Using Screens in COBOL

Screens allow a program to be interactive. Data can be entered at the screen by the user for use in the COBOL program and data can be displayed on the screen for the user. The data that is taken in can be used for all kinds of processing including creating and updating files. Data that is stored on files can now be accessed and viewed on the screen at the request of the user.

This interactively is vital to the successful processing of data in much of today's information based society. Editing of keyed input data is vital to successful processing of data provided through screens. Each entry that the user makes must be thoroughly checked for accuracy before the data is stored on a file. If the user keying in the data is a knowledgeable user, the program can edit the data interactively, inform the user of data errors and give the user a chance to make corrections. If the user is not familiar with the data being keyed, the programmer may decide to simply prepare a report of errors for later correction or reentry. The critical fact is, all data being keyed in must be valid or the integrity of the system is destroyed, therefore screen programs that allow data entry destined for files frequently include editing modules. The only exception would be data keyed into to create a file that will be edited before progressing into the system.

Setting up the Screen Section

When the programmer needs to set up screens in a COBOL program, a new section in the DATA DIVISION called the SCREEN SECTION is used to describe the screen.. The SCREEN SECTION has different rules from the rest of the DATA DIVISION - rules designed for laying out screens. The screen has a layout that is 80 characters wide and 20 to 25 lines deep. When you layout a record for a disk or a line on the printer, the layout is accumulative. That is, if you want to know what characters a field occupies on the disk, you need to add up all of the characters that proceed it. For example, if there are 25 characters described before the field the field starts in character 26 and if it has a PIC of X(5) then the field occupies characters 26, 27, 28, 29 and 30.

In the screen section the location of a field is determined by two new clauses LINE and COL. LINE 5 COL 20 means this field will appear on the screen on the fifth line starting in the twentieth character. If the programmer is printing something like a screen header or the words to identify a field, the literal is setup using a VALUE clause and the LINE and COL entry determine the starting point. There is no need for a field name. The length of the literal is not stated, the length is determined from the VALUE clause.

```
            05  VALUE "DATA ENTRY SCREEN"    LINE 1  COL 35.
```

The programmer needs to be careful to make sure that fields do not run into each other. For example, if the programmer put a VALUE "DATA ENTRY SCREEN" at LINE 1 COL 35, this would use columns 35 through 51. The programmer needs to calculate this so another field is not placed in the same area.

Dealing with data requires a little more setup. For example, the field in the SCREEN SECTION that would allow the user to key in the identification number might be set up like this:

```
       05  ID-INPUT                          LINE 5  COL 10
                     PIC X(4)  TO ID-IN-WS.
```

The ID-INPUT name is simply the name of the field on the screen, it is not the place where the data will actually be stored. The PIC X(4) sets up the screen to allow the user to enter up to 4 characters of data in the field. Data that is to be keyed in at the screen will be stored elsewhere in the DATA DIVISION. For example, the programmer could define the field ID-IN-WS in WORKING-STORAGE:

```
       05  ID-IN-WS   PIC X(4).
```

The TO clause indicates that data is being keyed in and will be moved to the specified location when the data take-in function is complete. The setup just illustrated would allow the user to key in 4 characters of data at LINE 5 COL 10. Because of the TO clause. these four characters of data would be stored in WORKING-STORAGE under the name ID-IN-WS. This data is now available to the programmer to use in any way that is needed.

In addition to the TO clause, COBOL also supports a FROM clause that is used to display data on the screen and a USING clause that displays data and allows the user to modify the data.

