Z77-6048-2

TECHNICAL
INFORMATION
EXCHANGE

IBM®

June 13, 1966

DATA ENTRY WITH IBM 2260

Mr. Fred C. Gielow, Jr.
IBM Corporation
Dept. 078, Bldg. 850
Neighborhood Road
Kingston, New York 12401

The facts and features of the IBM 2260 Display Station are documented and distributed. Yet though we may be familiar with its principles of operation, we may be unfamiliar with ways to use the IBM 2260 for data entry. Data entry is a difficult and fundamental system-design problem. This paper catalogs and organizes 15 data-entry techniques to stimulate and broaden data-entry thinking, to assist in information-system design, and to help solve the data-entry problem.

The data-entry problem, like Gaul, is divided into 3 parts: human factors, equipment, and programming. The root of the programming portion of the data-entry problem is simply field identification: communicating the identity of each data field to the computer. This report gives attention to the basic problem of field identification, and to other data-entry factors such as ease, training, speed, completeness, checking, communication-line use, and programming. Each technique is examined for its good and bad points as a means of solving the data-entry problem.

Major revision 8/67

For IBM Internal Use Only

IBM Corporation, Technical Publications Dept., 112 E. Post Road, White Plains, N.Y. 10601

Z77-6048-2

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

NOTE

The applications described in this report are not necessarily best suited for the particular data-entry technique they illustrate.

The figures in this report were designed for illustrative purposes only. The appearance of actual displays on the IBM 2260 Display Station will differ from those shown in the illustrations, as will the Display Station itself.

The Nondestructive Cursor is shown for each technique in the figures. Although some techniques demand its use, others can do very nicely without it. However, since the Nondestructive Cursor allows the operator freedom and convenience, it is often desirable even when it's not essential.

INTRODUCTION

Man speaks in English, Chinese, Greek, Afghan... and he has enough difficulties and misunderstandings trying to converse with his fellow man, even using the same language. So it's not at all strange then, that when man tries to converse with something that "knows" only Binary -- this is, a computer--he often encounters problems.

So the man-machine hurdle has been a high one to jump. But as the machine becomes more and more invaluable in our businesses and in our lives, we push harder and harder to simplify the man-machine relationship, to ease the job of entering data into a computer. From early punched-card days we have striven to find ways to enter data more quickly, accurately, easily, efficiently into the computer.

DATA-ENTRY PROBLEM

What, really, is the data-entry problem? What makes the man-machine relationship a difficult one?

The data-entry problem is a 3-sided one: one side is the human factors problem, another is the equipment problem, and the third is the programming problem. All three are so integrated that specifying any one intimately affects the definition of the other two. Further, improving one must often be paid for in terms of complicating another or both others.

So, the data-entry problem is a system-design problem. It's a trade-off problem. It is the integration of all system elements to achieve the required degree of data-entry ease, accuracy, speed, efficiency; all balanced against the cost of equipment, programming, and system operation.

What makes the man-machine relationship a difficult one? First, the language barrier. Man's language is very imprecise. Whether I say, "It's a difficult problem" or "It is a difficult problem, " you know what I mean. But, if the computer is only programmed to recognized the former, it will make no sense of the latter.

Second, the man-machine relationship is difficult because it is a trade-off problem. We could design a system (theoretically) which would allow the operator complete freedom of data entry. However, to do so is to impose an impossible programming burden. On the other hand, we could specify that only the simplest of programs will be allowed. But, by doing this we create wholely unreasonable human-factors requirements.

How does the IBM 2260 solve the data-entry problem? Unfortunately, it doesn't solve it. What it does is reduce the problem to more manageable terms. By providing an easy-to-use, high-speed, low-cost, remote-or-local terminal, the IBM 2260 (with of course, the IBM System/360) simplifies the data-entry problem by solving the equipment portion of the problem.

So let me concentrate my attention on the human factors and programming portions of the data-entry problem. That is the purpose of this report.

FIELD IDENTIFICATION

If we unravel the cloak of confusion from around the programming portion of the data-entry problem, if we examine it completely naked, we find it only amounts to a matter of field identification. That is, how do we, as terminal operators, identify to the computer each field of data we wish to enter? Or, putting the shoe on the other foot, how does the computer tell us what fields of data it needs to complete a transaction?

Once field identification has been accomplished, it is a simple matter to enter the data itself. With the IBM 2260 it's as easy as typing: even easier The Destructive or Nondestructive Cursor eases data positioning and greatly simplifies corrections.

DATA-ENTRY CONSIDERATIONS

The remainder of this report is a discussion of techniques for data entry. Attention is given to the basic problem of field identification, and other data-entry factors are considered:

Ease. Is the technique easy to use? Is it comfortable?

Training. Is the technique easy to learn? Can it be learned quickly? Can it be mastered by nontechnical people?

Speed. Is the technique speedy? Does it allow efficient data entry?

Completeness. Does the technique assure that all fields required will be secured? Does it do this conveniently?

Checking. Does the technique permit easy checking of the data by the operator and by the computer?

Programming. Does the technique allow simplified programming? Does it result in programs of relatively small size, programs that do not impose heavy CPU-processing burdens?

Communication Lines. Does the technique permit efficient use of the data transmission lines? Is the ratio of the number of data-entry characters to the number of transmitted characters a high ratio? Does the technique involve a low message-frequency? This topic is discussed under the assumption that one of the read-manual-input commands is used (Read DS MI, Specific Poll, or General Poll) rather than a read-full-buffer command (Read Full DS Buffer or Read Addressed Full DS Buffer).

Miscellaneous Considerations. Is the technique affected by length of field, data content, other factors? Are there any unique features? Advantages? Disadvantages?

Each technique is examined, for its good and bad points, as a means of solving the data-entry problem.

I have written this report assuming the reader is familiar with the IBM 2260 and the IBM 2848. Familiarity with the way the equipment works is absolutely essential for understanding the techniques and for placing them in proper perspective. If additional familiarity is desired, I recommend you review the System Reference Library manual: IBM System/360 Component Description, IBM 2260 Display Station, IBM 2848 Display Control, Form A27-2700.

DATA ENTRY TECHNIQUES

FIXED-FORM TECHNIQUE

This technique is the one we are most familiar with. It is best described as a "fill in the blanks" technique. By "fixed" I mean rigid; if the fifth field begins on the third line, 27th character position, that's the only place it may begin.

The Fixed-Form Technique has been used extensively in the past. For example, all key-punching operations are based on it. But despite its wide use, the Fixed-Form Technique severely restricts the operator. Human factors are principally ignored in favor of programming factors; for this technique is the easiest to program. Each field of data is in one and only one location. There is no flexibility or variability. Everything is fixed -- rigid.

Because of this, the man-machine ratio tips far in favor of the machine; programming is simplified, but operator training is complicated. It is not an easy technique to use. With the IBM 2260, the operator must properly position the cursor to the exact location for each field of data. Precise positioning of this kind is difficult with any data entry device.

Training for this technique is not easy, but neither is it hard. Experience is vital, but in spite of good training and good experience the technique may result in a relatively high error rate. Speed is impaired due to the time wasted in field positioning. Checking is complicated by the double-checking responsibility: position and control.

Completeness can be a problem. For example: Suppose data entry for transaction 1 requires eight fields, data entry for transaction 2 requires six fields, and suppose the same format is used for both transactions. How then does the operator have assurance of completeness for transaction 2? He doesn't, unless he knows positively each field required for each transaction--a severe restriction.

We can relax, somewhat, the human factors for the Fixed-Form Technique, to make it easier on the operator, but we do so at programming expense. If we allow the operator the freedom of several character spaces in field positioning, for example, we must check and adjust the field with our program.

Communication-line use can be very efficient, but the degree of efficiency depends highly upon the method of field identification. The more field identifiers and the longer the field identifiers, the greater the use of communication lines and the poorer the ratio of the number of data-entry characters to the number of transmitted characters.

Field identification for the Fixed-Form Technique is done by naming each field and identifying where it is to be located. This may be done in either of two ways: the computer can transmit the field names or they may be implied by system procedure.

The first method is generally easier to use, but the second is of definite value in certain circumstances. When the computer sends a display of field identifiers, the operator fills in the blanks according to a predefined location convention. Figure 1, for example, shows a display of field identifiers partially filled in; each data field starts one space after its identifier and colon.

If the field names are implied, the convention is expanded to include an understood fixed-sequence of fields and their starting points. Figure 2 shows this technique used for entering FORTRAN statements into a computer to create a program "on-line." This figure includes implied field identifiers: statement number in character positions 1 through 5, continuation designation (if used) in position 6, and the FORTRAN statement in positions 7 through 72.

Note that in Figure 2 the Start Symbol is located in character position 80, line 12. Positioning it so permits the first character in line 1 (the letter C) to be data. Wraparound makes this possible.

