KAYPRO

DATASTAR™ REFERENCE MANUAL

DataStar

Reference Manual

For DataStar Release 1.4

Third revision: 11/1/82

COPYRIGHT NOTICE

TRADEMARK

DISCLAIMER

References are made throughout this manual to the Control Program Monitor, commonly known as CP/M. CP/M is a trademark of Digital Research of Pacific Grove, California.

CONTENTS

INTRODUCTION

DataStar is an easy to learn, yet professionally powerful, comprehensive data entry, retrieval and update system for microcomputer systems. DataStar handles record keeping applications from initial design through updating, addition/deletion, and search/retrieval of records - quickly and easily, even for complex operations. DataStar is a powerful program that can be used for an extremely wide range of possible applications by experienced programmers or systems analysts as well as inexperienced first-time users. As an effective microcomputer data entry system designed to support high operator productivity, DataStar has no equal.

DataStar features include:

- **Comprehensive help messages and instructions displayed on the screen.** DataStar's help messages are designed to provide the assistance you may need without having to refer constantly to the user manual.

- **Extensive capabilities for use with other programs.** DataStar can easily be used as the data entry portion of your inventory, accounting or employee applications programs without requiring expensive modifications to your existing software.

- **Extreme flexibility in the design of data entry forms.** Your data entry format may be several pages in length or up to three pages wide, or some combination of length and width as desired. The size of the format is limited only by the amount of memory available on your system. You may even include instructions to the user as part of the form!

- **Field Verification and Edit Masking.** When you design the form to be used for data entry, DataStar allows you to specify a means of verifying the data to be entered. For example, you may specify the zip code field of a mailing list form to be verified against a list of zip codes. This feature allows you to prevent bad data from being entered by mistake. You might also specify an edit mask for your zip code field that would only allow numbers to be entered in that field. If an operator attempted to enter a letter by mistake, DataStar would not allow the letter to be entered. The field verification and edit mask features allow you to control the kinds of data entered and to make sure that the data is correct.

- **Search/Retrieval and Edit/Updating operations.** DataStar provides several ways to search through your records for any specific record you want. Once retrieved, the record may be edited and/or updated with additional information, or deleted entirely if desired.

- **Data entry power to support high volume operators.** DataStar is designed to support speed and accuracy, allowing fast typists to operate at their best rate. When verification and/or edit masks are used, DataStar promotes both exceptional operator speed and high accuracy on input data. Typing errors are quickly caught and eliminated before a problem can get started.

- **Batch Verification Processing.** DataStar allows you to enter a number of records into a temporary work file for later verification. Only correctly verified records will then be transferred into the main data file. This feature supports greater data entry speed by allowing you to complete the entry portion of a job and then verify the total entry instead of trying to do both steps together for each record. Batch verification supports greater accuracy by allowing a different operator to verify entered data.

- **Power and facilities usually found only in large key-to-disk systems.** DataStar has exceptional features that have previously be unavailable on microcomputer systems. For example, DataStar will allow you to access data files during data entry. This feature will perform functions such as automatic fill-in of data fields, so that an operator may only have to input the customer number – the rest of the standard information on that customer (e.g., name, address, billing terms, etc.) may be filled in automatically on the screen by DataStar. All the operator then has to do is to enter any new information into the form, and then go on to the next customer number!

- **Compatibility with most CP/M* supported programming languages,** including BASIC, FORTRAN, and COBOL. DataStar can be used to construct data files to be used with programs written in any of these languages. This feature allows greater programming flexibility and reduces programming costs.

---

* CP/M is a trademark of Digital Research Corp.

The DataStar system consists of two main program elements, DATASTAR and FORMGEN. Each program element has a distinct purpose and function within the system, and may be operated individually as required.

FormGen is used to prepare the form that will be used in entering data. As the name implies, FormGen generates a "form" to be "filled out" like a license application or a government tax form. FormGen allows you to design and create a form in many ways. You can define the datafields or "boxes" of your form so that only certain kinds of data can be entered. For example, a Zip Code datafield of a mailing list might be defined so that only the numbers could be entered. This kind of specification is called an "edit mask", and FormGen allows you to use a wide variety of different edit masks. Other possible form design specifications include:

- Maximum lengths of datafields
- Automatic decimal point alignment.
- Automatic generation of leading/trailing pad characters such as asterisks (*) used in check protection (e.g., Pay*****$1.00 Dollars).
- "Must Enter" fields or characters. This specification will require a data entry operator to make an entry in the datafield before going to the other fields. For example, you might require the customer number to be entered before any new orders for that customer could be entered.
- Inter-field arithmetic and/or character string operations. You could specify that the "Total Balance" datafield be automatically filled by subtracting the value of "Cash Withdrawals" from "Previous Balance" and adding "Cash Deposits." This means that the data entry operator would only have to enter the deposit and withdrawal data. The"Total Balance" datafield would then be calculated and entered automatically!

These are just a few of the ways that FormGen allows you to design your form.

DataStar is the name given to the second program element. The DataStar program handles the data entry and verification process according to the form requirements defined by FormGen. The DataStar program has several modes of operation, including: ADD records (Data entry), Select records by KEY (Retrieval), SCAN in Index order (Review file contents), SCAN in Data file order (Review file contents), etc. The ADD mode is used to enter records into the data file, or database. The Select-by-KEY mode allows you to search the database to retrieve a particular record by entering a specific part of the desired record. For example, if a customer number was entered, DataStar would search the database to find the record corresponding to that number. If the record was found, DataStar would display the entire record (i.e., customer name, address, orders, etc.). The SCAN modes allow you to examine the total contents of the database by displaying one record at a time.

DataStar provides you with extensive facilities for verifying and/or validating entered data. You may also specify whether the validation and/or verification will take place as each form is completed, or all at once "as a batch" similar to the way keypunch machines are used. Verifying allows you to double-check your work, while validating allows you to compare the data you typed against the data that is stored in a file. For example, a zip code might be validated against a stored list of zip codes to make sure that only correct codes can be entered.

HOW TO USE THIS MANUAL

This manual has been designed and written for both experienced microcomputer programmers and first-time operators. The topics have been arranged into sections according to the experience level of the user. A working example has been provided on the DataStar distribution diskette together with all necessary reference files. This example, called the ORDER form, has been used to illustrate the various concepts and operating procedures throughout the manual. You may find it useful to experiment with your system and the ORDER form as you read through the manual.

Chapters 1 and 2 of this manual are intended for the form designer. The form creation is described in Chapter 1, Beginning Form Design. Chapter 2 covers more advanced techniques of datafield definition. A summary of the datafield definition process is given in Appendix A.

Chapters 3, 4 and 5 are intended for the data entry operator. These chapters describe how to invoke DataStar; how to add records, and how to retrieve and modify previously entered records. The language of these chapters has been kept as simple as possible in order to assist inexperienced users.

The most complex material has been covered in the appendix. Information concerning installation and modification of DataStar is given together with program listings and file conversion techniques for experienced programmers.

Your comments and suggestions as to ways in which this manual may be improved are always welcome. Please forward your suggestions to Documentation Department, MicroPro International Corporation.

Chapter 1
Beginning Form Design

This Chapter describes the form generation process from initial entry procedures to completion of the designed form. Specification of individual datafield characteristics, such as what types of data may be entered into the field, etc., are covered in Chapter 2, Advanced Form Design. Chapter 2 also describes the error messages that may be displayed by FormGen, and the form listing command (^W).

The beginning form design section also covers help menus, cursor motion commands (Table 1-1), and procedures for character insertion, deletion and highlighting.

1.1 The Form Generation Process

The form generation process consists of designing a "form" to be used in entering data, and then entering the specifications for the form into a file through use of the FormGen program. A form is like a license or tax application in that there are "boxes" to be filled out by the user. Each "box" usually has a label that indicates what information the box should contain. The form constructed through FormGen is very similiar, although the CRT screen is used instead of paper, and the keyboard will be used instead of a pencil to "fill out" the form. FormGen is used to define the boxes and their labels that will appear on the screen when the form is used for data entry. We will refer to the boxes as "datafields" and to the labels as "background text" throughout this manual. The example below shows a common form design as it would appear on the CRT screen.

```
ACCOUNT NUMBER: ----------
Account Name: --------------------

QTY     ITEM          PRICE         TOTAL

-----   -----         ---------     ---------
-----   -----         ---------     ---------
```

The end result of the form generation process is a file containing the specifications of the form. This file will be used by the data entry program.

Before using FormGen, you may want to draw the proposed form layout on paper, including the length of the various datafields (boxes). This step will provide an easy reference for use while operating FormGen.

The size of the form to be designed is limited by the amount of memory available on your system (a minimum of 48K is required to run FormGen). The maximum size for a form layout is 255 characters wide OR 255 lines in length, but not both. A complete discussion of the memory requirements is given in the appendix.

1.2 Invoking FormGen

(Before use, be sure that you have properly installed FormGen for operation with the INSTALL program supplied on the distribution diskette. See Appendix B.)

After starting (booting) your system, wait for the system prompt (A>) to appear. To invoke Formgen, type:

```
FORMGEN
```

after the system prompt and press RETURN. The program will be read from the diskette and loaded into memory. Formgen will display the following prompt:

```
MicroPro FormGen release x.xx serial # FGxxxxxx
COPYRIGHT (c) 1980  MicroPro International Corporation

    (your CRT terminal name should appear here)

Enter name of form definition file (or press RETURN):
```

(If the name of your terminal does not appear, type ^C to reboot your system and run INSTALL to properly install FormGen.for use with your system.)