In the following example, the WORKING STORAGE and SCREEN SECTIONS are shown for a program where the user is going to key in an identification number and a name which will be stored in Working Storage (In this example, the section where the data is going to be stored is called DATA-FROM-SCREEN). The setup of the Screen Section has some distinct differences from the usual DATA DIVISION setups. The example below shows the layout of a screen designed to take in keyed information. An explanation follows:

Example #1:

```
WORKING-STORAGE SECTION.
01  RESPONSEZ.
    05  RESPONSE-IN-WS  PIC X        VALUE "C".
01  DATA-FROM-SCREEN.
    05  ID-IN-WS        PIC XXXX     VALUE SPACES.
    05  NAME-IN-WS      PIC X(20)    VALUE SPACES.

SCREEN SECTION.
01  DATA-ENTRY-SCREEN.
    05  VALUE "DATA ENTRY SCREEN" BLANK SCREEN     LINE 1 COL 35.
    05  VALUE "ID #"                               LINE 3 COL 10.
    05   ID-INPUT                                  LINE 3 COL 25
                  PIC  X(4)     TO ID-IN-WS.
    05  VALUE "NAME"                               LINE 5 COL 10.
    05  NAME-INPUT                                 LINE 5 COL 25
                  PIC X(20)     TO NAME-IN-WS.
    05  VALUE "C - TO CONTINUE"                    LINE 11 COL 30.
    05  VALUE "Q - TO QUIT"                        LINE 12 COL 30.
    05  VALUE "ENTER RESPONSE"                     LINE 14 COL 30.
    05  RESPONSE-INPUT                             LINE 14 COL 45
                  PIC X         TO RESPONSE-IN-WS.
```

- The first line under 01 DATA-ENTRY-SCREEN contains the title of the screen which will start at COL 1 in the 35 character and use as many characters as needed. Notice that this line also contains the clause BLANK SCREEN which assures that you start off with a clean screen. No PIC clause is needed when you are simply printing a literal like the title on the screen, you simply tell it where to start and it uses as many characters as necessary to display the entire literal in the VALUE clause.
- This screen is set up to allow the user to enter data in the screen which will then be stored in memory (in WORKING-STORAGE) for the programmer to use. When data is to be taken in I give the field a name just to show data will come in, then I give it a PIC to show how many characters will be allowed and what type of character. Giving give something a picture of 999, setups the screen to allow 3 characters to be entered and it will only allow the user to type numeric characters. If the picture is XXXX then 4 characters of any kind will be allowed. The place that the data is stored is given after the TO clause. As you can see from the example, I set up places in WORKING-STORAGE to hold the data and then put that field name in the TO clause to direct the data that the user types in to the appropriate field.
- Finally at the bottom of the screen, I tell the user what to enter when they have finished with the screen and want to move on. In my sample they can enter a C to continue or a Q to quit. (You should note when we look at the PROCEDURE DIVISION that if you enter data and then press Q, the program is set up so the data will not be processed - if you plan to enter Q you must move through the fields by pressing tab and not entering data).
- As indicated above, when the user enters data in the example, they will key the data and then press TAB to move to the next field. When they have entered Q or C, they will press ENTER to indicate that the screen is complete. This is FULL SCREEN PROCESSING where TAB lets you move through the screen, but the data is not actually taken in until the user presses ENTER.
- In my example, I only used the TO clause which takes in data. If you want to display data that is currently in stored in a field in memory, then the FROM clause can be used and if you want to display data from memory and allow the user to change it then the USING clause should be used.

Full Screen processing

There are two kinds of screen processing that can be done in COBOL. FULL SCREEN processing puts up the entire screen, allows the user to enter all of the data by tabbing from field to field and finally the programmer takes in and processes the FULL SCREEN of data when the ENTER key is pressed.

When you want to put the data up on the screen as a full screen (FULL SCREEN processing) you use the DISPLAY verb and display the 01 name of the full screen. In my example, this would be written as:

```
DISPLAY DATA-ENTRY-SCREEN.
```

When the user has finished keying in the data the program wants to take that data in - again this example takes in the full screen of data containing all the data that the user keyed in and moves the data to the storage area in WORKING-STORAGE. The verb that is used is the ACCEPT verb. In FULL SCREEN processing, the data is taken in by accepting the 01 name of the full screen.

```
ACCEPT DATA-ENTRY-SCREEN.
```