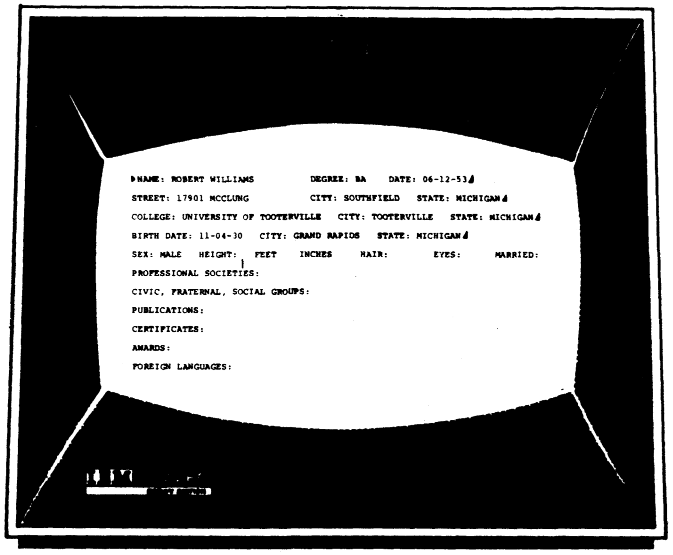
Note also that the Nondestructive Cursor Feature is required for the Fixed-Form Technique, if the field identifiers are supplied. The feature may also be very valuable even if the field identifiers are implied.

Someone suggested that an overlay mask of field identifiers might be placed on the screen of the IBM 2260 to facilitate the Fixed-Form Technique. The use of masks is strongly discouraged because the IBM 2260 does not have the ability to absolutely position the image on the screen. Further, since the character size may be adjusted, linearity is a function of relative character size and is not tied to any absolute value or dimension. Consequently, use of a mask is not a satisfactory means of field identification.

The Fixed-Form Technique is a fixed-sequence technique since the data is arranged on the screen in a never-changing order. Yet the "form" can be completed, that is, filled in, in any sequence. Nevertheless, the resulting message sent to the computer will always contain fields in the same order.

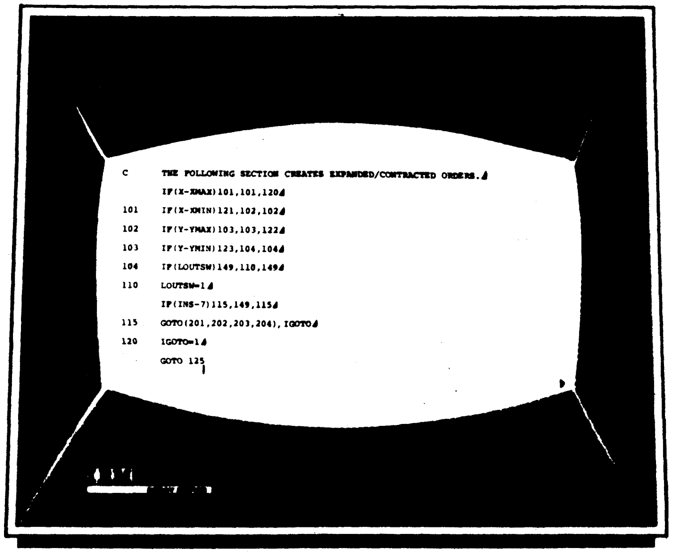
Note that after the data has been entered, the cursor (tle vertical bar in the figures) must be positioned following the last character of the last word of data to be entered. This is essential since when the Enter Key is depressed (assuming a read-manual-input command is used), only the data from the start-of-message character (▶) to the cursor will be transmitted to the computer. Furthermore, if the new line symbol (◢) is used, all data to the right of the first new line symbol, on each line, will not be transmitted. The point is that if the cursor is not properly positioned or if the new line symbol is carelessly used, data assumed to be transmitted to the computer may not be. Consequently, in Figure 2 for example, the cursor must not be positioned anywhere to the left or above where it is shown, without resulting in incomplete data when the message is sent to the computer.

Although there are many limitations to the use of the Fixed-Form Technique, there are many applications where it is the best means of data entry.



```
▶NAME: ROBERT WILLIAMS          DEGREE: BA    DATE: 06-12-53▲
STREET: 17901 MCCLUNG           CITY: SOUTHFIELD   STATE: MICHIGAN▲
COLLEGE: UNIVERSITY OF TOOTERVILLE   CITY: TOOTERVILLE   STATE: MICHIGAN▲
BIRTH DATE: 11-04-30   CITY: GRAND RAPIDS   STATE: MICHIGAN▲
SEX: MALE   HEIGHT:   FEET    INCHES     HAIR:        EYES:      MARRIED:
PROFESSIONAL SOCIETIES:
CIVIC, FRATERNAL, SOCIAL GROUPS:
PUBLICATIONS:
CERTIFICATES:
AWARDS:
FOREIGN LANGUAGES:
```

FIGURE 1. FIXED-FORM TECHNIQUE USED FOR PERSONNEL-RESUME DATA-ENTRY



```
C       THE FOLLOWING SECTION CREATES EXPANDED/CONTRACTED ORDERS.▲
        IF(X-XMAX)101,101,120▲
101     IF(X-XMIN)121,102,102▲
102     IF(Y-YMAX)103,103,122▲
103     IF(Y-YMIN)123,104,104▲
104     IF(LOUTSW)149,110,149▲
110     LOUTSW=1▲
        IF(INS-7)115,149,115▲
115     GOTO(201,202,203,204), IGOTO▲
120     IGOTO=1▲
        GOTO 125
```

FIGURE 2. FIXED-FORM TECHNIQUE USED FOR PROGRAM-STATEMENT DATA ENTRY

FORM CHECK-OR-CHANGE TECHNIQUE

This technique is similar to that above except the form is already filled in with data. The operator merely changes whatever fields are not correct. This technique has application where most of the data is predictable by the computer, or where certain fields contain the same data most of the time.

For example: Suppose the IBM 2260 is used to enter customer address information into a computer. Suppose, further, that 90% of the customers live in Detroit, Michigan and the remaining 10% live across the river in Windsor, Ontario. The Form Check-or-Change Technique is used, and the computer always supplies "Detroit" as the city for each transaction. The operator checks the city, and in only 10% of the cases is a change to "Windsor" necessary.

Figure 3 shows the technique used for entering weekly attendance data. In most cases employees work 8 first-shift hours a day, Monday through Friday, with Code 64. The display (Figure 3) is completely filled in with data for the usual case. The figure shows the cursor (Nondestructive) positioned to change the first-shift hour entry to reflect an employee's Saturday work.

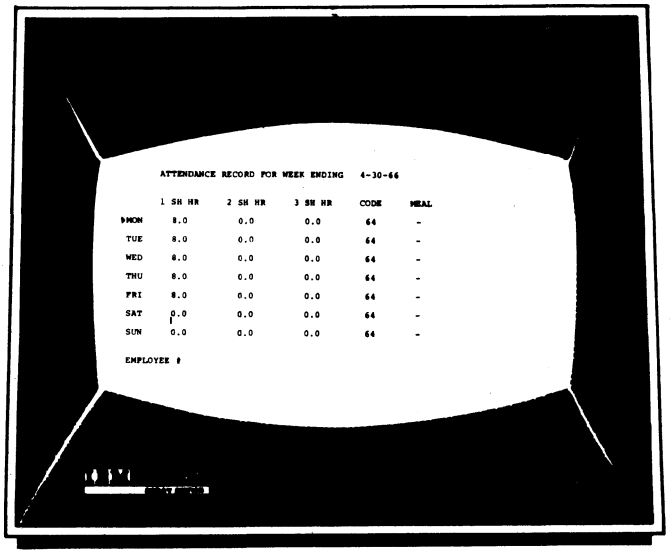
When the record is checked, and changed if necessary, the operator adds the employee number, and sends the entire message to the computer.

For this technique, the Nondestructive Cursor feature is essential. The ability to skip about the screen, changing only the fields needing change, leaving all others untouched, is a fundamental requirement. This ability imposes the restriction of careful cursor positioning before transmission, however.

The Form Check-or-Change Technique is an easy one to use. Since the data for entry is already displayed, field-positioning problems are greatly simplified. In some cases the change to a field of information results in fewer characters than the original field contained. This necessitates "blanking" the characters left over, but this does not severely affect ease of operation, particularly since changes are infrequent.

Training is not at all difficult for this technique. However, training must teach to guard against the tendency to skip over or ignore those fields, supplied by the computer, with unusually infrequent changes. Who can blame an operator who wants to bypass checking a field which needs no change 99 times out of 100?

Another possible danger that training must cope with is the inclination to "manipulate the situation" to result in no change to a field. Let me give an example. Suppose 82% of the customers for a telephone company order only one phone. Including "1" in the order-quantity field with the Form Check-or-Change Technique, may result in the Service Representative's selling only one telephone. That is, it may be easier to discourage a customer's desire for extensions that it is to change the order quantity.



```
ATTENDANCE RECORD FOR WEEK ENDING   4-30-66

      1 SH HR    2 SH HR    3 SH HR    CODE   MEAL
▸MON   8.0        0.0        0.0        64     -
 TUE   8.0        0.0        0.0        64     -
 WED   8.0        0.0        0.0        64     -
 THU   8.0        0.0        0.0        64     -
 FRI   8.0        0.0        0.0        64     -
 SAT   0.0        0.0        0.0        64     -
 SUN   0.0        0.0        0.0        64     -

 EMPLOYEE #
```

FIGURE 3. FORM CHECK-OR-CHANGE TECHNIQUE USED FOR ATTENDANCE-RECORD DATA ENTRY

This "danger" can be used to advantage by turning the tables: Include "2," "3," or more in the order quantity to encourage sales. You can see that the Form Check-or-Change Technique is filled with subtle implications and demanding system-design problems.

This technique is fast since the computer does most of the work of data entry. Completeness is satisfied by the technique, and checking is an intrinsic part of it. That is, checking is not done separately, as with other techniques; it is part of the basic data-entry procedure. Checking is enhanced by the fact that fields requiring no change are bound to be correct; they were supplied by the computer.