Enter a name for the form definition file and press RETURN. If RETURN is pressed before typing a name for the file, the following message will be displayed:

```
The Form definition file is where the specifications for your form are stored. If you
have not yet designed a form, choose a name for the form definition file and enter it
here:
```

You may avoid the filename prompt by invoking FormGen together with a specified filename, e.g.:

```
FORMGEN filename
```

1.3 Help Screens

FormGen has four Help Screens to assist the user. The Help Screens are rotated in sequence by entering ^J (press the CTRL key together with the J key), beginning with Help Screen 4. The purpose of the Help Screens is to avoid having to refer back and forth between the user manual and the screen. Commands for cursor movement, character insertion/deletion, on/off highlighting and entering datafields are given at the top of the screen for easy and rapid reference. The screens may be rotated at any time by entering ^J until the desired screen is displayed.

(The Help Screen display is set by the value of a variable called ITHELP in the program user area. ITHELP may equal 4, 3, 2 or 1 (default is 4). If the value of ITHELP is 3, Help Screen 4 will never be displayed; if ITHELP is 2, neither Help Screen 4 nor Screen 3 will be displayed, etc. ITHELP may be modified through use of your system debugger program.)

The texts of the Help Screens are given below.

Help Screen 4

```
                LIN = 000 COL = 000                          HELP SCREEN 4
This program is used to define a "form" for data entry. The form, much
like a tax or license application form, will consist of a number of
"boxes" to be filled in and background information to explain what goes in
the boxes. An example of datafields with background explanation is:

  First Name: _____      Last Name: _____________________

In the following procedure, you will be able to create a form for
data entry. You may move freely around the screen using the CURSOR
commands at the screen top, inserting background text at will. The
boxes/datafields are created using the underline or ^Q key. Note that
 1. you will not get a chance to actually enter data in this step and
 2. when the data does get entered, only the data in the datafields will
    actually get recorded. The rest is simply background text.

The help messages at the screen top list/explain most of the options
available. Use ^J to rotate the help display.

Datafields may be assigned various attributes via a special questionnaire.
To enter this phase, position the cursor at a datafield and type ^R.
Type ^J at any question for further information in this phase.
   Type ^J now to rotate to the next help screen:
```

Help Screen 3

```
    LIN = 000 COL = 000                                    HELP SCREEN 3
CURSOR:     ^A = left item     ^S = left char     ^D = right char    ^F = right item
            ^E = up line       ^X = down line     ^U = set/clear tab ^I = tab
DELETE:      DEL = char left   ^G = char right    ^T = entire column ^Y = entire line
INSERT:     ^P = line buffer   ^V = char right    ^B = entire column ^N = entire line
OTHER:      ^J = rotate help   ^W = list form     ^C = form done     ^K = toggle key
-------------------------------------------------------------------------------
```

Help Screen 2

```
    LIN = 000 COL = 000                                    HELP SCREEN 2
CURSOR:     ^A = left item     ^S = left char     ^D = right char    ^F = right item
            ^E = up line       ^X = down line     ^U = set/clear tab ^I = tab
FIELD:      ^Q,__ = add/extend field              ^Z = delete field  ^R = define field
HIGHLIGHT:  ^O = set/clear     ^L = delete block
OTHER:      ^J = rotate help   ^W = list form     ^C = form done     ^K = toggle key
-------------------------------------------------------------------------------
```

Help Screen 1

```
    LIN = 000 COL = 000                                    HELP SCREEN 1
-------------------------------------------------------------------------------
```

Help Screen 4 provides general information about the form definition process and FormGen operation. Procedures for assigning special attributes to datafields are discussed in Chapter 2, Advanced Form Design. Help Screen 4 only appears first when a new form definition filename has been entered. If the specified form definition filename already exists, the first help screen to be displayed will be Help Screen 3.

Help Screen 3 lists the basic cursor movement and character insertion/deletion commands together with a few miscellaneous commands.

Help Screen 2 lists commands for entering or deleting datafields and highlighting on/off commands together with the basic cursor motion commands given in Help Screen 3. The character insertion/deletion commands are not displayed. To redisplay these commands, use ^J to rotate the help display to Help Screen 3. Highlighting may be used if your terminal has either bright/dim screen display or inverse video (black characters on white background and/or white characters on dark background). The ^W and ^R commands are described in the Advanced Form Design Chapter.

Help Screen 1 displays only the Status Line (see Status Line subsection) to provide the maximum amount of display area. You may use ^J at any time to rotate the help screens to display the desired commands.

1.4 Window positioning

FormGen allows you to use the CRT screen as a moveable window to view or to create form layouts larger than the size of the screen. Forms longer and/or wider than the screen may be created or edited easily because the screen display is moved automatically as required by the movement of the cursor. To view form areas not displayed on the screen, simply move the cursor to the appropriate edge of the screen display and continue to enter cursor motion commands until the desired area is displayed. For example: to view areas to the right of the screen display, enter ^F or successive ^D's until the desired items appear on the screen. To view areas below the screen display, enter successive ^X's to scroll the display up as desired.

In other words, you may create or edit form layouts that are larger than the area displayed on the CRT screen (see Form Boundary Alterations section 1.9). The area displayed on the screen may be moved as desired by moving the cursor to the top, bottom or sides of the screen display and entering additional cursor motion commands. The screen display is shifted automatically to bring the cursor destination into view.

On terminals with 24 lines, the amount of available screen display area is 79 columns wide by 23 lines long when Help Screen 1 is displayed. These limits are referred to as the screen size. Help Screens 2 and 3 have the same width, but only 17 lines are displayed. The default form size is 79 columns wide by 17 lines long and matches the screen size for screens 2 and 3. It is important to note that extra columns or lines must be added to the form (see Form Boundary Alterations -section 1.9) in order to have a moveable display window within the form boundaries. If no extra columns or lines have been added, the cursor will automatically "wrap around" to the next line or to the top or bottom of the screen display in response to the cursor motion commands even when the screen size is greater than the form size.

1.5 The Status Line

The status line is displayed at the top of all Help Screens during FormGen operation. Information given in the status line includes: cursor location (LINe and COLumn numbers of the cursor position); datafield information (NUMber, LENgth, POSition number and EDit mask Characters); and the help screen number. The datafield information appears in the status line only when the cursor is positioned within a datafield.

LIN = ###

The LIN area lists the number of the line containing the cursor position in the form definition file. It is important to note that the number indicates the line of the form and not necessarily that of the screen. For example, a form may be up to 255 lines long so 255 is the maximum number that can appear after LIN while the screen can display a maximum of 23 lines at one time.

COL = ###

The column number of the cursor position is given in the COL field of the status line. As is the case with the LIN number, COL lists the column number of the cursor in the form, not the screen. The CRT screen normally displays 79 columns while a form may be up to 255 columns in width.

NUM = ###

The NUM area displays the number of the datafield as it appears on the screen; the first field on the form being 001 and so on. The maximum number of fields that can be created is 245.

LEN = ###

The number given after LEN is the total length of the datafield. The maximum length of a datafield is 255 characters because datafields can not be continued onto the next line.

POS = ###

The position of the cursor in the datafield is listed after POS in the status line.

EDC = ____

The EDit mask Characters are the characters in the entry and content control word associated with the cursor position in the datafield. See the Advanced Form Design Chapter for the explanation of the entry and content control words.

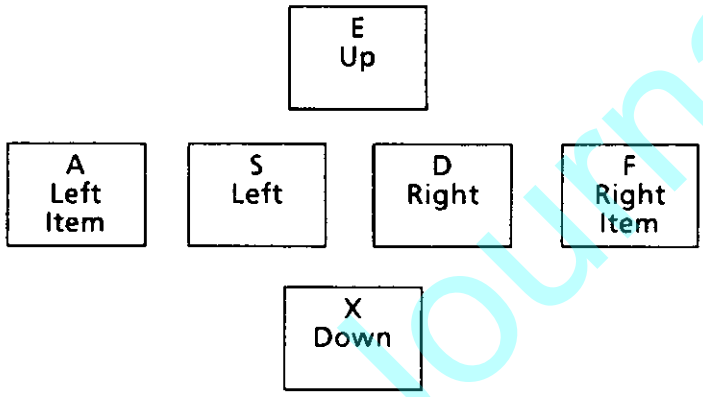
Help Screen #

There are four main help screens that are rotated in sequence by entering successive ^J's. The status line is the only message appearing during use of Help Screen 1.

The Status Line will normally only display the LIN and COL numbers unless the cursor is positioned within a datafield. The NUM, LEN, POS and EDC numbers are only displayed when the cursor is positioned within a datafield ***after the datafield has been created***. While a datafield is initially being created, these numbers are not displayed. The cursor must be moved back into a datafield in order to display the information for the field.

1.6 Cursor Motion

Cursor motion commands are used to place the cursor at a desired point on the screen to edit existing or enter new characters. All cursor motion commands move the screen display up, down or to either side as necessary to bring the cursor destination into view.



Note that the basic cursor motion keys ^S, ^E, ^D, and ^X, are arranged in a diamond on your keyboard. The position of the keys corresponds to the direction of cursor motion:

The A and F keys move the cursor more than one character position at a time. If the cursor is in a datafield when either ^A or ^F is entered, the cursor will be moved to the next datafield to the left or right. If the cursor is not in a datafield, then ^A will cause the cursor to be moved to the beginning of the next word to the left. ^F functions in the same way, except that the cursor will be moved towards the right.

All cursor motion commands will "wrap" around from the end of one line to the beginning of the next, and from the bottom of the form to the top and vice versa.

