at low cost by means of a microcomputer system.

Such programming techniques include 'humanized' computer interaction and detailed, easy to ready computer results which are explicit, fit the requirements of a reviewing agency, and yet are understandable to the first-time reviewer of the product.

By making the program humanized, the learning curve is essentially minimized in that all program information is prompted and scanned for acceptability and can be rejected if data is not within program specified limits. In general, the user's manuals associated with a wide variety of batch programs are eliminated because the humanized program guides the user through every possible logic path, providing the user with various checks and controls in order to further reduce parameter selection errors and an unreasonable choice of design options. The development of the humanized software may be defined as a uniform communication/presentation (C/P) to the program user. There are several C/P requirements for computer software which are described as follows:

1. For a sequence of data entry prompts, the pane number and calculation model description appears at the top of the page.
2. All words are written in their entirety, with abbreviations avoided wherever possible.
3. All units should be given for the information requested.
4. Allowable values are identified which limit the data entries to reasonable quantities.
5. Should the agency manuals suggest criteria for data entry, the recommendations are included on the display.
6. Any program operation commands should be consistently and uniformly displayed so that the user can operate the interaction or special data editing features without confusion.
7. A 'failsafe' line appears on the CRT screen for each page. This line is located near the bottom of the screen and appears in the same position on each page. Below the failsafe line occurs all program operating instructions. These instructions can be typed by the program user at anytime and will cause the program to respond instantly.
8. All computer-dependent requirements (such as data file manipulation) for data management should be interior of the software so the program user need not be knowledgeable of computer operations to use the program.
9. The C/P should be uniform between programs. By requiring uniformity of software interaction, individual library of software wherein each program operates, interacts, and responds identically. Consequently, the user learning curve is minimized.

By using the program which provides an optimum product, the usual design review procedure is minimized which reduces total cost to both the engineering designer and the design review team. Additionally, the computer product should be the actual fully prepared report which is to be submitted, containing the usual introductory pages, and the study results should be produced in the reviewing agency's required computer-printed forms or plotted graphs. The computer program then provides an actual engineering product, minimizing the need for secretarial and graphic efforts.

HISTORY AND CONCEPTS OF COMPUTER-HUMAN COMMUNICATIONS

In the past, the typical manner of communication with a computer was by means of punched computer cards. The cards were returned to the user along with a printed output.
which sometimes only indicated how well the data deck was assembled. The process was repeated until the desired result was obtained. This type of interaction with a computer is extremely inefficient and promotes only one direction of communication between the computer and user.

A marked improvement in computer-human communications was the utilization of the cathode ray terminal (CRT) for the data entry via the keyboard and program access. This gave the user a "hands on" approach with the computer and promoted the development of a conversational relationship. A user could now submit his own job, view the output on the terminal screen instead of waiting for a hard copy print, correct the errors if any, and re-submit the job. This type of computer-user interaction greatly increased the efficiency of the user and the cost effectiveness of the digital computer.

The interactive display presented to the user on the terminal by the application program was usually of a type called scrolling. Scrolling is presenting a line of characters or text on the bottom of the terminal screen. This line moves upwards continually as new lines are presented. This type of presentation differs from natural reading techniques such as reading a book in that the material moves upward instead of the eyes moving downward across a steady display.

When using the scroll technique in data entry situations, the user is not aware of the next input requirements until it actually appears on the bottom of the screen. If errors occur while entering data, messages are displayed and scroll up while repeat prompts for user input are again requested by the program. Multiple occurrences of errors usually result in a screen full of scrolling error messages which often confuse the user. Normally, scrolling interaction never allows a user to change a data entry once it is accepted by the program unless the user restarts the program (thus losing all previously entered data) or edits the data if the capability exists within the program.