Programming for this technique is not inherently difficult, but it could be complicated if complex methods are required to determine each field's contents.

Communication-line use is not very efficient for the Form Check-or-Change Technique because all data is transmitted first from the computer to the IBM 2260, and then it is all transmitted back again. This inefficiency may increase the number of communication lines required, but it may be a small price to pay for the benefits of the technique if the application demands its features.

Like the Fixed-Form Technique, the Form Check-or-Change Technique uses a fixed-field sequence. Field identification is either computer-supplied or implied. If implied, field identification is the responsibility of the operator, a human-factors degradation.

Because of its unique characteristic of supplying the data for each field, this technique has limited application.

FREE-FORM-WITH-DELIMITERS TECHNIQUE

For this technique the field identification is not specified on the screen. The field sequence is fixed, and the field identification is implied. That is, the operator must know what fields of data must be entered, and he must know their sequence. Note that if all fields are of constant length, this technique becomes very much like the Fixed-Form Technique with field identification implied.

This free-form technique uses special characters, "delimiters," to indicate the end of one field and the beginning of the next. You may use any convenient special character as delimiter: for example, the slash (/), the asterisk (*), the colon (:), the semicolon (;), and so on. Characters other than special characters may be used if desired. You may use 2 X's, 2 Q's, 2 or more blanks, or any combination of characters which convey no other meaning to the program.

Two adjacent delimiters indicate that a field is to be omitted; that it is unnecessary for a particular transaction. That is, delimiters separate each field, whether or not the field is filled.

A principal advantage of the Free-Form-with-Delimiters Technique is the variability of field length it allows. This technique easily handles data fields that vary in length from zero to as many as a hundred or more characters.

Speed is also an advantage. This technique allows rapid data entry. Fields may be entered as rapidly as they can be typed; faster, in fact. IBM 2260 wrap-around line-to-line frees the operator from end-of-line worries.

Another advantage is the high density of data this technique yields. Since no identification is displayed, all displayed information, except delimiters, is data.

The Free-Form-with-Delimiters Technique is not particularly difficult to use. No careful positioning or spacing is required. The operator merely enters field after field of data, each separated by a delimiter. The technique becomes tricky, however, if a series of fields have no entries. In these cases, the operator must enter the precise number of delimiters to maintain field identification with the computer. If the operator makes one mistake in doing this, all remaining fields will be incorrectly identified.

Training is difficult because the operator must know not only each field, but also the sequence of fields. This is a crucial training problem and a fruitful source of errors.

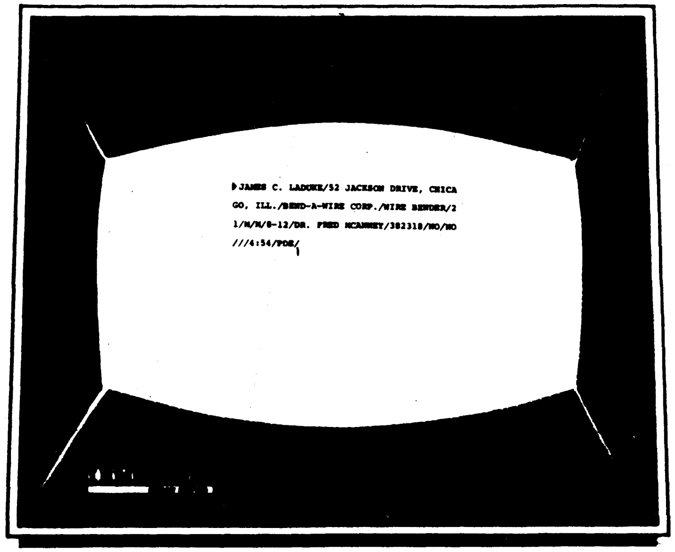
Speed, as I mentioned, is an asset of this technique. Completeness of data, on the other hand, is only as good as the operator's memory. If he remembers all fields, everything is fine. If he forgets one or more, he can cause all sorts of errors and problems for himself.

Checking for accuracy is awkward since the data fields are jammed one after another. And besides checking for accuracy, the operator must check for proper field sequence. To ease these checking problems with the Free-Form-with-Delimiters Technique, you may wish to first enter the data into the computer without checking it. Then the computer can format it and send it back, neat and organized, for checking.

Figure 4 shows hospital-patient admitting-data displayed on the IBM 2260 using the Free-Form-with-Delimiters Technique. Figure 5 shows the same data after it has been formatted by the computer and displayed to the operator for checking.

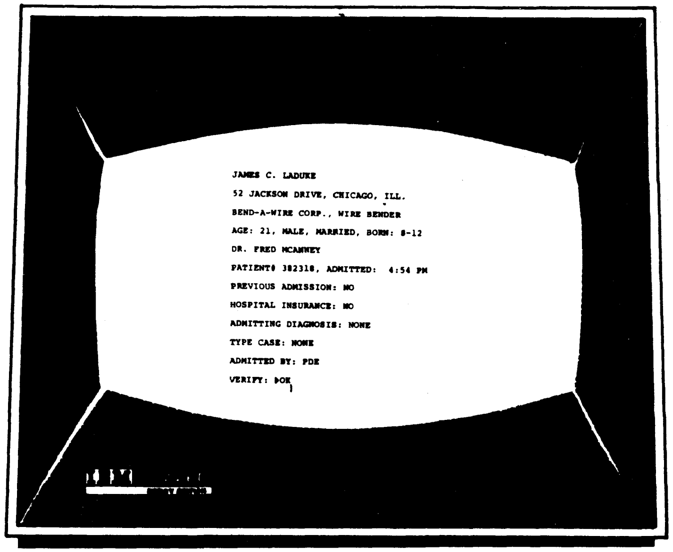
Programming for this technique is straightforward and is not much harder than programming for the Fixed-Form Technique or the Form Check-or-Change Technique. Inclusion of the checking procedure mentioned above complicates the programming somewhat, but the added complication is not severe.

The program must depend completely on delimiters to indicate field identity. This means that the program must examine each character of the entered data and it must format the entire message.



```
▶JAMES C. LADUKE/52 JACKSON DRIVE, CHICA
GO, ILL./BEND-A-WIRE CORP./WIRE BENDER/2
1/M/M/8-12/DR. FRED MCANNEY/382318/NO/NO
///4:54/PDE/
```

FIGURE 4. FREE-FORM-WITH-DELIMITERS TECHNIQUE USED FOR HOSPITAL-PATIENT ADMITTING DATA ENTRY



```
JAMES C. LADUKE
52 JACKSON DRIVE, CHICAGO, ILL.
BEND-A-WIRE CORP., WIRE BENDER
AGE: 21, MALE, MARRIED, BORN: 8-12
DR. FRED MCANNEY
PATIENT# 382318, ADMITTED:  4:54 PM
PREVIOUS ADMISSION: NO
HOSPITAL INSURANCE: NO
ADMITTING DIAGNOSIS: NONE
TYPE CASE: NONE
ADMITTED BY: PDE
VERIFY: ▶OK
```

FIGURE 5. COMPUTER-FORMATTED DISPLAY OF INFORMATION ENTERED WITH THE FREE-FORM-WITH-DELIMITERS TECHNIQUE

Communication-line use for the Free-Form-with-Delimiters Technique is as efficient as any data-entry technique. Only entered data (aside from delimiters) is transmitted. The ratio of the number of data-entry characters to the number of transmitted characters can be very high.

CONVERSATION-MODE TECHNIQUE

This is probably the easiest data-entry technique from a human-factors standpoint. Conversation Mode is a question-and-answer technique. That is, the computer asks a question, and the operator supplies the answer. The computer asks the next question, and so on. In this way each field of data is individually secured. At the conclusion of the conversation sequence, the computer signals END.

Conversation Mode is practically foolproof; it is very easy to use. There are no field positioning problems, no delimiters, no implied field identifications, no cursor positioning before transmission. The computer leads the operator "by the hand" through data entry. Each field is identified by the computer, and the sequence is fixed. The operator has no choice as to the next data fields entered; the computer takes care of this.

Training is simplified with Conversation Mode. The operator need only learn what is meant by each computer question.

Conversation Mode is not as fast a technique as the Free-Form-with-Delimiters technique, for example, but it is not slow. Each field is entered as a separate message, so computer intervention is required on a field-by-field basis, but this intervention is not expensive in terms of time.

Completeness is guaranteed. The computer does not signal END until all fields of data have been entered.

Checking is easy. With the IBM 2260, each field may be completely checked on the display screen before it is transmitted to the computer. The fields are easily identified and neatly displayed for easy checking.

Programming, on the other hand, is more complicated with Conversation Mode, because the computer must individually process every single field of data entered. While the other techniques I've described so far handle many data fields at a time, that is, many fields of data per message transmitted to the computer, Conversation Mode handles only one at a time. This means more messages, more computer interruptions, more computer time devoted to housekeeping, organization, and control. With hundreds of terminals each entering data fields, one at a time, this management problem can be significant. Fortunately, operating-system techniques help a great deal with this management problem.

The receipt of data field-by-field can be used to great advantage. Reasonableness checks and other processing can be performed for each field before sending the next "question." These complicate the programming still more, but allow the immediate flagging of unacceptable fields. Then the next "question" would be a request to correct the previously entered field. Catching errors this way permits their correction at the best possible time: as they occur.

The ratio of the number of data-entry characters to the number of transmitted characters for this technique can be high, but since each data field is entered individually, the message frequency is high. High message-frequency reduces the efficiency of communication-line use.

