ENHANCING FORMS

by Alvin Bruce Charity, I
and
Katherine Joan Dante

ABSTRACT

Screens can make or break a system. Attractive screens help sell the system to the user. Neat, logical screens can increase data entry rates, improve data entry quality, and present that "professional" image to the user. This paper will demonstrate how to turn screen enhancements on and off programmatically, and how to create quality screens using the line-drawing character set.

Examples will be provided which demonstrate the improvement in screen legibility and show how to incorporate the necessary es- cape sequences in COBOL, FORTRAN, and SPL programs.

This paper is meant for programmers and analysts who use VPLUS and want to improve the legibility of their screen design.

PART I USING FORMSPEC

INTRODUCTION

We hear the term "user-friendly" a lot: user-friendly systems, user-friendly computers, user-friendly screens. When we try to use these "user-friendly" what-evers, we quickly find that "user-friendly" is as hard to understand as non-user-friendly. A truly user-friendly screen would offer the user a day off with pay!

We aren't going to show you how to make user-friendly screens. That's like making good-tasting medicine. If it is done right, you then have to put on child-proof caps. No, what we are inter- ested in is legible screens. Screens that don't scramble your eyeballs when you try to read them. If the screens are used to capture data from pre-printed forms, we want them to resemble those forms as closely as possible (and have you ever seen a user-friendly form?)

LINE-DRAWING

One thing that amazes me about Hewlett-Packard is its lack of bragging. If you do not happen to read the manuals that come with the terminals, you might not know that a line-drawing character set even exists. And the only way to find out whether it works with FORMSPEC or not is to use it.

We tested it out, found that it worked, and began to use it to improve our forms:

The first system we installed on our HP was a personnel system. Now, as everyone in the federal government knows, our personnel system is fueled with the Standard Form 50, familiarly called the SF50. Nothing can be done without first filling in a request for a personnel action, so that the personnel office can respond with the SF50. The National Science Foundation has been producing these SF50's from the computer since the seventies.

Our first data input screens, designed on a Honeywell 6060N, were extremely primitive, formatted more for the convenience of the computer than for human beings. Data elements were listed on the left of the screen and the correct values were entered on the right. Errors were indicated with an asterisk or error code by the data element name.

On moving to the HP, we tried to make the screens look as much like the form as possible. However, without lines to separate the different parts of the form, the screen was confusing. We tried separating the elements using the line-drawing character set and--Voila! a clean screen, easy to read and equivalent to the actual form. Because the lines are distinct from the written material, they can be used without extra spacing. This helps minimize the number of screens needed.

ACCESSING THE LINE-DRAWING CHARACTER SET

Alternate character sets can be accessed either through function keys or through escape sequences. On some terminals, they can be accessed only through escape sequences. The sequence <ESC>)B will make character set B, which is usually configured as the line-drawing set, the alternate character set. Your terminal's manual will show you the proper keys for various line formats. Although many terminals treat upper- and lower-case alphabetic characters the same when translating to the line characters, use only upper-case alphabetics for those terminals which are case sensitive. Once you have established the alternate character set, use <CTR>N and <CTR>O to switch between it and the ASCII set. (N for new, O for old.) You can flip between <CTR>N and <CTR>O as often as necessary to get the exact effect you want. Remember that a line at a time is converted, not the entire screen.

If your terminals do not have the line-drawing character set, ask your HP representative for the simple field upgrade that will provide it.

ENHANCEMENTS

Perhaps because Hewlett-Packard started out making tools for scientists, it appears to leave much of its capability for the user to discover. Fortunately, our division is blessed with several people who, even under tight deadlines, have the curiosity to try various HP features in new situations. For ex- ample, FORMSPEC mentions underlining, half-bright, blinking, and inverse video, but only for the data entry fields. It was up to us to find that these could also apply to screen text.