In contrast, humanized form-fill out display interaction closely simulates natural reading or viewing characteristics. All textual information or data entry requests are assembled in logically related groups to fit comfortably on "pages". These pages are presented to the user by clearing the CRT screen of all previous information before displaying the current "page". The text information is displayed starting at the top of the screen and proceeds downward until the bottom limit of the screen is reached. The user observes a stable screen of related information in the case of data entry, related requests for user input are displayed simultaneously, enabling the user to foresee subsequent data entry requests. The cursor, which can be described as a small rectangular beam of light or pointer that emits characters on the screen, moves down the screen after each input request is satisfied.

This cursor movement is the key to powerful screen interaction. A typical terminal screen can be visualized as a matrix containing 24 lines by 80 columns or 1920 elements (see Fig. 1). One may individually address these elements by programmed movement of the cursor to the selected element. Textual sequences can be displayed anywhere on the screen at any time while retaining or erasing previous information. This allows warning signals or error messages to appear next to the data in question and disappear after the error has been corrected, thus leaving the original page of displayed information intact (see Figs. 2, 3 and 4). Another powerful use of cursor movement is the case of data entry on a page affecting the permissible values of subsequent data entries on the same page. The allowable values displayed under a subject data entry prompt can be cursor addressed, change to new values,...
and the cursor returned to the previous selection instantly. This method employs full conversational awareness by the computer system at all times, and differs from other techniques that 'disconnect' the system with the terminal until a full page of user data is filled out. This latter method is called 'form-fill out' because the system is now aware of anything that is entered on the screen until the entire page is returned for processing. This method is better than scroll but not nearly as effective as the C/P interactive method in discussion.

Since the viewing displays are constructed in 'pages', the user can manipulate the pages by a set of understandable interaction commands listed in the following:
1. TOP: request to clear the screen, re-display the page and return the cursor of the first input request on the current page. This permits modification of data on the page.
2. BACK: returns to the previous page for corrections or changes.
3. EXIT: terminates the program function in progress, closes all computer files as needed, and exits the program.

User directed page movements, coupled with the dynamic display qualities attained through programmed addressable cursor movements, provide a powerful and flexible interactive environment for experienced as well as first-time computer program users.

CONSIDERATIONS FOR INTERACTIVE COMPUTER SOFTWARE

As important consideration for selecting an interactive design method is compatibility among various computer hardware. Unfortunately, wide differences still exist between manufacturers of peripheral devices such as terminals and computer resident system programs called operating systems. Developing an interactive design methodology that is dependent upon a particular type of hardware or the operating system of a certain computer generally promotes the eventual demise of the approach. The constant change of operating systems due to computer vendor upgrades may render the programming required to accomplish such a design incompatible with the revised system.

More likely to occur is the typical change or upgrade of CRT devices or terminals which may not accommodate some of the interactive features programmed for the previous device. The hardware or operating system dependent functions can be justified in the case where the application software and hardware are bundled together to form a functional package. These types of packages normally deal with graphics applications such as CAD/CAM systems or elaborate word processing systems. The special purpose computers are mostly self-contained systems requiring certain hardware and operating system configurations.

An interactive design method has to merge comfortably with the major application systems that are already on the machine such as engineering, accounting information retrieval systems, and word processing. In order to accomplish this task and maintain compatibility among a wide selection of terminals and operating systems, a certain subset or core group of interactive functions are developed which will perform all the major tasks of a humanized form-filled interactive method. The basic functions that are compatible with over 95 percent of the terminal hardware and computer systems available today are absolute cursor addressing, clearing the screen, and ringing the bell.

Another requirement is the ability of the computer system to send out what is called control characters which excite these functions. Using these basic functions as building blocks, an entire sophisticated 'humanized' interactive approach can be accomplished while retaining compatibility across vendor lines. Changes in terminal control character sequences for various terminals are accomplished by an easily accessible hardware table contained within the interactive application driver routines or by table files. These tables map the function to the device. There are many other functions available in terminals such as highlighting or dimming of certain groups of text, flashing of warning messages, and split-screening for multiple tasks. These are not normally available in all terminals and are usually reserved for the higher priced models. In addition, the particular code sequences to start and stop these functions are widely different. These functions are just extensions of the basic three required to develop such a user friendly system. A well designed interactive system using the basic three functions of cursor addressing, clear screen, and bell satisfies fully the criteria required to produce a truly 'humanized' interactive methodology.