Figure 6 shows the Conversation-Mode Technique used by the telephone-company Service Representative to enter data for a new customer. This is an "In" Service Order. The computer has asked for name, address, city, service and equipment (S&E), credit reference, and deposit, and the Service Representative has entered all fields except the last, which she is about to enter.

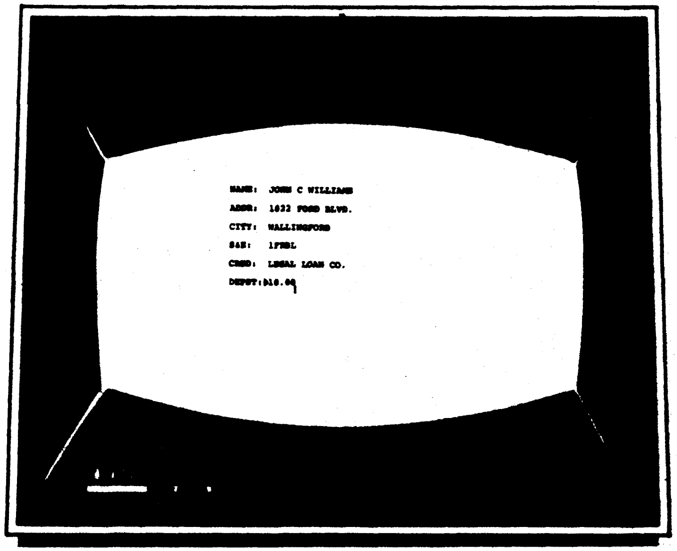
ADJACENT-FORM TECHNIQUE

This is another fixed-sequence technique where the computer supplies the field identification. For this technique the required fields are identified at the right side of the display screen, as shown in Figure 7. The operator merely fills in the "form" by supplying each data field next to its identifier. You will recognize the similarity between this technique and the first one I described, the Fixed-Form Technique.

Like Conversation Mode, this is a very easy data-entry technique, but unlike Conversation Mode it allows the operator freedom to choose the sequence in which the fields are entered. For example, in Figure 7, the name might be entered first, then the order number, then the phone number, and so on. The ability of free movement of the Nondestructive Cursor allows this action. However, cursor manipulation, though simple, imposes an additional responsibility on the operator, making this technique slightly more difficult than the Conversation-Mode Technique.

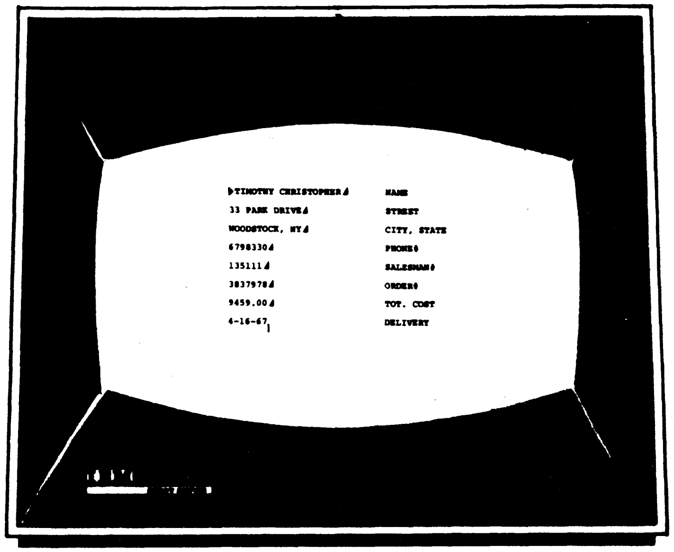
The Adjacent-Form Technique, like Conversation Mode is an easy one to learn. Speed is rapid, encumbered only by cursor manipulations. Checking is assured. Programming is not difficult. Communication lines are more heavily used for this technique than for Conversation Mode because a long message from the computer is necessary before data entry, to establish the field identifiers. On the other hand, message frequency is lower for the Adjacent-Form Technique.

There are two ways of handling messages with the Adjacent-Form Technique. The individual fields may be sent to the computer one at a time, like the Conversation-Mode Technique, or they may be sent collectively as one message. The first way complicates the programming, but it allows instant flagging of unacceptable fields. However, field identification becomes messy. That is, how does the computer know which field is being entered; how can it identify an entered field? Identification must be handled by numbering fields for entry, by transmitting the computer-supplied field identifier with the data, or by some other method.



```
NAME:  JOHN C WILLIAMS
ADDR:  1022 FORD BLVD.
CITY:  WALLINGFORD
S&E:   1FMBL
CRED:  LEGAL LOAN CO.
DEPST:$10.00
```

FIGURE 6. CONVERSATION-MODE TECHNIQUE USED FOR TELEPHONE COMPANY SERVICE-ORDER DATA ENTRY



```
▶TIMOTHY CHRISTOPHER◢      NAME
33 PARK DRIVE◢             STREET
WOODSTOCK, NY◢             CITY, STATE
67983304◢                  PHONE#
135111◢                    SALESMAN#
3837978◢                   ORDER#
9459.00◢                   TOT. COST
4-16-67                    DELIVERY
```

FIGURE 7. ADJACENT-FORM TECHNIQUE USED FOR SALES-ORDER DATA ENTRY

Sending all fields collectively as one message can be done in two ways: (1) insert a New Line Symbol at the end of each data field entered (as in Figure 7), thereby sending only data, no field identifiers, or (2) do not use the New Line Symbol at all, thereby sending data and field identifiers. The first way is recommended because the second necessitates difficult cursor manipulations for each new line. But for either way you must be careful to position the cursor at the end of the last data field before transmitting the message to the computer.

This is the price you must pay for flexibility in the sequence of data fields entered: cursor positioning. If you don't mind a fixed sequence for entering the fields, you don't have to worry about positioning the cursor. However, if you do demand this flexibility, you must assure that the cursor is properly positioned before sending each message. You may want to use read-full-buffer commands rather than read-manual-input commands to circumvent cursor-positioning restrictions, but this is not the answer. The read-full-buffer commands were intended for diagnostic purposes, not for normal data-entry operations. Read-full-buffer commands result in End of Message (EOM) conditions and cursor-reset conditions which make use of these commands unattractive.

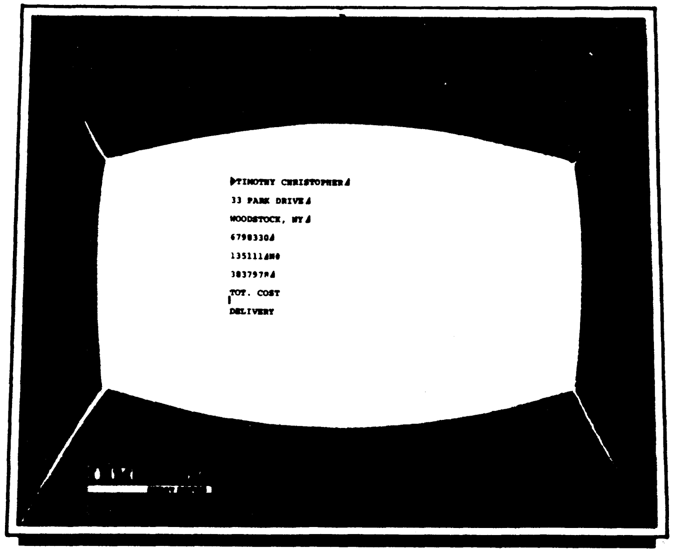
One limitation of the Adjacent-Form Technique is that the field length has a maximum size defined by the line length on the IBM 2260. For use with the IBM 2848 Models I and II, the line length is 40 characters; for Model III it is 80 characters. The length of the field identifier subtracts from the total line length available if the identifiers are not overwritten. While it is possible to overwrite the identifiers, this is not recommended because it is confusing for the operator. Data fields of length greater than one display line would have to be handled on a special basis.

OVERWRITE-FORM TECHNIQUE

This technique is exactly like the Adjacent-Form Technique above, except the field identifiers are listed on the left and the operator enters each field, overwriting its identification. Although destruction of the field identifier makes checking more difficult, this technique may have value in certain applications.

Figure 8 shows the Overwrite-Form Technique with the same data and identifiers used in Figure 7. Note that the data in the fifth line is shorter than the field identifier, leaving the remains of the identifier on the screen. This has no effect on the transmitted message, but it does make for confusion in checking. The cursor in Figure 8 is located in position to enter the total cost data.

The discussion for the Adjacent-Form Technique applies almost totally for the Overwrite Form Technique.



```
▶TIMOTHY CHRISTOPHER◢
33 PARK DRIVE◢
WOODSTOCK, NY◢
67983304◢
135111◢N#
3837978◢
TOT. COST
DELIVERY
```

FIGURE 8. OVERWRITE-FORM TECHNIQUE USED FOR SALES-ORDER DATA ENTRY

CODE-IDENTIFICATION TECHNIQUE

For this technique the operator enters a code for each field of data to be entered. For example (see Figure 9), the operator is entering insurance data into the computer. The operator first enters "CONTR:" followed by the contract number, then "NAME:" followed by the name, and so forth.

The data fields may be entered individually, as with the Conversation-Mode Technique, or collectively. Figure 9 shows the latter method.

This is a difficult technique because the operator must enter each field identifier precisely as it is programmed; no spelling or spacing errors are allowed. If the computer can not recognize an entry, it must reject it. Of course, you could program the computer to recognize several common misspellings of each identifier, but to catch all possible combinations is impossible. The identification problem becomes sizeable and a source of much operator error if there are many different identifiers.