Up until now, when the programmer has wanted to bring in data, it has been READ from a file. Screens present an alternative. Instead of reading data from a file, the program can display a screen and take in the data that the user keys. In this case, the combination of the DISPLAY and ACCEPT bring in data just as with files the READ brings in data. In writing a program that uses screens, the program construction will continue to use the concept of initializing code to bring in the first data transaction and then code at the bottom of the loop to handle all other data transactions. Therefore, the program will need an initializing DISPLAY/ACCEPT combination that will bring in data prior to the start of the loop and another DISPLAY/ACCEPT combination that will handle bringing in data at the end of the processing loop. (Again note, the DISPLAY/ACCEPT combination is being used in the same way that the READ is used in traditional file processing.)

In the traditional program, the processing continues until end of file is reached. When data is coming in on the screen, processing should continue until the user decides to terminate by entering a particular code (in the example, the code is Q for Quit). This means that the termination of the program is not based on the end of file test but rather on the user entering the Q on the screen. Therefore the perform of the loop is done until the response field contains the Q. The following is a comparison of file and screen processing:

FILE PROCESSING

```
WORKING-STORAGE SECTION.
01  INDICATORS.
    05  EOF-IND   PIC XXX
                    VALUE "YES".
PROCEDURE DIVISION.
...
...
B-100-PROCESS.
          READ INPUT-FILE
        AT END
           MOVE "YES" TO EOF-IND.
    PERFORM B-200-LOOP
        UNTIL EOF-IND = "YES".
B-200-LOOP.
    ... processing ...
    READ INPUT-FILE
        AT END
           MOVE "YES" TO EOF-IND.
```

SCREEN PROCESSING

```
WORKING-STORAGE SECTION.
01  RESPONSES.
    05  RESPONSE-IN-WS  PIC X
                    VALUE "C".

PROCEDURE DIVISION.
...
...
B-100-PROCESS.
    DISPLAY DATA-ENTRY-SCREEN.
    ACCEPT DATA-ENTRY-SCREEN.
    PERFORM B-200-LOOP
      UNTIL RESPONSE-IN-WS = "Q".
B-200-LOOP.
    ... processing ...
    DISPLAY DATA-ENTRY-SCREEN.
    ACCEPT DATA-ENTRY-SCREEN.
```

In the next example, the screen is being used to display data from a file. The data is read from a disk file and then displayed on the screen. (Compare with the processing to read a record and print the information on a report). Again, the program will use FULL SCREEN processing. When the user has finished viewing the data on the screen, a response must be entered to resume processing. Therefore, the DISPLAY/ACCEPT combination is used. The DISPLAY shows the information on the screen and the ACCEPT takes in the user response.

Example #2A:

This example displays the information on an input record on the screen. The input record has an identification number and a name. In the program the record would be read from the disk and be displayed for the user to view using the screen section.

```
DATA DIVISION.
FILE SECTION.
FD  INPUT-FILE.
01  INPUT-REC.
    05  ID-IN           PIC XXXX.
    05  NAME-IN         PIC X(20).
WORKING-STORAGE SECTION.
01  RESPONSEZ.
    05  RESPONSE-IN-WS  PIC X        VALUE "C".

SCREEN SECTION.
01  DATA-ENTRY-SCREEN.
    05  VALUE "DATA ENTRY SCREEN" BLANK SCREEN     LINE 1 COL 35.
    05  VALUE "ID #"                               LINE 3 COL 10.
    05   ID-INPUT                                  LINE 3 COL 25
                  PIC  X(4)     FROM ID-IN.
    05  VALUE "NAME"                               LINE 5 COL 10.
    05  NAME-INPUT                                 LINE 5 COL 25
                  PIC X(20)     FROM NAME-IN.
    05  VALUE "C - TO CONTINUE"                    LINE 11 COL 30.
    05  VALUE "Q - TO QUIT"                        LINE 12 COL 30.
    05  VALUE "ENTER RESPONSE"                     LINE 14 COL 30.
    05  RESPONSE-INPUT                             LINE 14 COL 45
                  PIC X         TO RESPONSE-IN-WS.
PROCEDURE DIVISION.
...
B-100-PROCESS.
    READ INPUT-FILE
        AT END
           MOVE "YES" TO EOF-IND.
    PERFORM B-200-LOOP
          UNTIL EOF-IND = "YES" OR RESPONSE-IN-WS = "Q".
B-200-LOOP.
    DISPLAY DATA-ENTRY-SCREEN.
    ACCEPT DATA-ENTRY-SCREEN.
    READ INPUT-FILE
        AT END
           MOVE "YES" TO EOF-IND.
```