SCREEN LAYOUT

As discussed in a previous section, the CRT screen is divided into a cell matrix of 80 columns by 24 rows. The screen layout has been designed to accommodate most data entry possibilities. The basic skeleton screen is set up as follows: (See Fig. 5)

Fig. 5

Columns on CRT Screen

1 - 4 Blank
5 - 56 All text prompts and allowable values start in column 5 and end in column 56. Additional lines may be used to continue text
57 Blank
58 - 61 Reserved for the "*err" designation which pulls the viewers eye to where the data should be entered
62 - 72 Used for the actual data input
73 - 79 Reserved for the "ERROR" designation when input errors are detected

Rows on CRT Screen

21 Reserved for explanatory error messages or engineering warning messages
22 Contains the 'failsafe' line of dashes or underscores
23 - 24 Contains the failsafe program instructions that vary from page to page

Using this uniform layout of screen design will promote adaptability for users of other systems using the same design. Once a user knows how to manipulate one program,
the others are basically the same except for the actual data entry prompts dependent on that calculation model.

FLOW OF USER DATA

The user-entered data flows through a library of form-fill out routines where it is checked for validity, allowable ranges, and real or integer values limits before it can be accepted by the calling calculation subroutine. Detailed descriptions of the subroutines and arguments will be discussed subsequently, but the overall logical flow of user entered data through the subroutines is as follows. All input from the user is read from subroutine GETVAL. This routine is the central or main controlling routine for all input processing. Within GETVAL, various checks are made to validate the data. One set of checks determines if TOP, BACK, EXIT, or MAIN has been typed by the user. In addition, NUMCK is called to determine if a number entered by the user is valid. For example, two decimal points or a '-' or a '+' within the same value would trigger an error message. The error messages are presented by a call to ERROR within GETVAL to display the appropriate message. A call to CLEAN erases the messages on the screen after the user enters correct data. Argument values returned by GETVAL to the calling calculation subroutine are processed to allow continuation of data entry from the user or to jump to logic that will return to a previous data entry page, exit the program, erase the current page and request resubmittal, or go to the main menu.

Using the library of CRT screen handling routines reduces considerably the amount of computer code required for input processing. Since the arguments feeding GETVAL setup permissible ranges, and other parameters unique to a certain input item, the only READ statement for input (for GETVAL process) is contained within the overall logic that processes errors and displays messages contained within ERROR and CLEAN. The area or column-row address of the CRT screen where the current input item resides is passed through all the form-fill out routines in order that error messages and re-enter prompts can be addressed to the CRT correctly. In this general fashion, all input following the basic skeleton frame for the CRT 'pages' can be processed very efficiently.

SCREEN DESIGN METHOD

The design of CRT interaction pages that will be presented to the program user for data entry requires a thorough design process in itself to be effective. An approach which has been used successfully in implementing production quality software is described in this section. The first step in the screen design process is to define all input entry prompts for a given function or calculation model. This is absolutely necessary in order that the screen designer may group related data entry prompts in a logical fashion. This global definition phase of prompts also helps to eliminate redundant data entries.

The wording of the prompt for single data items or value should start with 'enter' followed by the information request text which is then followed by the 'unit' designation. Data entry prompts which require a choice among several listed options demands a different strategy. The general title or description of the options is written first. The options are indented and listed below the title. Finally, the prompt starting with 'enter' or 'select' is written which describes the value being requested.

Any descriptive text that will precede prompts excluding the actual prompt must be defined for all input requests. The descriptive text is extremely helpful to first time or occasional users. The text should be concise and offer a description of what is to follow in the prompt. (See Fig. 6). After additional explanatory text is defined, all prompts must be inspected and assigned allowable values. This range of values should be set to restrict each prompt to a typical or normal range of values that the user may enter. For option selection prompts which provides choices, only those stated values are permitted. A value outside this range represents an incorrect data response and will be handled by an error message and corresponding re-enter request for data. After all admissible values are defined, any suggested values that may help the user should be defined for each prompt. These values may be typical values given a certain criteria or possibly a county-dependent suggested value or a certain input item.