Delimiters of one kind or another must be used to indicate the end of one data field to separate it from the identifier of the next. In Figure 9 both the New Line Symbol and a series of two or more blanks are used as delimiters.

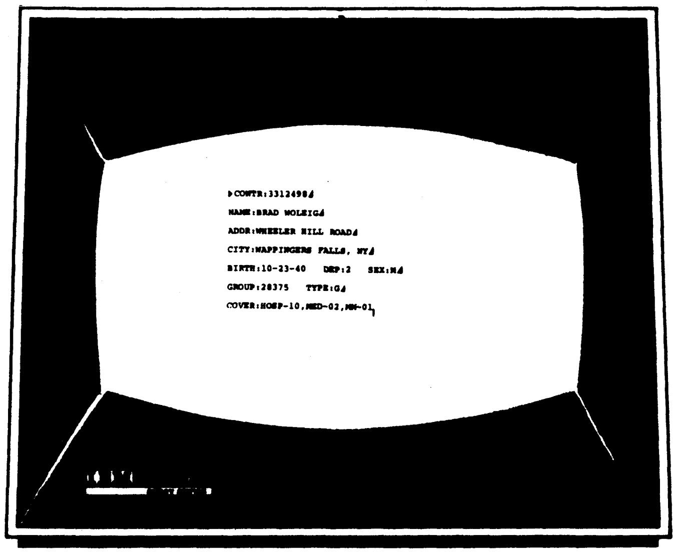
Since the operator identifies each field he enters, he may choose any sequence he desires. The Free-Form-with-Delimiters and Conversation-Mode Techniques do not allow any deviation from a fixed sequence.

Learning all the identifiers and their precise format, however, makes training a difficult task. Speed is greatly reduced, because besides entering data fields, the operator must also enter the data-field identifiers. Completeness is left to the operator; if he forgets a field, it is lost, unless the program includes completeness checks.

Checking is also difficult, because the operator must check both the data and the identifiers. Since a field identifier will be rejected by the computer if it is not exactly right, the operator may tend to devote more attention to the correctness of the identifiers than to the correctness of the data.

Programming must include decoding routines to decode each field identifier. Completeness checks with appropriate messages and procedures should also be included in the program.

The ratio of the number of data-entry characters to the number of transmitted characters for this technique and for Conversation Mode is approximately the same. But since the message frequency is lower, Code Identification makes better use of the communication lines.



```
▶CONTR:3312498◢
NAME:BRAD WOLEIG◢
ADDR:WHEELER HILL ROAD◢
CITY:WAPPINGERS FALLS, NY◢
BIRTH:10-23-40   DEP:2   SEX:M◢
GROUP:28375   TYPE:G◢
COVER:HOSP-10,MED-02,MM-01
```

FIGURE 9. CODE-IDENTIFICATION TECHNIQUE USED FOR INSURANCE DATA-ENTRY

Figure 9 shows Code Identification with the field identifiers preceding the data. The identifiers may follow the data if desired, but this only disrupts the neatness of the display and makes checking harder.

CODE-LINK TECHNIQUE

This technique starts with the computer supplying a collection of field identifiers, each of which is numbered. The operator chooses the data field he wishes to enter, he observes the identifier number, then he enters the number followed by the data. Note that this technique is similar to the Code-Identification Technique. But with Code Link, the operator has less information to enter, a single number. Consequently, it can be slightly faster and easier than Code Identification. There is a chance of number mix-up, however, since the operator is translating from identifier word to number, and this leads to error.

Code Link should be easier to use; it is easier to learn, and it can be significantly faster than Code Identification. Completeness is better handled also. Data checking is not hard, but assuring that the right data field is linked with the right field identifier requires reverse translation; from code to identifier name.

Programming can be easier for this technique than for Code Identification because translation is easier; numbers are easier to decode than words or mnemonics.

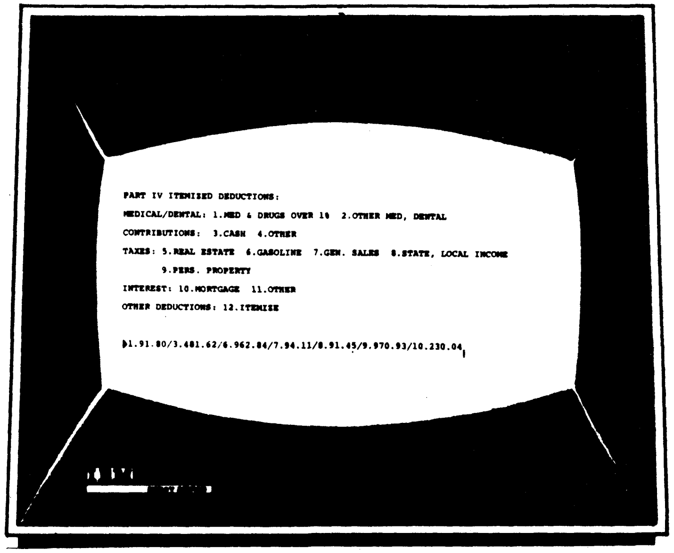
Efficiency in the use of communication lines for this technique depends on how the technique is handled; the efficiency can be quite high or it can be not so high.

Figure 10 shows how this technique can be used to enter income-tax data into a computer system. Figure 11 shows a more complicated use of this technique: to enter update information for an employee-skills inventory. The first several lines of the display identify the transaction type and list nine data field identifiers. The remainder of the display is information entered by the operator. The first entered line means the operator is adding a skill to the inventory. The skill is soldering. The length of experience is 4 months; it is in the company, at the Kingston location. The other entries may be similarly translated.

DATA-SELECTION TECHNIQUE

The Data-Selection Technique of data entry can only be used to select one or more alternatives from a given list. Figure 12 shows an example of this technique. The data to be entered is one of the seven west coast states shown. Rather than enter "NEVADA" with the alpnameric keyboard, the operator merely enters "4". The computer then responds with a display of data from which the next field is selected and entered.

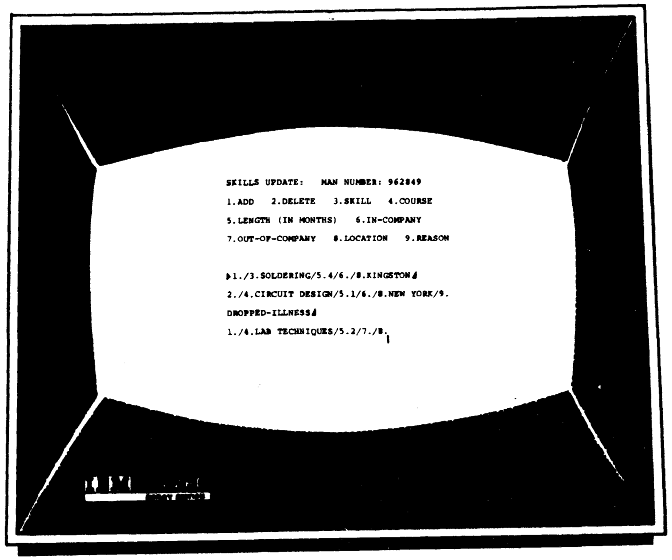
This technique can be particularly useful for data entry of alphabetic information, on a limited basis, with the IBM 2260 Numeric-Only Keyboard. That is, for certain



```
PART IV ITEMIZED DEDUCTIONS:
MEDICAL/DENTAL: 1.MED & DRUGS OVER 1%  2.OTHER MED, DENTAL
CONTRIBUTIONS:  3.CASH  4.OTHER
TAXES: 5.REAL ESTATE  6.GASOLINE  7.GEN. SALES  8.STATE, LOCAL INCOME
       9.PERS. PROPERTY
INTEREST: 10.MORTGAGE  11.OTHER
OTHER DEDUCTIONS: 12.ITEMIZE

▶1.91.80/3.481.62/6.962.84/7.94.11/8.91.45/9.970.93/10.230.04
```

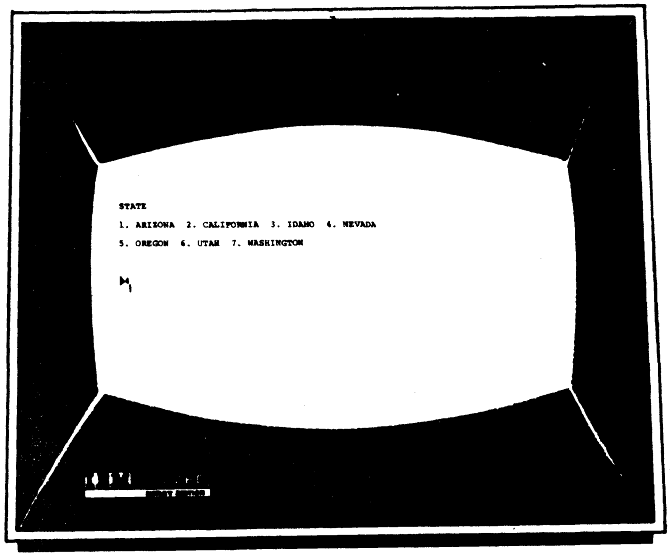
FIGURE 10. CODE-LINK TECHNIQUE USED FOR INCOME-TAX DATA ENTRY



```
SKILLS UPDATE:   MAN NUMBER: 962849
1.ADD   2.DELETE   3.SKILL   4.COURSE
5.LENGTH (IN MONTHS)   6.IN-COMPANY
7.OUT-OF-COMPANY   8.LOCATION   9.REASON

▶1./3.SOLDERING/5.4/6./8.KINGSTON◢
2./4.CIRCUIT DESIGN/5.1/6./8.NEW YORK/9.
DROPPED-ILLNESS◢
1./4.LAB TECHNIQUES/5.2/7./8.
```

FIGURE 11. CODE-LINK TECHNIQUE USED FOR EMPLOYEE-SKILLS INVENTORY DATA-ENTRY



```
STATE
1. ARIZONA  2. CALIFORNIA  3. IDAHO  4. NEVADA
5. OREGON  6. UTAH  7. WASHINGTON

▶4
```

FIGURE 12. DATA-SELECTION TECHNIQUE USED TO SELECT ONE OF SEVEN AMERICAN STATES

applications the numeric-only keyboard (no alphabetics) can be used to identify, hence enter, alphabetic data. This technique is similar to the Code-Link Technique, except selection of the identifying number is all that is required.