Table 1-1: Cursor Motion Commands

Command	Function	Description
^A	cursor left item	Moves the cursor one item to the left. If the cursor was in a datafield, ^A will cause the cursor to be moved to the first position of the next datafield to the left. If the cursor was not in a datafield, ^A will move the cursor to the beginning of the next word to the left.
^S, ^H	cursor left column	Moves the cursor one column to the left. If the cursor is in the first column of a line, ^S will move the cursor to the last column of the preceding line.
^D	cursor right column	Moves cursor one column to the right. If the cursor is in the last column of a line, ^D will move the cursor to the first column of the next line.
^F	cursor right item	Moves cursor one item to the right. If the cursor is in a datafield, ^F will cause the cursor to be moved to the beginning of the next datafield to the right. If the cursor is not in a datafield, ^F will move the cursor to the first position of the next word to the right.
^E	cursor up line	Moves cursor up one line while remaining in the same column.
^X	cursor down line	Moves cursor down one line, remaining in same column.
RETURN	cursor next line	Moves cursor to the first column of the next line below the current position.
^I	cursor right tab	Moves the cursor to the next tab stop to the TAB right. If there are no tab stops, the cursor is moved down one tab stop line while remaining in the same column.
CTRL/RUB	cursor left tab	Moves cursor to next tab stop to the left of the current position. Press both CTRL and RUB together. (May not be available on some terminals)
^U	set/clear tab	Sets or clears tab stops. To set or clear a tab stop, position the cursor in the desired column and enter ^U.

1.7 Form Construction

There are two different kinds of items involved in form construction, background text and datafields. Background text consists of instructions to the data entry operator as to what data is to be entered into which datafield. For example:

NAME: _____ Telephone Number: _____

The words "NAME" and "Telephone Number" are background text indicating what data is to be entered into the appropriate datafields.

Background text is entered by typing characters onto the screen at the desired locations. The entered characters will appear on the screen as typed, and will be saved in the form definition file when the form is completed. The background text will be displayed on the screen during data entry operations, but the data entry operator will not be able to move the cursor into any background text areas. During data entry, the cursor can only be moved within datafields.

It is important to note that the smaller the form is, the faster it will be loaded and used during the data entry phase. Extra blank lines or columns will adversely affect memory use and program load time.

Datafields will be used during data entry for entering data by the operator. Only characters entered into datafields are stored in data files; background text is only stored in the form definition file.

When designing form specifications, careful attention must be given to the lengths of the datafields. Each datafield, of course, must be long enough to contain the data to be entered there. If you discover later that a datafield was created too small, the length may be increased as needed provided that the datafield is not a key field. Extra care should be taken in designing the length of key fields as such fields cannot be altered later. (See section 1.10 for more information concerning key fields.)

Another matter that should be given careful consideration during the form design process is the order of the datafields in the form. Once the order has been established, and the form used for data entry, the form cannot be rearranged without losing all of the entered data unless a special program is used to rearrange the entered data to match the altered form. (See Advanced Form Design for more information about datafield order.)

Datafields are constructed through use of either the ^Q keys or by using the underline key (if available on your terminal). As the keys are pressed, a broken line will appear on the screen as shown below.

NAME: .....

If the cursor is moved back onto the line, the Status Line at the top of the screen will display the number of the datafield (in order of screen position), the total length and the position of the cursor within the field. Datafields may be deleted by positioning the cursor within the desired datafield and entering ^Z. Only the datafield will be deleted; the background text, if any, will not be affected. If ^Z is entered while the cursor is not located within a datafield, an error message will be displayed and the ESC key must be pressed in order to clear the error condition. Characters may be inserted or deleted anywhere on the screen by using ^V to insert and/or ^G or RUB to delete characters as desired.

^V inserts one space at the cursor position without moving the cursor. ^V will "drop" characters from the end of the line if there is not enough room on the line for the inserted space. After entering ^V, the desired character must then be typed to fill the space. ^V may also be used within datafields to create spaces, however, a warning message may be displayed if the attributes of the datafield are affected by the insertion. Inserting a space within a datafield will split the field into two parts, which may be rejoined by typing a ^Q or underline to fill the space.

^G may be used to delete characters and/or spaces as desired. When ^G is entered, the character at the cursor position is deleted, and the rest of the line to the right of the cursor position is moved one column to the left. A space is inserted at the end of the line. The cursor does not move. ^G can be used to delete successive characters or spaces by continuing to hold the CTRL and the G keys down until the desired number of characters have been deleted.

RUB moves the cursor one column to the left, and deletes the character at that position without moving the rest of the characters on the line. RUB is used to erase the previously typed character.

1.8 Highlighting

During the form generation process, the operator may select areas or characters to be highlighted when the form is used for data entry.

Highlighted areas may appear in one of two different ways; bright/dim display or inverse video, depending on your particular terminal.

Highlighting on bright/dim display terminals appears dim while non-highlighted characters or areas appear bright. Terminals with inverse video capabilities display highlighted areas as white background with dark letters, while normal text is displayed as white characters on a black background. Either mode will operate easily with DataStar.

Highlighting is entered by use of ^O. ^O is a toggle switch, characters are highlighted by positioning the cursor on the desired letter and entering ^O. The cursor is automatically moved one column to the right each time ^O is entered. If the cursor is placed on a highlighted character and ^O is entered, the highlighting for the character is turned off.

Blocks of highlighting may be turned off, or deleted from background text or datafields by using ^L. If the cursor is positioned within a highlighted block, ^L may be entered to turn the highlighting off. A highlighted block is a string of characters on a line that have been highlighted. A space may be marked as highlighted on bright/dim terminals even though the space will not display as highlighted until the cursor is positioned in the space.

1.9 Form Boundary Alterations

Although the maximum number of lines allowed in a form layout is 255, and the maximum character width of a form is also 255, the default form size is 79 columns by 17 lines. This means that when a form is created, that is the amount of memory allocated to the form. Even when Help Screen 1 is displayed, any attempt to move the cursor beyond line 16 by typing ^X will cause the cursor to "wrap" around back to line 0.

In order to create forms wider than 79 columns or longer than 17 lines, extra columns and/or lines must be added to the original boundaries of the form design area. The boundaries may be altered by inserting or deleting extra columns and/or lines through use of the following control keys:

^B, ^T

> Columns may be added to or deleted from the form design area by entering ^B to add columns or ^T to delete columns. ^T deletes a column at the cursor position while ^B inserts a column of spaces at the cursor position. It is very important to note that ^B inserts

columns, not just individual spaces on one line. Care must be taken in order to avoid entering unwanted spaces on every line in the form. It is recommended that columns be added only by first positioning the cursor in the last column (78) before ^B is entered. Likewise, columns should be deleted only from the last column of a line. Entering ^T in the middle of a line may produce undesirable results throughout the form layout as a character will be deleted from every line.

^N,^Y

^N and ^Y are used to add and/or delete lines from the form layout. ^N is used to add lines at any point on the screen; existing lines below the cursor position are moved down to make space. When ^Y is used to delete lines, any lines below the cursor position will automatically be moved up. ^N does not simply insert a blank line; instead, it inserts a copy of the last line of the form at the cursor position (see next paragraph). When constructing forms that are longer than 17 lines, you may want to mark the form end first by putting an "x" or some other character in the next to the last line of the form.

^P - Line Buffer

The last line of the form is treated as a line buffer. The ^P command extends the form by copying an entire line from the cursor position to the bottom of the form. The ^N command may then be used to insert the contents of the line buffer at the cursor position. By using these two commands, you may copy or move lines in the form as desired. Note that only one line at a time is stored in the line buffer. Position the cursor on the desired line and type ^P. The line will be copied at the last line of the form, and may be recalled by positioning the cursor at the desired line and typing ^N.

All of the form boundary alteration commands are slow to take effect. Caution should be exercised to avoid typing a second command, such as ^Y, thinking that the first command was missed. When using the form boundary alteration commands; type the command and then wait a second for it to take effect.

1.10 Key Field Creation

Every form created through FormGen is required to have at least one key field before the form can be completed or used for data entry. A key field is a datafield that has been specially set up so the records are sorted according to the data in this field and that data entered into the field can

be used as an index for the record. If there is more than one key field in a form, the data entered into the key fields will be combined to form the key.

For example, in a name and address form, the name field is usually designated as a key field. When the form is used later for data entry, the names entered into the key field are treated a little differently than the rest of the data (i.e., the address, telephone numbers, etc.). All of the data for the person, including the name, will be stored in a data file, but a copy will be made of the name entered into the key field. The copy will be stored in a different file, called the INDEX file, together with a pointer to the rest of the data for that name. This allows records to be accessed directly without having to scan through the entire data file. When the Select-by-Key mode of DataStar is used, the operator is instructed to enter only the key for the desired record. DataStar searches the index file until the specified key is located, then follows the pointer to find the rest of the data for that record in the data file, and displays the data on the screen. Since the index file is ordered, a fast search algorithm is used. For more information about key fields, see the Data Entry Chapter.

Selecting a key field is an important part of the form construction process as the keys will be used as the basis for the Select-by-Key mode of DataStar.There are three main considerations involved in designating key fields:

1. There must be at least one key field designated for each form. If no key field is specified, FormGen will display an error message, and will not mark the form as completed.

2. The key field must be unique for each record if you want the Select by Key mode to locate unique records. For example, if the last name field of an address file is the key field, a problem may result if there are records for more than one person with the same last name. DataStar will display only the first key encountered in the Select mode, ignoring all other duplicate keys. For an address file, the first and last name fields should be designated as key fields (or any other field that will contain data which is unique for each record.) FormGen allows you to specify that only unique keys may be entered by the data entry operator (See Section 2.04).

3. The number of key fields should be as low as possible, with a minimum of one key field per form. As DataStar will require that the total key be entered in the select by key mode, and entered exactly, it is recommended that the number of key fields be as low as possible. For example, if the first and last name fields and the

telephone number field are all designated as key fields, when the select by key option is used, ALL of the key fields must be filled out exactly. If any key field is left out, or incorrectly entered, DataStar will not be able to find the desired record.