After all the data entry prompts have been defined along with any descriptive text to preclude the prompt, allowable values, and suggested values, the screen designer may now group related items together to fit comfortably on CRT 'pages' (see Fig. 7 for a sample CRT page which illustrates the suggested spacing between prompts, descriptive text and allowable values). Figure 8 illustrates the computer code to produce the CRT display screen presented in Fig. 7. This code implementation may be used as a guide for development of a complete system of user-friendly or humanized form-fill out data entry displays.

SUBROUTINE DESCRIPTIONS AND LISTINGS

SUBROUTINE CINIT

Abstract

CINIT initializes the terminal control tables for functions of clear screen, cursor addressing, and bell.
CALL CURSOR
CALL CURSOR(11)
CALL CURSOR(13)
WRITE/X70
WRITE/X79
CALL ALLOCA(10,6,0000)
WRITE/INP,INP
CALL CURSOR(0.1)
WRITE/INP,INP
CALL ALLOCA(10,6,0000)
WRITE/INP,INP
CALL CURSOR(13.1)
WRITE/INP,INP
CALL ALLOCA(35,6,0000)
WRITE/INP,INP
Fig. 8

Description
The routine contains all codes necessary to set up the screen functions. After the control codes have been determined for the terminal selected, these values are then loaded into the IP and ICLEAR arrays. The IP array contains the ASCII codes required for positioning the cursor from 1 through 60 on the CRT screen. ICLEAR array contains the ASCII codes to clear the screen. IESC is loaded with the lead in character for the cursor addressing sequence. JBELE contains the code to excite the bell function on the terminal. The terminal codes set up in the listing are compatible with Lear Siegler, Televideo, and SOROC terminals. The ASCII values of the terminal functions are stored in common block CURS for availability throughout the library of screen subroutines. Routine is called once to set up the codes.

Arguments
None.

SUBROUTINE CURSOR

Abstract
Positions the cursor anywhere on the CRT screen.

Description
Receives arguments of column, row from calling subroutine to position the cursor before a write is performed on the screen. The column, row argument is used as indexes to the IP array table to select the appropriate ASCII codes. The write statement submits the ASCII string of characters to the screen for cursor positioning. The Z (computer dependent) in the format statement holds the cursor after positioning for subsequent writes or prompts. Some systems will not need the Z type of specification in the format statement.

Arguments
IX (IN) Column number (1-60)
ICY (IN) Row number (1-24)

SUBROUTINE GETVAL

Abstract
Central data input capture routine which gets and returns a legal value from user input.

Description
Input is received in an 11-character buffer (BUF) in Alpha format. The buffer is first checked for any legal function such as TOP, BACK, or EXIT. Depending on MODE, certain functions will not be legal at certain times. NUMCK is called to transform the alpha characters into a legal real value. If an error occurs, ERROR is called to display a message. The logic remains in GETVAL until the error is corrected, after which CLEAN is called to erase the previous error messages.

Arguments
IX (IN) Column number to read on CRT screen
ICY (IN) Row number to read on CRT screen
INT (IN) 0 = Read data input anticipated
1 = Integer data input anticipated
FMIN (IN) Minimum allowable value passed to NUMCK
FMAX (IN) Maximum allowable value passed to NUMCK
VAL (OUT) Legal, checked real value returned to calling routine
NCOND (OUT) 1 = EXIT input by user
2 = TOP input by user
3 = BACK input by user
4 = LINE input by user
Values returned to calling routine for appropriate action
MODE (IN) 1 = allow EXIT
2 = allow EXIT, TOP
3 = allow EXIT, TOP, LINE
NULL (IN) 0 = do not allow blanks as input
1 = allow blanks as input, but set VAL = 0 at this occurrence

SUBROUTINE NUMCK

Abstract
Extensively check a data input value (numeric) for legal syntax.