The technique is very easy to use; training is minimal, and it's speedy. Checking is simple, but programming is difficult: this technique is harder to program than Conversation Mode. Completeness is assured since each field of data is handled individually. Since the ratio of the number of data-entry characters to the number of transmitted characters is low, you may think the communication lines are inefficiently used. This is true if you define your terms that way, but with Data Selection you can convey a lot of information to the computer with the little bit of data you actually enter.

POST-IDENTIFICATION TECHNIQUE

This technique is another which allows a variable sequence for data-field entry and requires the operator to identify the fields. The technique is similar to Code Identification; the difference being that the field identifiers are not entered until the very last.

Figure 13 shows this technique used to enter police information about a stolen car. Each data field has been entered and all are identified on the last line of the display.

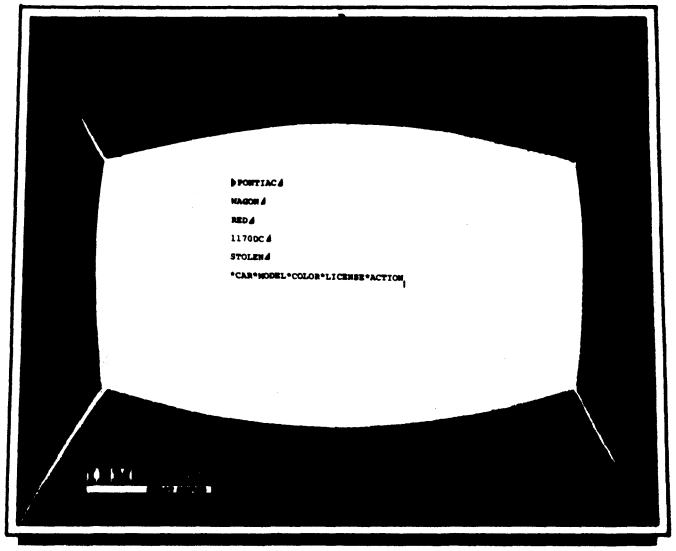
This technique requires delimiters between data fields (New Line Symbols in Figure 13), and also between field identifiers (asterisks in the example). If you use fixed-length fields, however, you don't need the delimiters. The fixed length takes the place of the delimiters to indicate field separation. This applies to both the identifiers and the data fields.

This technique is subject to most of the comments made about the Code-Identification Technique.

SEQUENCE-KEY TECHNIQUE

This technique is useful where there are a few basic forms or sequences for the data fields. Once the form or sequence is identified, the data is entered via the Free-Form-with-Delimiters Technique. Sequence Key has the advantage over Free-Form-with-Delimiters in that it allows format selection and includes a display of the field identifiers in sequence. The advantage of Sequence Key over Code Link is that it allows all fields to be entered with only one identification.

The sequence identifier may be positioned before or after the data field. I recommend that you position the identifier before the data, however, for the sake of checking and programming.



```
▶PONTIAC◢
WAGON◢
RED◢
1170DC◢
STOLEN◢
*CAR*MODEL*COLOR*LICENSE*ACTION
```

FIGURE 13. POST-IDENTIFICATION TECHNIQUE USED FOR STOLEN-VEHICLE DATA ENTRY

Figure 14 shows the use of this technique where each format contains the same field identifiers, but in different sequence. This example shows entry of airline reservation data.

Figure 15 shows the use of the technique where each format contains a different type of identification for the second field. This example shows entry of inventory data. Both examples use the asterisk as delimiter.

CONTENT-DECODE TECHNIQUE

This unique data-entry technique involves no field identification at all during the data-entry phase. Field identification is done by the computer based on the data content alone. Because of this, its use is limited. Only those applications which involve data fields, all of which are uniquely identifiable by content, are prospects for this technique.

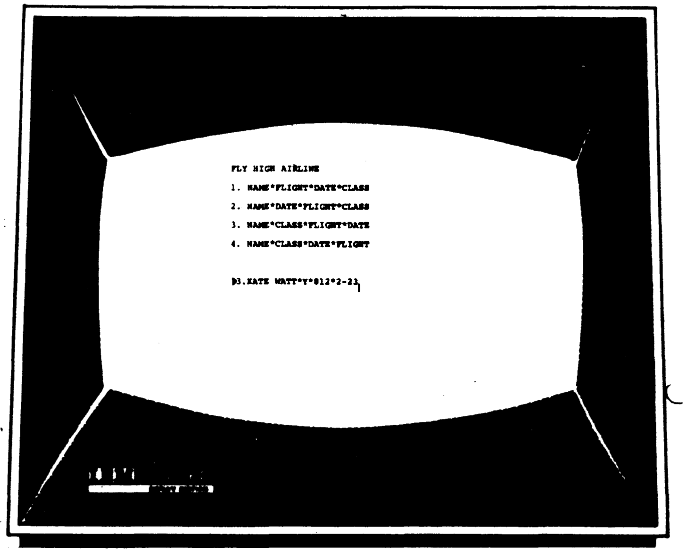
The technique is very easy to use: no complicated field identifications, no fixed-field sequence, only delimiters to set fields apart. The data fields may be entered individually or collectively.

The technique is easy to learn and is a fast means of data entry. Completeness is not inherently a part of the technique and would have to be built into the program. Checking is easy, since all attention can be devoted to data content. Programming is difficult, since each field must be uniquely identified, but this amounts to little more than a slightly different form of the reasonableness checks which would probably be included in a program anyway. This technique permits very efficient use of the communication lines. The efficiency is the same as for the Free-Form-with-Delimiters Technique. The length of field and field-length changeability have no effect on this technique, other than influencing uniqueness.

Figure 16 illustrates use of the Content-Decode Technique for entering employee data. The employee name is distinguishable from the address by the address number and the word "Road." Note, however, that this is a dangerous distinction. How would the computer know the difference between Paul Road and his unnumbered address of Pine Road? The age, birth date, and social security number are more easily differentiated.

CYCLE-STOP TECHNIQUE

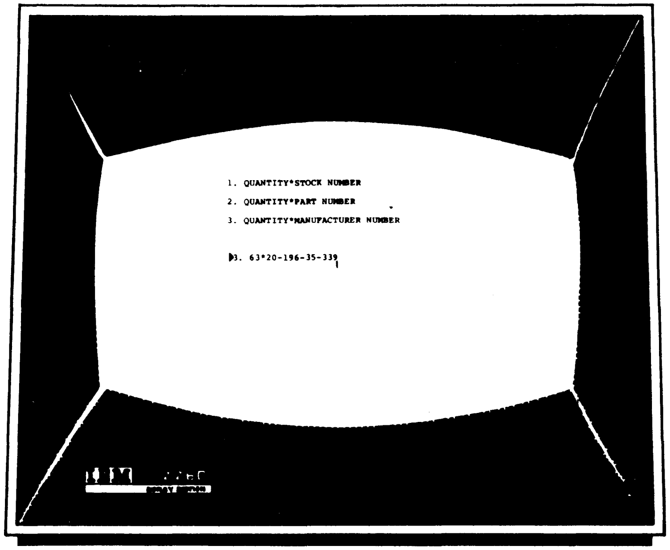
This technique, an unusual one which will only be applicable in rare cases, works in the following way: The computer sends one field identifier followed by a Start Symbol. A fixed time-interval later, perhaps one second, the computer sends the next identifier (again followed by a Start Symbol) and destroys the first identifier. This continues on a regular basis until the operator finds the identifier of the field he wishes to enter. He then quickly depresses the Shift and Enter keys, before the next identifier appears. This "message" signals the computer to discontinue the sequence and await entry of the data field. When the data is entered, the computer continues flashing identifiers.



```
FLY HIGH AIRLINE
1. NAME*FLIGHT*DATE*CLASS
2. NAME*DATE*FLIGHT*CLASS
3. NAME*CLASS*FLIGHT*DATE
4. NAME*CLASS*DATE*FLIGHT

▶3.KATE WATT*Y*812*2-23
```