In addition, the size of the index file increases as the key size increases, resulting in slower DataStar performance. The maximum key length is 120 characters.

To create a key field, position the cursor in the desired field and type ^K.The field will be filled with asterisks (*) to indicate key field status. To change the status of a key field, position the cursor in the field and enter another ^K. The asterisks will be removed, and the normal datafield line restored.

If more than one key field is desired, position the cursor in the fields and enter ^K in each field. The order of creation of key fields will determine the order of the values in the index file, and will also determine the order of entry in the select by key mode of DataStar.

You may designate key fields, or change key fields as desired before exiting from FormGen. Changes may also be made later ONLY if the form has not been used for data entry. You may not make changes in key fields to a form that has been used for data entry without re-entering all of the data previously entered or re-constructing the index file (see Appendix).

It is important to note that there is one and only one index file for each data file. Alternate index files are not maintained for the same datafile by DataStar.

1.11 Completing the Form

After all of the background text and datafields have been entered, and a key field established, the form may be completed by entering ^C. The cursor may be positioned anywhere, it is not necessary to move it to the end of the form before entering ^C.

Entering a ^C causes FormGen to display a list of errors, if any exist (see section 2.18); if there are no errors, the following prompt will appear:

```
Enter exit command:
   A = Abort without saving form              B = save form & Boot operating system
   C = save form and Continue                 D = save form and chain DataStar
   SPACE = continue without saving form
                                              (A/B/C/D/SPACE):
```

If A is entered, the current version of the form is abandoned and the operating system prompt will appear. The "A" command is used to abandon the current edit. To avoid accidental loss of your work, FormGen will require a second step by displaying the following prompt:

```
Abandon form edit? (Y/N)
```

Before exit will occur, a Y must be typed. If N is entered, FormGen will ignore the exit command. If you are sure you want to quit, you may avoid these two prompts by rapidly typing ^CAY.

If B is entered, the form will be saved, and the operating system prompt will be displayed. The B command is used to take a break in the middle of entering a large form.

If C is entered, the current version of the form will be written on the disk, and then re-displayed on the screen for further entry or editing. The C command is used to save partial results in the middle of entering a large form, and is particularly useful where you don't trust your system to stay up for long periods of time.

If D is entered, FormGen will store the designed form in a file on the disk and automatically invoke DataStar. The D command is used when form entry is complete. At any time during the loading of DataStar, a ^C may be typed to exit to the operating system instead.

If a space is entered, the exit command (^C) is ignored and form edit may be continued. The space command is used when ^C has been entered by mistake.

In all cases where the form is saved (B/C/D), FormGen automatically renames the old version of the form to "filename.BAK" before writing the current form to the disk.

If FormGen detects errors or omissions in the form design, a list of errors will be displayed on the screen. If the list is longer than one screen, FormGen will wait after filling the screen for a character to be typed before continuing. At the bottom of the error list, the following prompt will be displayed:

```
Enter exit command:
   A = Abort without saving form          B = save form & Boot operating system
   C = save form and Continue             L = List errors on LST: device
   SPACE = continue without saving form
                                          (A/B/C/L/SPACE):
```

The A, B, C and SPACE commands are described on the previous page. If L is entered, the error listing that is displayed on the screen will be printed on the line printer. NOTE: The line printer must be set as the CP/M LST: device in order to use the L command. Make sure that the printer is ON and ready before entering L.

Any of the above commands may be typed as the error message listing is being displayed on the screen. If a command is typed, the error display will be aborted and the command performed; it is not necessary to wait for the entire error list to be displayed before the next command may be entered.

A complete description of the error listing is given together with a list of possible error conditions in section 2.18. The only two error conditions that can occur using the commands explained in this chapter are:

No Key Field

and

Key field longer that 120 characters.

All of the exit commands (A/B/C/D/L/SPACE) will be accepted in upper case, lower case and or control characters. They may be typed immediately after the ^C without waiting for the exit menu to appear.

Chapter 2
Advanced Form Design

This chapter covers the advanced form design process of assigning special qualities to individual datafields through use of the ^R (Field Definition) command. Together with Chapter 1, this chapter completes the documentation of the FormGen program and the form design process. It is recommended that the preceding chapter, Beginning Form Design, be read before attempting to read Chapter 2.

The ^W command to print the form design on a line printer is explained near the end of this chapter. A listing of possible error conditions is also given, together with recommended procedures for resolving errors.

2.1 The Datafield Definition Process

Chapter 1, Beginning Form Design, described the way datafields and background text are created and portions of them highlighted. This chapter covers the attributes, or special characteristics, that may be assigned to individual datafields. Attributes are assigned through use of the ^R command, and are entered into the form definition file by the FormGen program.

An attribute is a special characteristic that is assigned to an individual datafield during the form design process. An example of an attribute is an entry control mask that will permit only specified characters to be entered into the datafield when the form is used for data entry. Another example is the derived-by calculation attribute, where the value of the field is the result of an arithmetic or string operation involving other datafields and/or constants. The instructions controlling the operation of the datafield are interpreted by DataStar, and are automatically executed whenever the form is used. Attributes are stored along with the form in the form definition file.

Attributes are assigned by FormGen in a series of question and answer prompts. The sequence is begun by positioning the cursor within the desired datafield and typing ^R. Each query lists a default answer at the cursor position, which may be selected by pressing RETURN. If the datafield has already been defined through use of ^R, the default answer for each query will be the answer given during the last assignment session.

This allows the user to make changes in the field definition while retaining any earlier assignments as desired. You may also use the ^R sequence to determine the current assignment for each attribute of a datafield by positioning the cursor within the datafield, typing ^R, and pressing RETURN for each query.

It may be useful to know that if a field is defined, and then later split by inserting spaces, both new fields will have the same attributes as the original field. Also, if two datafields are joined together, the attributes of the second datafield will be lost. The combined datafield will have the attributes of the first datafield.

2.2 Help Screen R

To begin the process of assigning attributes to a datafield, position the cursor within the datafield and type ^R. The ^R command is listed only in Help Screen 2, although you may type ^R to begin the process from any help screen except Help Screen 4. After typing ^R, the following menu will be displayed:

```
  LIN = ### COL = ### NUM = ### LEN = ### POS = ### EDC = ___HELP SCREEN R
CURSOR: RETURN = next item     ^S = left char     ^D = right char    ^E = prev item
OTHER:   ^A = previous field   ^F = next field    ^G = delete char   ^V = insert char
         ^C = end definition   ^R = start over    ^Q = locate field  ^J = more help
-------------------------------------------------------------------------------------
```

FormGen will display between two and four lines of the form on the screen (depending upon the location of the datafield in the form), with the field that is being defined shown in the middle line.

The cursor motion commands appearing on Help Screen R apply to the attribute responses. In most cases, the attribute response is on the same line as the attribute question. However, when defining items that assign a different value to each character position within the field, the attribute response is located in the image of the form that appears on the screen. In this case, the ^V and ^G commands apply only within the field even though other fields are displayed on the screen.

You may type ^J at any time during the field definition process to display more information about the current step. After displaying the help message, type any character to return to the field. A complete listing of all of the help messages is given in the Appendix.

The discussion on the following pages does not cover the attributes one by one. Instead, in certain cases, the attribute assignments required to obtain particular results are described. The subsection headings list the options that are available. In each subsection, the attribute is defined together with the assignment procedure and a description of the attribute's effects when the form is used for data entry through DataStar. Some attributes require other fields to be also specially defined, e.g., as in calculated datafields. In this case, the procedure will include instructions for properly defining any other required datafield.

The examples used are taken from those available on the DataStar distribution diskette. You may want to invoke FormGen, and use the form definition file ORDER to display the examples on your screen.

Table 2-1: Help Screen R Commands

Command	Function	Description
RETURN	enter as displayed, go to next query	Enters the current query response as displayed and proceeds to the next question. The current response entered may be either the unaltered default or whatever the user may have typed in.
^S	cursor left one column	Moves cursor one space to the left within the current response field only. If cursor is already in the leftmost position in the field, the command is ignored.
^D	cursor right one column	Moves the cursor on column to the right within the current field. If cursor is in the rightmost column of the field, the current response is entered and the next question is displayed.
^E	return to previous query	Returns to previous query, retaining answers to current query. May be used successively to "back-up" to the beginning of the sequence.
^A	move to previous field	Moves cursor to previous field while remaining in the field definition process. Allows you to check answers given during definition of previous field, edit if desired.
^F	move to next field	Moves cursor to next field while remaining in the field definition process.
^G	delete character	Deletes one character at the cursor position and moves any characters to the right of the cursor position one column to the left.
^V	insert character	Inserts the current default character at the cursor position, moving characters to the right of the cursor as necessary to make room.
^C	end definition	Ends the query sequence when entered, saving any answers to previous queries.
^R	start over	Returns to the beginning query of the sequence, remembering all previous answers.
^Q	locate field	Moves cursor into current field in the form image. Most commonly used to provide a quick visual check to make sure that the right field is being defined. Type any character to return to the query sequence.
^J	more help	Displays a brief help message at the top of the screen describing the current query. Type any character to return to the query sequence.

2.3 Field Order

The field order query is the beginning prompt of the field definition sequence. Position the cursor in the desired field and type ^R. The field order query will be displayed below the Help Screen R and the two lines of the current form as shown below.

```
  LIN = ### COL = ### NUM = ### LEN = ### POS = ### EDC = ____HELP SCREEN R
CURSOR: RETURN = next item     ^S = left char     ^D = right char    ^E = prev item
OTHER:  ^A = previous field    ^F = next field    ^G = delete char   ^V = insert char
        ^C = end definition    ^R = start over    ^Q = locate field  ^J = more help
-------------------------------------------------------------------------------------
     Field order:  001
```

The default value of the field order query is the order of the datafield as it appears in the form. In the above example, the default value of the field order query for the first Date field would be 002 as this field is second on the form.