Description
The user data input is received in alpha format in buffer KFDL. KF DL is then parsed by character by character to determine a legal numeric form. If the form is legal, KFDL is transformed by ENCODE and DECODE operations to a REAL value. If an integer is expected, range checks are performed to assure an integer between -32767 and 32767.

Arguments
KFDL (IN) Alpha buffer containing user data input
LENTH (IN) Length of KFLD, usually 11
VALUE (OUT) Real value returned to calling routine
INTG (IN) Error flag, set to various values if an error occurs
NERR (OUT) Error flag, set to various values if an error occurs

SUBROUTINE ERROR

Abstract
Contains a set of error messages which appear in a reserved row on the CRT screen for illegal data entries.

Description
Displays any of three error messages in row 21. The 'ERROR', and 'RE-ENTER' designations appear in columns 73-79 of the row of the data entry prompt and columns 62-70 of the row immediately below the input field. Selection of the particular error message displayed is controlled by GETVAL.

Arguments
ITYP (IN) Error message type to display
Y (IN) Row number of current data entry prompt
ERF (OUT) Set to 1 when error message is displayed. CLEAN will reset it to 0 after the messages are erased

SUBROUTINE CLEAN

Abstract
Erases and cleans the screen of previous error messages.

Description
Routine is called by GETVAL when a user input error has been corrected. CLEAN clears the 'ERROR', 'RE-ENTER', and any message residing in row number 21 of the CRT screen.

Arguments
Y (IN) Row number of current data entry prompt
ERF (OUT) Set to 0 upon exit to indicate messages have been cleared

SUBROUTINE BELL

Abstract
Produces an audible 'beep' to the terminal when a user input error is detected.

Description
Routine is called by ERROR. The ASCII bell code contained in JBELL is initialized by CRINIT.

Arguments
None

SUBROUTINE CLRSCR

Abstract
Clears and erases all information on the CRT screen.

References
PROGRAM 1

SUBROUTINE CHARR

Routine to initialize the cursor control.

FUNCTIONS OF CTRL key:
- E0: Vertical scroll.
- E2: Horizontal scroll.
- E4: Backspace.
- E8: Tab.
- ED: Cursor lock.
- EF: Cursor unlock.

CALL CLEAN(1,1), CALL CLEAN(7,1)

RETURN

END

PROGRAM 2

SUBROUTINE CURSOR(IN,OUT)

Routine to position the cursor anywhere on the screen.

CALL CLEAN(1,1), CALL CLEAN(7,1)

RETURN

END

PROGRAM 3

SUBROUTINE GETVAL(IN,OUT)

GUTS AND RETURN A LEGAL VALUE FROM INPUT.

VALUE IS STUFF CHECKED, SIMM CHECKED, AND CHECKED FOR
OVERFLOW IN THE INTEGER CASE.

INTEGER BUF(128), EXIT, TOP, BACK, NT, ESP, BLK

CALL CORRECT(NT)

DATA EXIT/4, TOP/7, BACK/7, NESL/7, "/", "N", "A", "C"

MCOND<0 ESP<0

CALL CORRECT(IN)

READ(NT,200) BUF

IF BUF(B1) .NE. EXIT GO TO 10

IF MODE(L2) .GE. 55 MCOND<0 GO TO 999

GO TO 999

IF BUF(1).NE.TOP GO TO 30

IF MODE(L2) .GE. 55 MCOND<0 GO TO 999

GO TO 999

IF BUF(1).NE.BACK GO TO 70

IF MODE(L2) .GE. 55 MCOND<0 GO TO 999

GO TO 999

CALL NUMBBUF(IN,VAL,INT,HEBT)

IF(HEBT.1) GO TO 40

IF MODE(L2) .GE. 55 MCOND<0 GO TO 999

GO TO 999

CALL NUMBUF(IN,VAL,INT,HEBT)