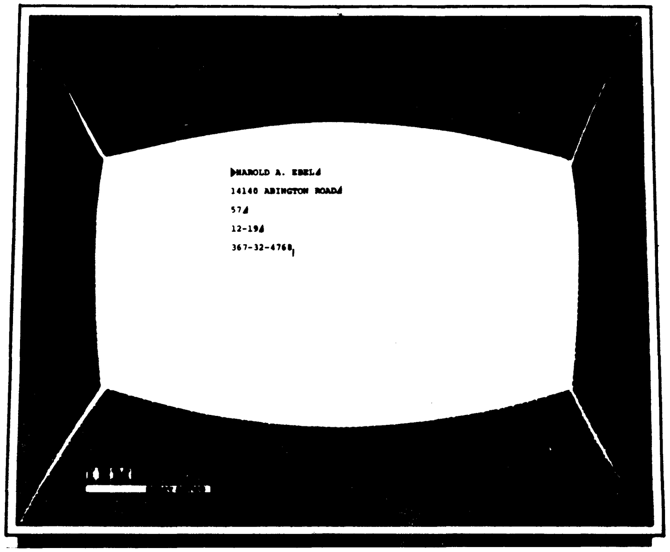
FIGURE 14. SEQUENCE-KEY TECHNIQUE USED FOR AIRLINE-FLIGHT RESERVATIONS DATA-ENTRY



```
1. QUANTITY*STOCK NUMBER
2. QUANTITY*PART NUMBER
3. QUANTITY*MANUFACTURER NUMBER

▶3. 63*20-196-35-339
```

FIGURE 15. SEQUENCE-KEY TECHNIQUE USED FOR INVENTORY DATA ENTRY



```
▶HAROLD A. EBEL◢
14140 ABINGTON ROAD◢
57◢
12-19◢
367-32-4768
```

FIGURE 16. CONTENT-DECODE TECHNIQUE USED FOR NEW-EMPLOYEE DATA ENTRY

Although the technique is riddled with problems -- excessive computer use, excessive communication-line use, awkward human factors, etc., -- there may be an application which could put it to good use.

Needless to say, the technique is not easy to use. Although the training is minimal, the speed is poor. Completeness is assured since the cycling continues over and over again until all data is secure; checking is not difficult, but programming complexity and overall operating factors make the value of this technique questionable.

Figure 17 shows how one computer-displayed field identifier appears to the operator for the Cycle-Stop Technique. A second later this identifier's successor (perhaps "ADDR") will replace "NAME" and the cycle will continue indefinitely, if the operator does not disrupt the sequence.

CYCLE-GO TECHNIQUE

This technique is like the Cycle-Stop Technique with one exception. The cycling is not under computer control: it's under operator control. That is, the operator depresses the Enter Key after each display is presented, thereby triggering the following display, until the desired field identifier is reached. Then the data is keyed in, and sent to the computer.

Though this technique may have limited application, it does bypass the major problem associated with the Cycle-Stop Technique. That is, Cycle-Stop cycles continuously until stopped, while Cycle-Go does not cycle until triggered by the operator. However, if used recklessly, this technique can be as wasteful as Cycle-Stop.

A possible use of this technique is for demonstrations at business shows. Interested people could, at their own rate, cycle through a series of displays of possible data-field identifiers. Each display would include instructions of what data to enter and how to enter it. Displays could describe how to enter data to identify a record required for retrieval, data to update a list, and so forth. So, with Cycle-Go, the demonstration can explain the application as it demonstrates it.

DATA-BRACKET TECHNIQUE

The Data-Bracket Technique is similar to the Data-Selection Technique in that the computer supplies the data, and the operator determines what portion of the displayed data he wishes to send back to the computer.

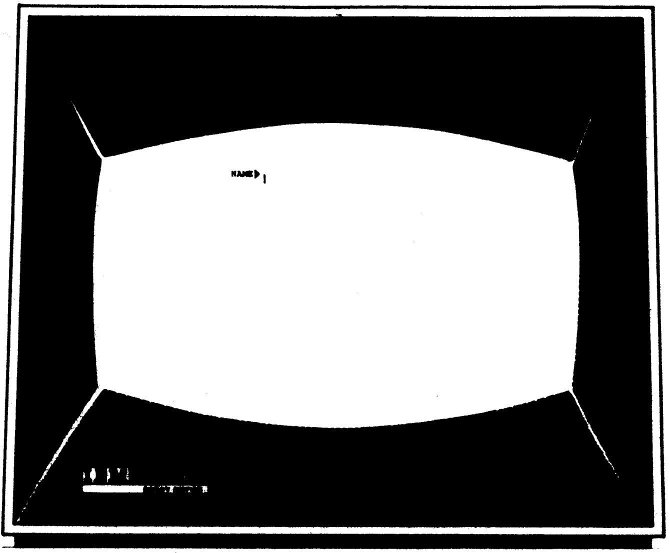


FIGURE 17. CYCLE-STOP TECHNIQUE DISPLAY OF FIELD IDENTIFIER

Figure 18 shows an example where the Data-Bracket Technique is put to good use. Here the IBM 2260 is used to edit a reference manual. The operator wishes to delete a portion of the text. He has surrounded that portion with the Start Symbol and the cursor. When the computer receives this data it can respond with an action interrogation: Should the data be deleted? Added? Removed and saved?

When the operator signals data deletion, the computer removes the designated data, reformats the text to close the gap, and redisplays the corrected text for the operator to check and continue with his editing.

The Data-Bracket Technique is easy to learn and easy to use. It is very speedy: certainly faster than keying in the data to delete, for example. Completeness is a question of how the technique is used: In most cases the technique is used to pick out a section of information displayed, as in the editing example, so completeness isn't a problem. Checking is easy since the only checking involved is the proper placement of Start Symbol and cursor.

Programming data entry for the Data-Bracket Technique is quite easy, but programming follow-on operations may not be so easy. For example, for text editing, after the entered data is identified as deletion data, the program must locate the corresponding data in the text, delete it, and reformat it. These kinds of operations can involve complicated character manipulations.

For this technique, the ratio of the number of data-entry characters to the number of transmitted characters may be quite low, but like the Data Selection Technique, you convey a lot of information to the computer with very little actual data entered via the keyboard.

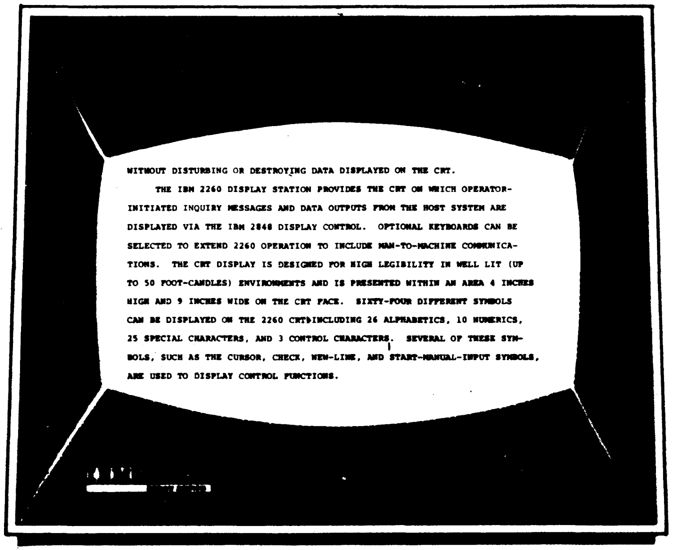
Only certain applications use the Data-Bracket Technique. The major restriction is that the computer must display whatever data is to be entered; that is, no new data can be entered.

The technique demands that the Nondestructive Cursor Feature be specified for the IBM 2848 Display Control.

COMBINATIONS AND MODIFICATIONS

While there are good and bad sides to each of the 15 techniques I've discussed, probably no one technique will solve all the problems of a data-entry application. Consequently, in all likelihood, your application will require a combination or modification of techniques to meet your specific data-entry needs.

There is certainly nothing sacred about the details of the 15 techniques as I've described them. They can be modified, revised, altered. For instance, suppose you wish to enter several columns of data, each of which is directly below a computer-supplied identifier on the display screen. This isn't exactly the Fixed-Form Technique, or any of the 15; it's a modification of Fixed-Form. What I'm saying is that variations on a technique's theme are indeed allowed.



```
WITHOUT DISTURBING OR DESTROYING DATA DISPLAYED ON THE CRT.
     THE IBM 2260 DISPLAY STATION PROVIDES THE CRT ON WHICH OPERATOR-
INITIATED INQUIRY MESSAGES AND DATA OUTPUTS FROM THE HOST SYSTEM ARE
DISPLAYED VIA THE IBM 2848 DISPLAY CONTROL.  OPTIONAL KEYBOARDS CAN BE
SELECTED TO EXTEND 2260 OPERATION TO INCLUDE MAN-TO-MACHINE COMMUNICA-
TIONS.  THE CRT DISPLAY IS DESIGNED FOR HIGH LEGIBILITY IN WELL LIT (UP
TO 50 FOOT-CANDLES) ENVIRONMENTS AND IS PRESENTED WITHIN AN AREA 4 INCHES
HIGH AND 9 INCHES WIDE ON THE CRT FACE.  SIXTY-FOUR DIFFERENT SYMBOLS
CAN BE DISPLAYED ON THE 2260 CRT▸INCLUDING 26 ALPHABETICS, 10 NUMERICS,
25 SPECIAL CHARACTERS, AND 3 CONTROL CHARACTERS.  SEVERAL OF THESE SYN-
BOLS, SUCH AS THE CURSOR, CHECK, NEW-LINE, AND START-MANUAL-INPUT SYMBOLS,
ARE USED TO DISPLAY CONTROL FUNCTIONS.
```

FIGURE 18. DATA-BRACKET TECHNIQUE USED FOR TEXT-EDITING DATA ENTRY

How can you tell what technique you need for your application? Will one of the 15 do the trick, or do you need a combination of techniques or a modification of one or more? Let's take an example. Let's look at the data-entry needs of the telephone-company Service Representative and see how they might be satisfied with the IBM 2260.