To make matters more difficult, the HP2645A, sold as the program- mer's terminal, does not give you access to these screen enhance- ments through the function keys; while the HP2624B, billed as the data entry terminal, does.

Interestingly, when using FORMSPEC on the HP2624B, you can gain access to these enhancements only while designing a screen! It seems that HP did intend these to be used in form design, it just was not made clear in the HP documentation.

When using a terminal without function key access to the screen enhancements, use <ESC>&d followed by the appropriate code chosen from the table given in the "Display Enhancements" section of your terminal manual. A more detailed look at these escape sequences will be given later.

The first use of these enhancements was to emphasize parts of a form. Even with the line-drawing set, the legibility of forms can be improved with full- and half-bright inverse video. We first used full-bright inverse video only to high-light sections of a form. (Blinking, unlike other enhancements, appears to ir- ritate the user more than help.)

SCREEN STANDARDS

With our expanded knowledge of HP capabilities, we began to standardize our screen formats. Our current standards require the screen be divided into three sections:

1. A title line, which defines the screen and provides date and time
2. A data entry or retrieval portion, which displays data and asks for input

3. An instruction portion, which tells the user which key to press and what data to enter

The different sections of the form are separated with lines. The instructions are emphasized with the use of half-bright inverse video, set up in the three columns: TO do, ENTER, and PRESS, with each column heading underlined. (In some cases, space can be saved on the form, without losing legibility, by dividing the in- struction section in the middle of the screen, putting directions in both halves.)

The title line will be helpful if there are ever any problems, undiscovered bugs, in the production programs. It contains the date and time the screen was invoked, as well as the title of the system and a screen identification which enables the programmer to find exactly where the program may have gone wrong--of course, our programs never fail!

The instruction section solves the problem of not all Hewlett-Packard terminals supporting the display of function key labels. (In fact, the ones we supplied our data entry people, the HP2645A's, lack this feature.)

CONCLUSION TO PART I

Using screen enhancements and the line-drawing character set in FORMSPEC, we have seen how to produce clean, legible screens. In the next part of this paper, we will learn how to change the en- hancements of the data-input fields in application programs.

PART II PROGRAMMATIC CONTROL OF TERMINAL ENHANCEMENTS

As just described, the terminal en- hancements are alive and well. Now that we know they exist, let's take it a little further. Did you know that these features can be programmatically con- trolled? Well, the answer is they can! So now we want to make you the programming "Gurus" that you think you are. How can this be done? Well, it's a piece of cake.

Most often online applications are designed using character prompts or a screen writer package. For this discus- sion the screen writer package is VPLUS (V3000). There are occasions in many online applications where it's desirable to turn on and off data field enhancements. This feature gives the online applica- tion what we consider to be "Terminal Special Effects". These terminal special effects can show various ways the information is dis- played to the user:

1. required data values,
2. required data values condi- tionally based on other data values,
3. data values in error,
4. key values for the process or
5. other reasons as defined by the application.

Using terminal enhancements, we can add a touch of "pizzazz" to the user/computer interface. But remem- ber, nothing is free. in this lovely world of computers; it will cost just a little in computer resources to employ these fancy features. However there should not be any noticeable change in the user response time.

WHAT ARE THE TERMINAL EN- HANCEMENTS?

Terminal Enhancements are a stan- dard feature of your HP ter- minals. These enhancements are ways to dis- play characters and background, which consist of the following:

```
- Security Video - the character display is suppressed
                   when entering or displaying data to the
                   terminal.  This enhancement is used in
```

conjunction with fields for which passwords or similar security-sensitive data must be entered through the keyboard.

- Inverse Video - black characters are displayed against a full-bright white background.

- Half-Bright Video - characters (or background for inverse video) are displayed at half intensity.

- Blinking Video - characters repetitively blink on and off.

- Underline Video - characters are underscored.