IF(HEBT.1) GO TO 40

IF MODE(L2) .GE. 55 MCOND<0 GO TO 999

CONTINUE

CALL ERROR(1,1,ESP)

GO TO 9

IF VAL. GE. FINH .AND. VAL .LT. FKMA: GO TO 10

CALL ERROR(1,1,ESP)

GO TO 10

CALL ERROR(3,1,ESP)

LC TO 10

IFERR(ED,1): CALL CLEAN(1,1), CALL CLEAN(7,1)

CONTINUE

RETURN

END

PROGRAM 4

SUBROUTINE HUNKR(FLD, LEN, VALUE, INPTR, HHERE)

FREE FORMAT EXTRACTION FROM A1-ARRAY (FLD)

FLD A1-ARRAY CONTAINING THE TARGET STRING
LEN LENGTH OF FILD ARRAY
VALUE RETURNS A REAL VALUE RECIVED FROM TARGET STRING
INPTR INTEGER FLAG
HHERE ERROR FLAG

SUB INTEGER CHARACTERS IN STRING

HHERE ERROR FLAG

DIMENSION FILD(LENGTH), INPTR(15), TYPE(4)

DATA TER END

HHERE = 0

VALUE = 0

IF (LENGTH .LE. 0) GO TO 1000

SCAN FIELD TO DELINITE NON-BLANK CHARACTERS

DO 100 X = 1,LENGTH

IF(INPTR(X) .NE. 1) GO TO 1000

END

SCAN FOR VALID REAL OR INTEGER CHARACTERS

IF (NUM = 0 .OR. INCR = 0) GO TO 200

IF (END = 0) GO TO 200

END

MICROSOFT FOR ENGINEERS, 1986, Vol. 2, No. 1 19
PROGRAM 5

C DISPLAYS ERRORS MESSAGES FOR ILLEGAL USER INPUT
C
I
CTE
GINT J, EXP
CALL CURSOR(1,7)
WRITE(WT, 630)
C CURSOR AT ERROR INDICATOR
CALL CURSOR(1,11)
WRITE(WT, 604)
C CURSOR AT MESSAGE
CALL CURSOR(1,11)
IF(TYP.EQ.2) WRITE(WT, 600)
IF(TYP.EQ.3) WRITE(WT, 605)
C CLEAR INPUT FIELD
CALL CURSOR(1,1)
WRITE(WT, 606)
C
I
C FOR M A K E S

C PROGRAM 6

C SUBROUTINE CLEAR(I, EXP)
C
C CLEAR THE ERROR MESSAGES AFTER USER ERROR IS CONNECTED
C
I
CTE
GINT J, EXP
CALL CURSOR(1,7)
WRITE(WT, 630)
C CURSOR AT ERROR INDICATOR
CALL CURSOR(1,11)
WRITE(WT, 604)
C CURSOR AT MESSAGE
CALL CURSOR(1,11)
WRITE(WT, 605)
C FOR M A K E S

C PROGRAM 7

C GENERAL NAME FUNCTION: INITIALIZED BY CRMIT
C
I
C
GINT J, EXP
CALL CURSOR(1,7)
WRITE(WT, 630)
600 FORMAT(A, I)
C
I
C
C SUBROUTINE ALLOWS (WT, I, 18, 192)
C
C DISPLAYS ALLOTTABLE VALUE MESSAGES
C
I
DIRECTION I, I, I
I
C
GINT J, EXP
CALL CURSOR(1,7)
WRITE(WT, 630)
C CURSOR AT PLACEHOLDER OF ALLOTTABLE MESSAGE
CALL CURSOR(1,11)
IF(TYP.EQ.1) WRITE(WT, 601)
IF(TYP.EQ.2) WRITE(WT, 602)
IF(TYP.EQ.3) WRITE(WT, 603)
IF(TYP.EQ.4) WRITE(WT, 604)
IF(TYP.EQ.5) WRITE(WT, 605)
IF(TYP.EQ.6) WRITE(WT, 606)
IF(TYP.EQ.7) WRITE(WT, 607)
C FOR M A K E S