In this example, the programmer has written the code to perform the B-200-LOOP until EOF-IND = "YES" OR RESPONSE-IN-WS = "Q". This means that if the user does not enter the Q, all records on the file will be processed. However, if the user wants to stop processing at any point, entering a Q to quit will terminate the processing.

In the example just covered, the data was used from the file section. Just as the programmer has the option to WRITE FROM, there is also the option to READ INTO. This means that the programmer can define the screen data in WORKING-STORAGE and read the information from the file directly into WORKING-STORAGE. Essentially, this means that a copy of the data is put into WORKING-STORAGE and a copy remains in the file section. Example 2B illustrates the use of the READ INTO.

Syntax:

```
READ INPUT-FILE INTO DATA-FOR-SCREEN
        AT END
              MOVE "YES" TO EOF-IND.
```

Example #2B:

```
DATA DIVISION.
FILE SECTION.
FD  INPUT-FILE.
01  INPUT-REC           PIC X(24).
WORKING-STORAGE SECTION.
01  RESPONSEZ.
    05  RESPONSE-IN-WS  PIC X        VALUE "C".
01  DATA-FOR-SCREEN.
    05  ID-IN-WS        PIC XXXX.
    05  NAME-IN-WS      PIC X(20).
SCREEN SECTION.
01  DATA-ENTRY-SCREEN.
    05  VALUE "DATA ENTRY SCREEN" BLANK SCREEN     LINE 1 COL 35.
    05  VALUE "ID #"                               LINE 3 COL 10.
    05   ID-INPUT                                  LINE 3 COL 25
                  PIC  X(4)     FROM ID-IN-WS.
    05  VALUE "NAME"                               LINE 5 COL 10.
    05  NAME-INPUT                                 LINE 5 COL 25
                  PIC X(20)     FROM NAME-IN-WS.
    05  VALUE "C - TO CONTINUE"                    LINE 11 COL 30.
    05  VALUE "Q - TO QUIT"                        LINE 12 COL 30.
    05  VALUE "ENTER RESPONSE"                     LINE 14 COL 30.
    05  RESPONSE-INPUT                             LINE 14 COL 45
                  PIC X         TO RESPONSE-IN-WS.
PROCEDURE DIVISION.
...
B-100-PROCESS.
    READ INPUT-FILE INTO DATA-FOR-SCREEN
        AT END
           MOVE "YES" TO EOF-IND.
    PERFORM B-200-LOOP
          UNTIL EOF-IND = "YES" OR RESPONSE-IN-WS = "Q".
B-200-LOOP.
    DISPLAY DATA-ENTRY-SCREEN.
    ACCEPT DATA-ENTRY-SCREEN.
    READ INPUT-FILE INTO DATA-FOR-SCREEN
        AT END
           MOVE "YES" TO EOF-IND.
```

Field or Group Screen Processing

In contrast to FULL SCREEN processing, data can be processed one field or group of fields at a time. This means that the program can accept a group of fields or each field the user keys in and check it before allowing another entry. Notice, the setup for screen by screen processing involves more design thought on the part of the programmer. In the sample, the input was broken down into three sections: one for id, one for name and one for the response. Within each of these sections there was a piece of data to be keyed. In the processing, the section is displayed and then the piece of data is accepted. At this point, the programmer could edit the data that was keyed before continuing.