These enhancements may be used separately or in any combination. When used, they cause control bits to be set within the display memory of the terminal. If the content of the display memory is subsequently transmitted to a host computer, these control bits are translated into escape sequences which are transmitted along with the displayable text characters.

From the keyboard, you can enable and disable the various video enhancements using the "enhance video" set of function keys. This is accomplished by pressing the "AIDS" function key followed by the enhance video function key.

Programmatically these enhancements can be enabled or disabled by embedding escape sequences within the data. The general form of the escape sequence is as follows:

```
<ESC>&d<enhancement code>
```

where the enhancement code is an @, s, or S or one of the upper- case letters A through O specifying the desired enhancement(s). Some of these enhancements are defined as follows:

- Inverse Video = ESC&dB
- Half-Bright Video = ESC&dH
- Underline = ESC&dD
- Blinking = ESC&dA
- Security = ESC&dS
- Security with the Inverse Video = ESC&dsB
- End enhancement = ESC&d@

Additional information about the terminal enhancement features can be obtained in the "DISPLAY CONTROL" section of your HP ter- minal manual.

HOW TO PROGRAMMATICALLY INVOKE TERMINAL ENHANCEMENTS

Various methods for turning on and off the video enhancements will be discussed. For the sake of simplicity most of the em- phasis is focused on the inverse and half-bright video. Let's start by setting the stage with a sample application.

Sample Application

This sample application is for a personnel system. Our task is to capture and update employee data using an employee-data form. The data contained on the form will be:

1. Last-name
2. Rest-of-name
3. SSN
4. Sensitivity
5. Date-of-birth
6. Veteran-preference
7. Sex
8. Citizenship
9. Effective-date
10. Position-code
11. Nature-of-action-code

All of this data is required at initial capture time but can be updated later.

CASE 1 - USING ENHANCEMENTS WITH VPLUS AND NO ESCAPE SEQUENCES

The Case 1 example will use a VPLUS (V3000) application to ini- tially capture the employee data. All required fields will be shown in inverse video. Once the required data is captured from the form, the data will undergo edits. Data values that don't pass the edit criteria will remain in inverse video, while values that passed will be set to half-bright inverse video on the screen.

One way to show the required data fields in inverse video is to define the employee-data form in VPLUS via FORMSPEC. The draw- back of this method is that the fields will remain in inverse video after the data has passed edits, which counters the philosophy behind the use of inverse video versus half-bright in- verse video. Since we want the required data values to be placed in half-bright inverse video after passing the edits, we can ac- complish this by defining all data fields as half-bright in VPLUS and programmatically controlling the inverse video when needed. The VPLUS error enhancement will be set to inverse video via FORMSPEC.

Case 1 suggested programmatic code:

The following code consists of only the VPLUS intrinsics to show how to paint (or print) the employee data form to the user's ter- minal. This scenario assumes that the form name and other data have already been initialized.

1. CALL "VOPENFORMF"
2. CALL "VOPENTERMINAL"
3. CALL "VGETNEXTFORM"
4. CALL "VINITFORM"
5. CALL "VSHOWFORM"

At this point the terminal is painted with the enhancements as set in VPLUS via FORMSPEC--half-bright inverse video. This is not what our specifications called for.

The following shows two additional VPLUS calls that can be used to turn on inverse video for all required data fields.

1. CALL "VOPENFORMF"
2. CALL "VOPENTERMINAL"
3. CALL "VGETNEXTFORM"
4. CALL "VINITFORM"

4.1 CALL "VEDITFIELDS"

4.2 CALL "VPUTWINDOW" USING COMAREA, MESSAGE, MSGLEN. (where MESSAGE contains all blanks)

5. CALL "VSHOWFORM"

The execution of the VEDITFIELDS intrinsic will cause all required data fields to be flagged with an error, since the fields have been defined in the form as required data and no data has been entered. The VEDITFIELDS intrinsic uses the error en- hancement as defined in VPLUS which is inverse video. The use of the VPUTWINDOW intrinsic will suppress or blank out any error message to the user at this time.