Which data-entry techniques are best suited for the Service Representative? Before we answer that we must examine what data-entry operations the Service Representative performs.

The first phase of nearly all customer contacts involves information retrieval; retrieval of the particular information from a particular customer record to handle the customer's particular problem. So, before information retrieval comes data-entry. The Service Representative must identify the inquiry type (balance due, toll, pending order, party interference, etc.) probably with a nmemonic, and she must identify the specific customer record, probably with the telephone number.

The data-entry technique used here should be fast and easy; it should not require a preliminary step where the computer supplies field identifiers. In the interest of speed the Service Representative should not have to identify the data fields. Consequently, only three techniques are readily applicable: Fixed-Form (with implied field identifiers), Free-Form-with-Delimiters, and Content-Decode. The one with greatest flexibility is Content Decode. As long as a basic delimiter rule is observed, the Content-Decode Technique will accept the inquiry type and telephone number in any sequence or spacing arrangement.

If the Content-Decode Technique offers too much programming complication, one of the other two techniques may be used. This costs convenience for the Service Representative, but it may be a valid system design choice, in view of other system considerations. If none of these three techniques precisely fits the requirements, some combination or modification may turn the trick.

What about the second phase of the customer contact, the action phase? During this phase the Service Representative must take action, usually in the form of data entry. She must enter a few or dozens of fields of information into the computer. What technique would you use here? What are the requirements?

Ease and speed are again important, but the three above mentioned techniques will not do. Fixed Form is not flexible enough, and it imposes too many cursor-positioning manipulations. Free-Form-with-Delimiters is too error-prone, too dependent on the operator's remembering the fields and their sequence. And Content Decode probably just won't work at all because of lack of data-field uniqueness. There are perhaps 5 good candidates: Conversation-Mode, Adjacent-Form, Code-Identification, Code-Link, and Post-Identification. And of course a modified technique or a combination of tecnniques is always a source of potential candidates.

Once we've chosen the techniques for Service-Representative inquiries and actions, we can examine the other data-entry operations she performs and choose techniques for these. And after we've designed the system for the Service Representative, what about the other Commercial Department people who must enter data into the computer? What about the people in the Plant Department, Revenue Accounting, Traffic, ... ?

One technique may not be sufficient, as we saw, to handle an entire application. Nevertheless, if at all possible, only one technique should be used for any one class or type of data-entry messages.In the telephone company, for example, the Service Representative may need the ability to enter 16 different kinds of messages to handle the different kinds of actions a customer contact demands. Good system design would provide only one data-entry technique to handle all 16 types of messages. This is important because it is confusing to enter similar data with different techniques. We expect a car to require a different driving technique than an airplane, but we expect all cars to require pretty much the same technique. So, while there may well be several data-entry techniques used in an application, similar types of data-entry activities should be handled with the same technique.

SUMMARY

Every phase of data entry must be carefully explored, analyzed, before the right data-entry technique can be chosen. What about ease, training, speed, completeness, checking, programming, and all those other complicating factors? What about overall system design considerations?What kinds of trade-offs are involved with data entry, with processing, with data output, with equipment capabilities, with procedural requirements ? The total system must be designed and engineered, and compromises with its ingredients are inevitable.

As we have seen, there are a multitude of data-entry techniques: the 15 I described and dozens of combinations and modifications. Like every system-design problem, data-entry can be fitted with many solutions. Selection of the "best" solution is dependent on not only the application itself, but also on the specific way you want to handle your application.

What I'm trying to say is this: One technique will not solve the data-entry problems for all applications, nor will one technique solve all the data-entry problems for any one application. And further, the best technique for your application may not be the best technique for someone else's, even though his application and yours are identical.

Data entry is a difficult and fundamental system-design problem. But, as we must communicate man to man, and cross the language barrier to reach an understanding and an improved mankind, so must we communicate man to machine, and solve the data-entry proolem, to reach improved service, better control, reduced expenses, and greater overall company efficiency.

TECHNIQUE COMPARISON CHART

The chart on the following page compares the 15 techniques I described in this report. The charts cover operator considerations, field-identification considerations, computer considerations, and other considerations.

Comparisons of this kind are difficult, at best, because each technique has unique uses. It's a little like comparing apples and oranges. That is, although each technique helps to solve the data-entry problem, all have individual characteristics which place them in separate categories.

The comparison charts give a YES or NO answer for each question considered. Unfortunately, in some cases a YES or NO is not appropriate. Some questions are not applicable to some techniques, and some questions depend on the particular use of the technique. These cases are appropriately designated.

Some of the questions considered have answers that lie somewhat between YES and NO depending on precisely how the technique is handled. That is, certain answers are a shade of gray rather than a strict white or black. In these cases the most general use of the technique is assumed, and a YES or NO answer is given on this basis. In this regard, variable-length rather than fixed-length data fields are assumed.

A word of warning: there is often an inclination to count the good points for each technique and to claim the technique with the highest score as "best." This is invalid. The technique with the lowest score may have a unique feature that makes it the best choice for a particular application. Of crucial importance is the evaluation of a technique and its features relative to all the requirements of the application. There is no one best technique. However, there may be one best technique for you and your application.

Type A: Individual data field transmission
Type B: Collective data field transmission

KEY:

■	YES
	NO
N/A	Not applicable
D/U	Depends on use of technique

	OPERATOR CONSIDERATIONS									FIELD IDENTIFICATION CONSIDERATIONS							COMPUTER CONSIDERATIONS										OTHER CONSIDERATIONS								
	Must the operator enter data fields in a fixed sequence?	Must the operator remember the data-field sequence?	Must the operator remember all data fields for data entry?	Must the operator enter each field identifier?	Must the operator enter the code of the field identifier(s)?	Must the operator position the cursor for each data field?	Must the operator assure the cursor is specially positioned immediately before transmission?	Must the operator insert delimiters?	Are time restrictions imposed on the operator?	Is field identification the operator's responsibility?	Are the field identifiers displayed before data is entered?	Are field identifiers displayed on a field-by-field basis?	Are all field identifiers displayed after data is entered?	Is each field identifier next to the entered data field?	Must field identification be coded?	Are field identifiers implied?	Is only data sent to the computer; that is no identifiers or delimiters?	Are only data and delimiters (or codes) sent to the computer; that is no identifiers?	Are data and identifiers (or identifier codes) sent to the computer?	Are data fields sent to the computer in only one fixed sequence?	Are data fields sent to the computer individually?	Must the computer scan a portion of the data entered?	Must the computer scan each character of data entered?	Must the computer send a message to the operator before data entry begins?	Must the computer send a message to the operator for each data field entered?	Does the computer supply the data which the operator will enter?	Is the Nondestructive Cursor required?	Does the technique assure completeness?	Is the technique adversely affected by data-entry sequences requiring more than one full display?	Is the technique adversely affected by variability (within reason) of data-field length?	Is the technique adversely affected by absolute length of field (within reason)?	In general are the number of messages to the computer less than the number of data fields?	Are the number of messages to the computer equal to the number of data fields entered?	Are the number of messages to the computer greater than the number of data fields entered?	Does the technique have potential cursor-positioning problems?
Fixed-Form Technique with supplied field identifiers (Type B)						■	■			■	■		■	■					■	■				■			■			■		■			■
Fixed-Form Technique with implied field identifiers (Type B)		■	■			■	■			■					N/A	■	■			■										■		■			■
Form Check-or-Change Technique (Type B)						■	■			■	■		■	■					■	■				■		■	■	■		■		■			■
Free-Form-with-Delimiters Technique (Type B)	■	■	■					■		■					N/A	■		■		■		■	■									■			
Conversation-Mode Technique (Type A)	■									■		■	■	■			■			■	■			■	■			■					■		
Adjacent-Form Technique (Type A)					D/U	■	■			■	■		■	■					■		■	■		■			D/U	■		■	■		■		D/U
Adjacent-Form Technique with New Line Symbol (Type B)						■	■	■		■	■		■	■				■		■		■	■	■						■	■	■			
Adjacent-Form Technique without New Line Symbol (Type B)						■	■			■	■		■	■					■	■				■			■			■	■	■			■
Overwrite-Form Technique (Type B)						■	■	■		■	■							■		■		■	■	■						■	■	■			
Code-Identification Technique (Type A)			■	■						■		■	■	■					■		■	■											■		
Code-Identification Technique (Type B)			■	■				■		■		■	■	■					■			■	■									■			
Code-Link Technique (Type A)			D/U		■					■	■		■		■				■		■	■		■				D/U					■		
Code-Link Technique (Type B)			D/U		■			■		■	■		■		■				■			■	■	■								■			
Data-Selection Technique (Type A)	■									■		■					■			■	■	■	■	■	■	■			N/A				■		
Post-Identification Technique (Type B)			■	■				■		■			■						■			■	■						■			■			
Sequence-Key Technique (Type B)	■				■			■		■	■		■		■				■	D/U		■	■	■					■			■			
Content-Decode Technique (Type A)			■												N/A	■	■				■	■	D/U										■		
Content-Decode Technique (Type B)			■					■							N/A	■		■				■	■									■			
Cycle-Stop Technique (Type A)									■	■		■	■	■				■			■			■	■			■						■	
Cycle-Go Technique (Type A)										■		■	■	■				■			■			■	■			■						■	
Data-Bracket Technique (Type A)			D/U			■	■			■	D/U	D/U				D/U	■				■	D/U	D/U	■	D/U	■	■	D/U	N/A				■		■