The field order determines the order of data entry in DataStar, and also the order of values in the data file. When the form is first presented in DataStar, the cursor will be positioned in field 001 (unless the field is derived).

To enter the field order number, press RETURN to use the default value or type in a new number. Only characters 0-9 may be used; if any other key is pressed an error message will be displayed.

The field order number may not be lower than 001 nor higher than the total number of fields in the form. Numbers lower than 001 or higher than the total number of fields will be ignored and replaced by the closest legal value (e.g. 001 or the highest field number).

The next query will automatically appear when RETURN is pressed or a value is entered (unless the field is a key field, in which case 2 or more different prompts may intervene–see next section):

```
Copy attributes of field  001
```

The "Copy" query allows you to automatically copy all of the query responses of the specified field number to the field currently being defined. This feature is especially useful where two or more fields are to

have the same or nearly the same attributes. If a number is typed here, the attributes of the named field are instantly copied. The default responses to the following queries will be those of the named field. The key order and key-related queries are not copied.

Subsequent queries will be displayed below the field order prompt until the screen area is filled. When this occurs, the screen area is automatically cleared and a new page is begun.

2.4 Key Order, Refuse Duplicate Keys, Tie Breaker fields

These queries will appear only when the cursor has been positioned within a key field for the field definition sequence. If your form has only one key field, the key order query will not occur. The key order query will appear only when a key field is being defined, and only when there is more than one key field in the form.

```
 LIN = ### COL = ### NUM = ### LEN = ### POS = ### EDC = ____HELP SCREEN R
CURSOR: RETURN = next item      ^S = left char      ^D = right char     ^E = prev item
OTHER:  ^A = previous field     ^F = next field     ^G = delete char    ^V = insert char
        ^C = end definition     ^R = start over     ^Q = locate field   ^J = more help
--------------------------------------------------------------------------------------------
Order #: ______    Date (M/D/Y): __/__/__    Customer #: ______
--------------------------------------------------------------------------------------------

    Field order:                  001
    Key order:                    001
    Refuse duplicate keys? (Y/N)  Y
```

The above example would occur if the the Order # field were being defined and if there were another key field on the form. The default value shown indicates that the field was the first to be designated as a key field.

The key order is the order in which the key fields are joined together in the index file to form a key. When the Select-by-Key mode of DataStar is used, the cursor will be positioned in the first key field at the start of the key field entry. During data entry, the values entered into the key fields will be copied into the index file according to the order established by this query.

To enter the key order number, press RETURN to enter the default value or type in a new number and press RETURN. Only characters 0-9 may be

used; if any other key is pressed an error message will be displayed. The key order number may not be less than 001 nor more than the total number of key fields in the form. Numbers higher than the total number of key fields or lower than 001 will be ignored and replaced by the closest legal value (e.g. 001 or the number of key fields in the form).

If a key field is split by a later modification to the form, the key order will be modified to reflect the split by automatic renumbering of all of the other key fields. Any other attribute assigned to the original field will be assigned to both new fields created by a split.

The "Refuse duplicate keys" query allows the form designer to require that only unique key values may be entered into the field. If the data entry operator types a value into the key field that already exists in the index file, an error message will be displayed. The check for duplicate keys occurs after any verification processing has been done, and is the last step before a record is written into a disk file. If a batch file is being used (see Chapter 5), the check for duplicate keys is not performed; duplicate keys are not refused until the batch file is verified.

If there is more than one key field in the form, an additional query may be displayed before the "Refuse Duplicate Keys" query. This query will only appear during the field definition process of the least significant key field (i.e., the field with the highest key order number) of the form:

Tie breaker field? (Y/N)

A tie breaker field is a key field entered by DataStar when necessary in order to create a unique key. The cursor will not be positioned in this field during data entry. In addition, DataStar will only complete the field when a record with the entered key already exists. DataStar fills the field with the smallest integer that will create a unique key. Note that the process of finding a unique key is slower than simply recording the data; for this reason we recommend that you do not overuse this feature.

If a field is assigned the tie breaker attribute, the list of additional attributes that may be assigned to the field is reduced. In such cases, the only other attributes that may be assigned to the field are:

Right justify
Pad field
Record edit characters.

If an additional key field is added to a form that already contains a key field with the tie breaker attribute, the key field with the attribute will

lose the tie breaker definition as the field will no longer be the least significant key field. For example, if a form had three key fields, with #3 having the tie breaker attribute, and then a fourth key field was added; the tie breaker attribute would be automatically removed from field #3 when the fourth key field was designated.

2.5 Calculated Datafields

A calculated datafield is a datafield whose value is derived from the result of an internal operation rather than directly entered by the operator during data entry. Calculations may involve data located in other fields of the form, or constants. A calculation may be either numeric or string (alphabetic characters).

To define a calculated datafield, several elements of the calculation must be established. The query sequence for a calculated datafield begins with the prompt displayed below:

Field derived? (Y/N) **Y**

Before beginning the query sequence, we recommend that all of the fields that will be involved are identified and defined if necessary. The two important questions that need to be answered before defining a calculated datafield are: Which fields (or constant values) are involved? What operations are to be performed using the values? The fields or constants involved are referred to as operands.

After the "Field derived?" query is answered (Y), the following query is displayed:

Allow operator entry? (Y/N) (see section 2.7)

If a Y answer is entered, the data entry operator will be able to move the cursor into the field during data entry. If a N answer is selected, the cursor can not be moved into the field during data entry. After the operator entry query is answered, the next query is automatically displayed.

List/Calculated? (L/C)

Enter C. (See section 2.6 for list derived datafields.)

Calculations in DataStar are specified by the use of expressions. An expression is comprised of operators and operands; operators are the

actions to be done, while operands are the "building blocks" used by the operators. For example, in the expression 2 + 2 the two's are the *operands* and the "plus" (+) is the operator. These terms will be used frequently in the discussion on the next pages. It is important that these terms be clearly understood as the various messages that may be displayed on the screen will use either or both terms to describe specific conditions.

NUMERIC EXPRESSIONS

A numeric expression is used whenever a field value may be derived from other field values and/or constants by performing some arithmetic operation or series of operations. Numeric expressions are defined through the responses to two queries. To begin defining a numeric expression, type "N" in response to the prompt displayed below:

```
Numeric/String (N/S): N
```

After N is entered for the above prompt, the following prompt will appear:

```
Enter algebraic expression for field calculation:
  #XXX =
```

where XXX represents the number of the field being defined. The expression is entered after the " = " sign. The following paragraphs describe the method for entering an algebraic expression. If you are already familiar with expressions, you may skip ahead to the subsection on "rounding" without missing any important information.

Operands:

A numeric operand in DataStar is either a datafield or a constant. Datafields are represented by the pound sign (#) followed by the number of the datafield with any number of leading zeroes. Examples of legal field numbers for the ORDER form supplied on the distribution diskette include:

```
#1
#010
#00000007
```

Examples of *illegal* field numbers for the same form include:

#0	(first field must be #1)
5	(not preceded by the "#" sign)
#33	(number is greater than the total number of fields in the form.)

Constants are expressed as a series of numbers, with one optional decimal point and/or an optional leading minus sign (-). Examples of legal constants include:

1
-1.023
127
15999234.666

Examples of *illegal* constants include:

1.10.06	(only one decimal point is allowed)
+ 234	(leading plus signs not allowed)
15-	(minus sign must be in front of number)

Operators:

Numeric expressions are formed by alternating numeric operands and operators. The following operators (functions) may be specified for use with numeric fields or constants: add (+), subtract (-), multiply (*), divide (/) and exponentiate (^). Right and left parentheses () may be used to enclose parts of expressions , e.g. (#8 * 4.5 - 2). The value of field 8 would be multiplied by 4.5 first, and then 2 would be subtracted from the result.

Examples of legal expressions:

Enter algebraic expression for field calculation:

#010 = (#8 + #9)*.15	[add field 8 to field 9 then multiply the result by .15 to find the value of field 10.]
#10 = #2	[copy field 2 to find the value of field 10]
#010 = #5*#6	[multiply the value of field 5 by the value of field 6 to find the value of field 010]

Precedence in numeric expressions:

A numeric expression is not simply evaluated from left to right. Instead, certain operations take precedence over others. In DataStar (as in most programming languages), the order of precedence is as follows:

any part of the expression that is enclosed in parentheses ( ),
any exponentiation,
multiplication and/or division
addition and/or subtraction

This means that a part of an expression that is enclosed in parentheses has the highest precedence in the evaluation of the expression, and will be resolved first before any other part of the calculation is performed. In cases where more than one set of parentheses are used ("nesting"), the innermost set will be resolved first; otherwise they are evaluated from left to right. In the example given below, the order of precedence is as follows:

```
#010 = (#4*(#3 + #1))-#6
         B      A       C
```

Field #10 is equal to–the sum of field #3 plus field #1–multiplied by the value of field #4–MINUS the value of field #6. Operation A is performed first, followed by B, and then C.

Rounding:

DataStar does not round the results of numeric calculations. This factor brings up two important points about numeric calculations performed by DataStar:

First, in order to express a non-integral result, a decimal point must be defined in the field (see section 2.16, Edit Masks). If a decimal point is not defined in the field, the non-integral part of the result will be chopped off or truncated and lost.

Second, in order to obtain the effect of rounding-off of results, the effect must be explicitly specified within the calculation. This may be done by adding 1/2 of the smallest value that can be contained in the field to the result of the calculation.

For example:

#010 = (#4 + #5 + #6)/3 + 0.5	will round the average of fields 4,5,6 and store the result as an integer.
#010 = (#4 + #5 + #6)/3 + 0.005	should be used for rounding when the result is to be put into a field with two decimal places.