If any data errors occur during the user's entry process, only those data fields in error will remain with the inverse video en- hancement. The data fields that passed the edits will be placed in half-bright inverse video.

This example illustrated the use of the inverse and half-bright video with VPLUS and requires no additional programming using the escape sequences.

CASE 2 - USING ENHANCEMENTS WITH VPLUS AND THE ESCAPE SEQUENCES

The Case 2 example will require the program to conditionally turn on and off the enhancements based on certain data statuses. For example if the nature-of-action code changes on the employee-data form, additional data must be updated in our personnel system. This will require a second form, entitled the nature-of-action form, be displayed. This form will consist of:

1. Name-of-office
2. Location-of-office
3. Position-title
4. Salary
5. Grade

When the nature-of-action code is a "P" (for promotions):

- position title
- salary
- grade

will require updates. This means that these data values should be highlighted with the inverse video enhancement prior to the form being painted.

On the other hand, if the nature-of-action code is a "D" (for detail assignments), this will require:

- name-of-office
- location-of-office

to be updated, and they should be displayed with the inverse video enhancement prior to painting the form.

For both of these situations, the program will retrieve the data from the personnel data base for the nature-of-action form prior to painting the screen. Again, once data values pass the edits, any subsequent form-painting should be in half-bright video for those fields only.

Case 2 - Suggested VPLUS Interface

One solution is to have separate forms showing the different fields with the inverse video enhancement via the FORMSPEC definitions; but we need to set them to half-bright for sub- sequent painting once they pass the edit criteria. Also, using multiple forms to show the different enhancements could possibly cause our forms file to get extremely large, especially for large or complex applications.

The following code depicts the VPLUS calls required for painting the nature-of-action form to the terminal:

1. CALL "VGETNEXTFORM" (using the data retrieved from the personnel data base)
2. CALL "VPUTBUFFER"
3. CALL "SHOWFORM"

By adding the VSETERROR intrinsic, we can programmatically turn on the inverse video enhancement for those data values requiring update. For example, the nature-of-action-code "D" change would require the following fields be enhanced:

- name-of-office - field #1
- location-of-office - field #2

The VPLUS calls would then look like this:

1. CALL "VGETNEXTFORM"
2. CALL "VPUTBUFFER"

   2.1 CALL "VSETERROR" USING COMAREA, FIELD#1, MESSAGE, MSGLEN

   2.2 CALL "VSETERROR" USING COMAREA, FIELD#2, MESSAGE, MSGLEN (where MESSAGE contains all blanks)
3. CALL "VSHOWFORM"

Resulting from this, the name-of-office and location-of-office will be shown with the inverse video enhancements and the cursor will be positioned at the first field requiring a data change.

Case 2 suggested code using VPLUS and escape sequences:

Using the escape sequences to turn on inverse video for the Case 2 example would require the data buffer used for the VPUTBUFFER to carry these enhancements. As VPLUS users already know, the VPUTBUFFER is a mirror image of the VPLUS form. So now we can define a field or fields with the enhancements we want to use. This can be accomplished as follows:

- Inverse-video-enhance value "ESC&dB"
- Half-bright-enhance value "ESC&dH"

In our program we would:

```
COBOL   - move inverse-video-enhance to field#1, field#2
```

```
SPL     - field#1(*)inverse-video-enhance
          field#2(*)inverse-video-enhance

FORTRAN - field#1=inverse-video-enhance
          field#2=inverse-video-enhance
```

In this solution, the cursor will be positioned at the first changeable data field and not necessarily at the first required data field. (In this example, the first changeable data field is the first required data field.)

The examples presented are just some ways to use the enhancements for terminal special effects. Using your own creative ingenuity and further research, you will find many ways to enhance applica- tions using these features.

Pretty user friendly huh!!

Don't worry, as you use these concepts, they will become programmer-friendly.