Example #3:

```
       WORKING-STORAGE SECTION.
       01  RESPONSES.
           05  RESPONSE-IN-WS    PIC X        VALUE SPACES.
       01  DATA-FROM-SCREEN.
           05  ID-IN-WS          PIC XXXX     VALUE SPACES.
           05  NAME-IN-WS        PIC X(20)    VALUE SPACES.
       SCREEN SECTION.
       01  DATA-ENTRY-SCREEN.
           05  ID-SECTION.
               10  VALUE "DATA ENTRY SCREEN"      BLANK SCREEN
                                                  LINE 01 COL 30.
               10  VALUE "ID #: "                 LINE 05 COL 05.
               10  ID-ON-SCR-IN                   LINE 05 COL 15
                       PIC XXXX         TO ID-IN-WS.
```

```
        05  NAME-SECTION.
            10  VALUE "NAME:"              LINE 07 COL 05.
            10  NAME-ON-SCR-IN             LINE 07 COL 15
                    PIC X(20)        TO NAME-IN-WS.
        05  RESPONSE-SECTION.
            10  VALUE "C - TO CONTINUE"    LINE 16 COL 30.
            10  VALUE "Q - TO QUIT"        LINE 17 COL 30.
            10  VALUE "ENTER CHOICE:"      LINE 19 COL 30.
            10  RESPONSE-SCR               LINE 19 COL 45
                    PIC X     TO RESPONSE-IN-WS.
    PROCEDURE DIVISION.
    ...
    B-100-PROCESS.
        ...
        DISPLAY ID-SECTION.
        ACCEPT ID-ON-SCR-IN
        DISPLAY NAME-SECTION.
        ACCEPT NAME-ON-SCR-IN.
        DISPLAY RESPONSE-SECTION.
        ACCEPT RESPONSE-SCR.
        PERFORM B-200-LOOP
            UNTIL RESPONSE-IN-WS = "Q".
    B-200-LOOP.
        ... process...
        DISPLAY ID-SECTION.
        ACCEPT ID-ON-SCR-IN.
        DISPLAY NAME-SECTION.
        ACCEPT NAME-ON-SCR-IN.
        DISPLAY RESPONSE-SECTION.
        ACCEPT RESPONSE-SCR.
    C-100-TERMINATE.
        CLOSE OUTPUT-FILE.
```

Looking at the combination of display and accept. The program displays the ID-SECTION which puts out the part of the screen requesting the id and allowing the user to key in the id. The name of the field where the id is being keyed is called ID-ON-SCR-IN. The program accepts that field which takes the id that was keyed and stores according to the TO clause in ID-IN-WS. The id can now be checked for validity. The screen shown does not have an area to print out the result of the validity check and the procedure division does not contain code to check, display the results and allow the user to make the corrections. All of these components must be added to the program to allow for interactive edit checking. After the id has been taken in, the next display/accept combination asks for and takes in the name. The final combination asks for and takes in the user response to continue or quit.

Full screen processing deals with the entire screen of data. Field or group screen processing deals with the data one field at a time or several fields at a time depending on the layout of the data entry screen. The programmer should select the most appropriate processing for the application being developed.

Display and Accept as stand alone commands

The DISPLAY and ACCEPT can be used without a SCREEN SECTION as stand-alone commands. Frequently they are used this way in debugging programs or display messages during development. The stand alone ACCEPT is frequently used to take in the date from the system.

```
DISPLAY "NAME = " NAME-IN "AMT-WS = " AMT-WS.
ACCEPT JUNK.
```

In this example, the word NAME = followed by the contents of NAME-IN followed by the word AMT-WS = followed by the contents of AMT-WS would be displayed everytime the DISPLAY statement was encountered in the code. The ACCEPT JUNK is used to temporarily stop processing so the program or user has time to read what was displayed (note that JUNK or some other data name to be used in the ACCEPT is defined in WS).

```
ACCEPT DATE-WS FROM DATE.
```

This would bring in the DATE from the system and store it in DATE-WS.

In the two examples above, there is no screen section. The DISPLAY and ACCEPT are used as stand alone commands to show and receive information.