Precision:

DataStar calculations are carried out to 14 digits. The method used is binary coded decimal so that precision is never lost, even for the most complicated expressions.

The only limit on expression complexity is that the expression must fit on the line provided for defining the expression.

STRING CALCULATIONS

A string calculation is used whenever a field value may be derived by joining parts of other fields and/or character string constants. To define a character string constant, type S in response to the prompt shown below:

```
Numeric/String (N/S)    S
```

The following query will appear:

```
Enter string expression for field:
#xxx =
```

where xxx represents the number of the field being defined. The following paragraphs describe the procedures for defining string expressions.

Operands:

String operands may be fields, sub-fields, and/or string constants. A field in a string expression is represented by a pound sign (#) followed by a

field number exactly as in a numeric expression. (See numeric operands.) A string sub-field is used whenever only part of a field is needed. A sub-field is represented by a pound sign and field number, followed by a starting position and number of characters, separated by commas and enclosed in parentheses. For example; #003 = #2(1,3) will cause the value of field 3 to be found by taking the first three characters of field number 2. If the number of characters is not specified, the remainder of the field will be assumed. Examples of legal sub-field expressions include:

#005(1,2)	the first two characters of field #5
#5(3,1)	the third character of field #5
#5(4)	all but the first three characters of field #5

Examples of *illegal* sub-field expressions include:

#5(0,4)	position 0 can not exist in DataStar
#5(1,300)	maximum field length is 255
#5(1,4	missing right parentheses

A string constant is any number of characters enclosed in quotation marks ("). The quotation mark character itself is represented by two quotation marks (""). Examples of legal string constants include:

"a ""legal"" string constant"

"#!'* + "

Examples of *illegal* string constants include the following:

"needs a trailing quotation mark

"single embedded (") marks are not allowed"

Operators:

The only operator for string values is JOIN. The symbol used to represent joining is the ampersand (&).

Expressions:

A string expression is defined by alternating string operands with the string operator. Fields, sub-fields and constants may be joined as desired. Examples of legal string expressions include the following:

```
#006 = #4&"/01/"&#5          [join the value of field #4 to /01/ and
                              field #5 to find the value of field #6.]

#006 = #3 " per pound "& ", "&#9&" per ton"
                              [The value of field 6 will be the value of
                              field #3 followed by the constant (per
                              pound) and a comma and the value of
                              field #9 followed by the constant (per
                              ton) –e.g., 1 per pound, 2000 per ton.]
```

ERROR MESSAGES

There are three error messages which may occur concerning calculated datafields:

Illegal operand. Hit ESC key.

Illegal operator. Hit ESC key.

Unclosed left parentheses. Hit ESC key.

When any of the three error messages are displayed, after the ESC key is pressed, the calculation will also be displayed with the cursor positioned at the illegal area or symbol of the expression. The cursor position indicates the first illegal position of the expression.

Illegal operators should be self explanatory. There must be an operator between all operands, and only the following symbols may be used as operators in expressions:

```
numeric:  + - * / ^
string:   &
```

Illegal operands may occur when the field number specified in the expression does not exist (e.g., the number is higher than the total number of fields in the form, or lower than 001) or because the operand is not recognized as one of the legal types defined in the previous paragraphs.

Unclosed left parentheses can easily be corrected by inserting the right parentheses to close the expression or sub-expression.

Self-referenced fields:

If the field being defined is itself used as a part of the calculation expression to find the value of the field, it is called a self-referenced field. For example:

```
Enter the algebraic expression for field calculation:
    #002 = #002 + 100
```

When this field (002) is encountered during data entry, the final value of the field will be equal to the value of the field in the previous record plus one hundred. Self referenced fields are most often used for automatic incrementing (e.g., record numbers or running totals). In the ORDER form, the order number field is automatically incremented whenever a new order is entered. This illustrates a common use of the self-referenced field capability.

EXAMPLE QUERY SEQUENCE

An example sequence for a calculated datafield is given below. The example used is from the form definition file ORDER provided on the DataStar distribution diskette.

```
  LIN = ###  COL = ###  NUM = ###  LEN = ###  POS = ### EDC = ____HELP SCREEN R
CURSOR: RETURN = next item     ^S = left char  ^D = right char     ^E = prev item
OTHER:  ^A = previous field    ^F = next field ^G = delete char    ^V = insert char
        ^C = end definition    ^R = start over ^Q = locate field   ^J = more help
-------------------------------------------------------------------------------
  Tax rate: __%  Sales tax: ________     Total: ----------
-------------------------------------------------------------------------------

      Field order:                  030
      Copy attributes of field      030
      Field derived? (Y/N)          Y
        Allow operator entry? (Y/N) N
        List/Calculated? (L/C)      C          (for List, see section 2.12)
        Verify/Calculate order?     012        (see section 2.13)
        Numeric/String (N/S)        N

      Enter algebraic expression for field calculation:
      #030 = #29*(#23 + #28)/100
```

When this form is used for data entry, the value of field 030 (Sales Tax) will be automatically entered during the end of record processing. The cursor can not be moved into field 030.

One final note about calculated datafields: In order to install an entered expression in a form definition, you must press the RETURN key. If the expression is accepted as correct, it will then be properly installed. If ^C or ^E is typed instead of RETURN, the entered expression will be lost.

2.6 File Derived Datafields

In many data entry applications, certain data entry items are repeated over and over again. An example of this is the client name and address in a billing system , or product description and price in an order entry system. It is useful in such cases to avoid having to retype the same information many times. DataStar provides a facility, called a file derived datafield (FDD), that allows such data items to be automatically entered without being retyped by the operator.

An FDD is a datafield whose values are obtained by reading values from a separate datafile rather than being typed by the data entry operator. If one of a group of related data items (such as product code, price, description) is specified, DataStar will retrieve the rest from a specified datafile. This facility allows greater data entry speed and increased accuracy. For example, such an arrangement would result in the correct product price being entered every time, and/or notify the operator that an entered product code was not on file.

Implementing an FDD requires several steps. First, a datafile with the appropriate information must be created. (A datafile is a collection of records, where each record usually describes a single item or event.) In the client file example, each record would contain a name and address; in the product file example, each record would contain code, description and price.

Second, one of the items in the record must be selected for use as a key to the record's contents. For example, the product code might be selected for use as a key; DataStar would search the datafile until the particular key was found, and then use the other items in the specified record as values for FDD's. The key to the record should be unique in the file; if there are identical keys within a datafile, DataStar will simply use the first one that is encountered and ignore all others. The item selected as the key is called a ***file index***, all other items in the record are called ***file data***.

The third step in implementing FDD's is to create an association between the items in the data file and the datafields in the form. The file index (key) must be associated with an index field in the form, and any other item in the datafile record to be retrieved must be associated with an appropriate FDD. These associations are created by assigning attributes to the various datafields on the form. Procedures for completing the attribute list will be given later.

CREATING REFERENCE DATAFILES:

There are at least two ways to prepare a file for use as a DataStar reference file. The first method may be used for converting existing files; the second may be used to create new files.

If a file already in existence has the appropriate information, it may be converted for use as a reference file. First, the file format must be checked; DataStar records have variable length fields separated by commas and terminated by the carriage return and line feed characters. In addition, any fields containing a comma or a quotation mark (") as a part of the field must be enclosed in quotes (e.g., "Alameda, California"). This is standard CBASIC format; if the file is not in this format, a simple program must be written to convert the file. The Appendix contains instructions on how to convert a NAD file along with a sample conversion program. Once the file is in standard CBASIC/DATASTAR format, an index file must be created to go along with it. There are two ways in which to create such an index file; the first is to use SuperSort with the "KP FIXED" command, while the second is to create a form for the file through FormGen and use DataStar's VERIFY BATCH facility. Both methods are described in detail in Appendix D.

If a file does not already exist with the appropriate information, DataStar itself can be used to create it. Use FormGen to design a form and then use DataStar to enter the file. (See Chapter 3, Data Entry.) DataStar will create both the index and the data file. After all of the data has been entered, use the file maintenance command, ^EF (See Section 3.17) to put the file into the form that will optimize performance.

Whichever method is used to prepare the reference file, as long as a form has been created to go along with it, DataStar can be used to maintain the file. This includes deleting records as they are no longer required, adding new records or modifying information as it changes.

When the form for the reference file is created, only one field may be selected as a key field; this is the file index. It must be exactly as long as the index field in the form that will use the reference file. These two conditions are very important as DataStar will not retrieve data for FDD's if the conditions have not been met. In addition to failing to retrieve data, DataStar will NOT display any error messages; the FDD will simply not work. If SuperSort is used to create the index file, these two conditions must still be satisfied.

Associating reference file data with datafields:

Although the reference file records may contain information that is not needed by the current form, the usual case is that one item in the record will be used as the file index while the rest of the record items are used to fill FDD's. Each item in the file record is given a number, beginning with 1, called the item number. This number is used by DataStar to identify which item of data goes into which FDD on the form. The item number is requested in the datafield definition query sequence for the index field and all of the FDD's.

Defining the Index Field:

There are two points that must be specified in order to have a datafield act as an index field for an FDD. These are the name of the reference file and the item number of the file index. Note that the file index does not have to be the first item in each record; it can be anywhere. The index field must be given the *list verify* attribute in order to display the queries concerning the reference file name and the item number. (See section 2.12 for detailed instructions.)

Defining the FDD:

After the index field has been established and correctly defined, the actual file-derived datafield can be set up. The example shown below is Field #6 of the ORDER form supplied with the DataStar Distribution diskette. To display the example on your screen, position the cursor within the Customer field (6) and type ^R to begin the query sequence. Press RETURN to enter the default values. The following queries will be displayed:

```
Field order:                          006
Copy attributes of field              006

Field derived? (Y/N)                  Y
    Allow operator entry? (Y/N)       N
    List/Calculated? (L/C)            L
      Index field number:             005
      Item number in list:            001
```

The ALLOW OPERATOR ENTRY attribute is for fields that sometimes require non-standard values (See section 2.07 on operator entry). The index field number is the datafield number of the field chosen to be the index field for the FDD's in the form, and has been defined with the list verify attribute (to specify the name of the reference file to be used). In this example, the Customer Number field (#5) is used as the index field. During the field definition process for field #5, the field was defined with the list verify attribute. The file CUSTOMER was named as the file containing a list of legal values for field #5. The first three records contained in the CUSTOMER file are displayed below.

```
Automated Products,23467 Hollywood Blvd.,Los Angeles,CA,90039,1111111  <
Business Electronics,1700 Fifth Ave.,New York City,NY,10011,2222222    <
Computer Devices,1455 Van Ness, San Francisco,CA,94926,3333333         <
```

When the ORDER form is used for data entry, and a customer number is entered into field #5, DataStar will access the CUSTOMER file. After verifying that the entered number exists in the file, DataStar will take the first value of the record containing the specified customer number and automatically enter it in field #6. Notice that the response to the "Item number in list?" query (above) is 001, indicating that the first value in the specified record is to be used. The address fields which follow field #6 are also FDD's; the street address, city, state and zip code values will all be automatically entered after a legal customer number is entered into the index field (#5).

2.7 Operator Entry

The operator entry option is only appropriate for derived datafields, and the query will only appear if a Y answer is entered for the "Field derived?" prompt. As the value for a derived datafield is the result of an internal operation, normally operator entry is neither needed nor desirable. However, for some complex applications, operator entry of values into derived datafields is appropriate. An example where operator entry might be required is a discount rate field. In this example, such a field is normally filled in from the customer file, but allowance is also made for the operator to enter special discounts for sale items.

If the operator entry option has not been selected, the cursor can not be moved into a derived datafield during data entry. The cursor motion commands available in DataStar will act as if the derived datafield was not present on the screen.

If the operator entry option was selected, the cursor will normally bypass the derived field and go on to the next datafield. To allow the operator to enter data into the derived datafield, the cursor motion commands must be used to "back up" into the derived datafield. NOTE: If the derived datafield is the LAST field in the form, the operator entry option is useless because the cursor can not be moved past the field.

To select the operator entry option, answer Y to the "Allow operator entry?" query.

2.8 Required Entry Datafields

There are two ways to define a datafield so that the data entry operator is required to enter a value during data entry. The effect of a required entry datafield is seen during data entry; the record is not considered complete until all required fields have been entered.

The first way to define a required-entry datafield is to enter a Y response to the "Required? (Y/N)" query. The required-entry query is displayed immediately after the "Field derived?" sequence.

The second way to define a required-entry datafield is discussed in section 2.16, Edit Masks.

A Y response to the "Required?" query should be entered in order to define a required entry datafield where no edit mask will be specified. If

an edit mask (to control the types of characters that can be entered) is to be specified, skip the "Required?" query by entering an N response.

Either or both methods may be used as desired, however, the purpose of the separate "Required?" query is to allow a required-entry datafield to be defined without having to specify an edit mask.

2.9 Justification

Characters are normally entered into a datafield from left to right. However, it is often useful to have characters entered from right to left, especially for numeric datafields not involving decimal points. This attribute is called right justified.

When the data entry operator types characters into a right justified datafield, the cursor is positioned following the rightmost position of the field. Typed characters appear and are "pushed" to the left while the cursor remains in this position.

If pad or float characters (see Sections 2.10, 2.11) are used, the justification characteristic will affect the way the pad and/or floating characters are entered and displayed. Right justification may be used with any field as desired, and in any combination with other special characteristics.

To specify right justification for a datafield, enter a Y response to the "Right Justify? (Y/N)" query. The justification query is displayed immediately after the "Required?" query.

2.10 Pad Characters

Pad characters are characters that are automatically entered to "fill up" or pad a datafield. For example, asterisks (*) are commonly used to pad datafields containing monetary amounts (e.g., bank checks) for protection.

```
Pay Two dollars and twenty cents**********
```

The pad character query sequence is in two parts, and appears immediately after the justification query. The following prompt is displayed first:

```
Pad field? (Y/N)
```

If a Y is entered, the following query is displayed:

Enter pad character:

Any character may be used to pad a field as desired. Enter the character to be used as a pad character. If an N is entered in response to the "Pad field?" query, the second prompt will be skipped.

Pad and float characters are not normally recorded in the datafile in order to save space on the disk. However, a later query (displayed after the Float character query sequence) provides the option of storing the pad and/or float characters in the datafile. A Y answer to the following query will cause the entire contents of the datafield, including any pad or float characters, to be stored in the datafile.

Record edit characters? (Y/N) **N**

The default response is N. Press RETURN to cause the pad and/or float characters to be stripped, or type Y to store the characters in the datafile.

2.11 Float Character

A float character is a character that "floats" to the left (or right, depending upon the justification) of any value entered into the datafield. For example, a datafield that will contain monetary amounts will often be defined with a floating dollar sign ($) that appears to the left of the numbers (e.g., $2.20). The $ character is moved as numbers are entered so that the sign is always just to the left of the leftmost number. Another example of a floating character is the # sign, commonly used for non-monetary numeric fields (e.g. #177, #23, etc.). Any character may be used as a floating character.

Float characters, like pad characters, are defined in a two step query sequence that appears immediately after the pad character queries.

Floating character? (Y/N)

A Y answer will cause the following prompt to be displayed, an N response will cause the prompt to be skipped:

Enter floating character:

Enter the character that will "float" in the datafield. When the form is used for data entry, the floating character will be displayed in the field to the left of the cursor position.

As is the case with pad characters, floating characters are not normally recorded in the datafile. To record floating and pad characters, enter a Y in response to the following prompt:

```
Record edit characters? (Y/N)     N
```

A Y response will cause DataStar to record the floating character, together with the entered value, in the datafile.

2.12 Verification

There are three different verification types, called modes, that are available for use with data entry. The verification modes are sight, retype and list verify. The verification process can occur either all at once as a "batch" or after each individual record is entered. The selection of batch versus individual record verification can be made either during the form design process or at data entry time. The selection of verification mode for each field is made during the form design process. The different verification modes may be interspersed throughout the form as desired. When verification processing (either batch or individual record) is begun, the process follows the calculate/verify order in performing whatever verification modes that were selected by the form designer.

Verification modes:

Sight verification mode requires the data entry operator to move the cursor through the datafield a second time to visually check the accuracy of the entered data. After the values have been initially entered, Datastar positions the cursor in the first sight-verify datafield to begin the sight verification pass. The operator is then required to use the cursor motion commands to move the cursor through all of the specified datafields one more time before going to the next record. Sight verification is usually specified when records are to be verified as each one is entered.

Retype verification mode requires the data entry operator to retype the specified datafield a second time after the value has been entered. After all datafields have been entered, the fields are cleared one at a time of the entered values and the cursor is positioned at the first position of the

datafield. The operator is then required to re-enter the value. As each value is re-entered, DataStar compares the re-entered value with the previous value. If an error is detected, the re-entered value is deleted and the operator is required to try again. Retype verification is generally used when records are to be verified in a batch.

The list verify mode causes each value entered into a specified datafield to be checked against a list of values kept in a separate file. If the entered value is found in the specified file, the value is accepted. If the entered value is not found in the separate file, the entered value is rejected. The comparison occurs as soon as the value is entered, so that any derived fields may be immediately displayed. (A common use for a list verified datafield is as an index for a file-derived datafield on the same form. In this case, the values entered into the list verified datafield are used as keys to search the specified file to locate a particular record. When the record is located, specific values from the record are automatically entered into the file-derived datafields. See section 2.6, File Derived Datafields.)

For example: if a customer number datafield is defined for list verification, the master customer list file is usually specified as the verification file. During the verification process, each customer number entered into the specified field is checked against the master list of all customer numbers. If the entered customer number is found in the master file, the number is accepted. If the entered number is not found, the entry is rejected. If the data entry process is done as a batch, the correct as well as the incorrect records are written to the batch file. When the batch is verified, if there are no sight or retype verify fields, the correct records will be automatically transferred to the main file. If there are any incorrect records, they will be displayed on the screen with an error message and the cursor will be positioned at the incorrect datafield. If the verification is done record by record as entered, incorrect values are rejected during the end-of-record processing.

In order to list verify a datafield, a datafile containing all legal field values must be created. Section 2.6, File Derived Datafields, describes the procedure for creating a reference datafile. A brief summary of two important points concerning reference files is given here; a complete description is given in section 2.6.

First, notice that if the file is to be used only to verify a datafield, each record need only contain a legal field value, terminated by the carriage return and line feed characters. (These terminator characters are automatically supplied if the file has been created through DataStar.) It is also acceptable for the reference file to have more than one item per record, in case such a file already exists or is needed for deriving FDD's.

Second, remember that the file can have only one field chosen as a key field per each record, and this field is the one that will supply the legal values for the list verify datafield. The field in the datafile must be exactly the same length as the list verified field in the form.

Entering Verify Attributes:

The verification query sequence is displayed after the pad and floating characters attribute query sequences. This example is from field #16, the PO# field of the ORDER sample form. The first query in the verification sequence appears below:

```
Verify field? (Y/N)              Y
  Sight/Retype/List (S/R/L)      R
  Batch verify? (Y/N)            N
  Verify/calculate order:        005
```

The batch verify option is only presented if sight or retype verify attributes are selected. This option serves to force the data entry operator into batch mode for data entry. Although the query is presented for every sight/retype verify field during the definition process, a yes answer for a field will require the option for all fields; a NO answer will reverse the status for all fields. If any datafield has the batch verify attribute option selected, the data entry operator will no longer be free to select online entry. If the batch verify attribute is not selected, the data entry operator can choose to enter the records into the main file with immediate verification, or choose batch entry mode with verification as a separate step.

For example, the sample form (ORDER) does not contain any datafields with the batch verify attribute. For this reason, the operator may or may not select a batch file as desired. If the form did contain a datafield with the batch verify attribute, the operator would be required to select a batch file in order to use the form.

Field #14, the second state datafield of the ORDER form, is another example of a verified field. The query responses for this field include:

```
Verify field? (Y/N)              Y
    Sight/Retype/List (S/R/L)    L
    Keep file in memory...       Y
    Enter list file name:        OKSTATES
    Enter list file disk:        -
    Item number of field:        001
    Verify/calculate order:      004
```

The list verify attribute requires specification of four attributes that are not required by either Sight or Retype verification. These four attributes are: whether or not the file is to be kept in memory; the name of the file; the disk drive the file is located on; and the item number in the disk record of the index field.

Whenever the reference file is small enough to be kept in memory, it should be. Not only will data entry be speeded up, but if the reference file is smaller than about 500 characters, keeping it in memory will save memory space.

The filename must be letters or numbers; other characters are not permitted. When file access is performed, both the data and the index files are required, and the file types are assumed to be DTA and NDX.

The allowable responses to the disk drive query are the characters A-P or a hyphen. However, most systems will only have two disk drives (A, B). Do not enter a character for a disk drive which will not be present on your data entry system. If a hyphen is entered, the current logged disk drive will be assumed when the form is used for data entry. If the file is not found on the current logged drive, the operator will be prompted to specify a disk drive.

The "Item number of field" query is for datafiles that have more than one item in each record. If the datafile named above is only a list of legal values, then use the default (001) for this query. However, if the records contain more information, specify which item of each record is to be used as the list value. Item numbers may range from 001 to 255.

2.13 Verify/Calculate Order

The purpose of the verify/calculate order is to allow the form designer to have control over the order in which fields are presented for verification. In addition, in cases where one calculation depends on the results of another, it is absolutely essential for the form designer to be able to schedule the sequence of operation in order to produce the desired results.

The calculation and verification orders are combined so as to allow file derived datafields to be used as part of calculations, and for the results of calculations to be used to index a file for a derived field. It is recommended that verification be performed first (unless otherwise

required, as is the case with the latter example) so that any calculations will be performed using verified values.

The verify/calculate order is the sequential order of execution for verification or calculation operations during the end-of-record processing. All operations having a verify/calculate order are performed in verify mode. For example, if a batch file is selected for data entry, the operations will be performed when the batch is verified. If the main file is selected, operations will be processed as the operator completes entering each record. Whatever entry mode is selected, operations will be processed when verify mode is entered in the order specified by the query.

The Verify/Calculate order query will only appear when the specified datafield is defined as a calculated or verified field. The query appears after the field verification prompt sequence. The order is specified by entering the appropriate order number in response to the following prompt:

Verify/calculate order:

The default value is the order in which the field was defined as a calculated or verified datafield.

2.14 Check Digit

The check digit query is displayed immediately after the "Verify/calculate order" query if the field being defined is verified, or after the "Verify field?" query if the field is not assigned a verify attribute.

Although the check digit query is always displayed, check digits should only be implemented for datafields whose values will always be numeric.

The check digit attribute requires that the values entered into the defined field by the data entry operator be divisable by 11 with no remainder. For example, the values 11, 22, 33, 44, etc are all valid entries for a field that has the check digit attribute. The values 12, 25, 31, etc., are not valid entries for such a field.

The purpose of a check digit is to make certain that only those entries which obey the check digit rule (for DataStar, entries divisable by 11) will be accepted as valid. Since it is unlikely that incorrect entries will obey the rule, the accuracy of data entry is therefore increased.

If the data entry operator attempts to enter a numeric value that is not divisable by 11 into a datafield that has the check digit attribute, an error message will be displayed.

2.15 Range Check

The range check query occurs after the check digit query, and is always displayed. The range check query allows a minimum and a maximum value to be specified for the datafield (e.g., a "low" and "high" range). A range check may be specified for alphabetic as well as numeric fields. This is because the comparison is alphabetic rather than numeric. The range check values are entered into the field in the form image rather than to the right of the queries. This serves as a reminder that the range check is positional in the field (i.e., field values are not justified before performing the range check). This factor is usually only important when the low value is entered, as the high value generally fills the entire field. For example, if a range check is specified for a numeric field whose legal values may range from 7 to 65, care must be taken when entering the low range number. With a two position datafield, a low range check value of 7 might be entered as either _7 or 7_. The first method (_7) is correct, while the second will result in an error condition because DataStar will assume that the low value is to be 70, not 7. Blank character positions in datafields act as "wild cards" for the purpose of setting high and low range check values, accepting any character. The exact collating sequence is given in the Appendix.

A range check is defined through the responses to the following prompts:

Range Check? (Y/N) Type Y to display the following prompts, or N to skip the range check sequence.

Enter/Change the minumum field value:

Enter the lowest acceptable value for the datafield. When specifying the minimum field value, the character "_" is considered to be lower than the lowest character in the collating sequence.

Enter/Change the maximum field value:

Enter the highest acceptable character for each position in the datafield. The underscore character is considered to be higher than the highest character in the collating sequence.

If the data entry operator attempts to enter a character that is lower or higher than the specified value range, an error message will be displayed that gives the minimum and maximum values for the datafield.

2.16 Edit Masks

The edit mask characteristic allows the user to specify controls on entering characters and also on the content of specified datafields. For example, the operator may be *required* to enter a value into a field, and *only certain kinds* of characters are allowed to be entered. The edit mask query consists of three prompts. The reponse to the first prompt either begins the sequence (if Y) or causes the last two queries to be skipped (N). A Y response displays the following prompt:

Entry Control Character Codes:

! = must enter a character
X = automatic copy
" = constant in this position
' = constant if data on both sides of constant
_ = may enter or leave blank
Y = auto-copy/may enter
~ = constant/may enter
/ = constant if data on one side only

Enter/change the entry control mask:

The cursor is located in the first character position of the field, which is displayed on the screen. Each character position may have an entry control code as desired. If control codes are entered into every position, the next prompt is automatically displayed. If not, press RETURN to display the next prompt.

Enter the appropriate entry control code for each position of the field as desired. The cursor will automatically be moved to the next position after each character is entered. A description of the effects of each code is given below:

_ This is the default character for the field. If no other character is entered, there will be no entry control.

! This code requires the data entry operator to enter a character in this position in the field.

X, Y The automatic copy feature causes DataStar to copy the value from the previously entered record. If X is chosen, the operator will not be permitted to enter or edit the character in this position during data entry. (However, the operator may use the Edit Scan Mask mode of DataStar to enter characters in the position -- see section 4.6 EDIT SCAN MASK.)

", ~ The constant codes allow the designer to specify a character to be used as a constant. The actual character to be used will be entered during the next query. If " is chosen, the operator will not be able to enter or edit the character at this position in the field; if ~ is chosen, the operator will be allowed to override the constant with an entered character.

', / These two codes allow the form designer to specify a conditional constant to be entered only under certain circumstances. If the specified conditions are not met, the constant will not be entered. Again, the actual character to be used as a constant is entered during the next step.

Content Control Character Codes:

A = A-Z only	a = a-z only
B = A-Z, space	b = a-z, space
C = A-Z, a-z ->A-Z	c = A-Z, a-z
D = A-Z, a-z ->A-Z, space	d = A-Z, a-z, space
E = A-Z, 0-9	e = a-z, 0-9
F = A-Z, 0-9, space	f = a-z, 0-9, space
G = A-Z, 0-9, a-z ->A-Z	g = A-Z, 0-9, A-Z
H = same as G with space	h = A-Z, 0-9, a-z, space
9 = 0-9 only	8 = 0-9, #$%()* + -.
. = align decimal point	_ = any character allowed

Enter/change the content control mask:

Enter the appropriate control code for each position in the field. The form line containing the datafield is displayed on the screen, and the cursor will be moved as the code for each position is entered.

The content control mask specifies the range of legal characters which may be entered into the positions in the field by the data entry operator. If an illegal character is typed by the operator, an error message will be displayed on the screen together with a list of the legal characters for the position.

Note that C, D, G and H provide automatic conversion to upper case, as well as restricting the allowed character set.

The acceptable character set for the alphabetic content control characters A-H and a-h can be extended by modifying a file contained in the DataStar customization package.

In addition to the character codes given above, the designer may also specify constants to be entered automatically by DataStar. If a constant were specified as an entry control mask, the actual character to be used as the constant is entered in the content control mask. For example, if the " code was entered in response to the entry control query, the designer would enter the actual character to be displayed as the response to the content control query. An example of the use of a constant in a field is a hyphen for telephone number fields (e.g., 415-457-8990). The total length of the datafield is 12 positions. The entry and content control masks required to produce such a field are given below:

!!!"!!!"!!!!	entry control
999-999-9999	content control

The hyphens are automatically displayed on the screen in the appropriate positions. (Note that the operator would NOT be able to move the cursor on to the positions occupied by the hyphens. The cursor would automatically skip over positions 4 and 8 in the datafield.) This is not how decimal points are entered into datafields -see paragraph below.

Conditional constants are used to insert constants in a datafield depending upon the number of characters entered into the field. Two examples where this feature is useful are amount fields and telephone extensions. Commas are conditional in amount fields:

-'---'---	entry control
9,999,999	content control

Note that this mask is always used in a right justified field. A telephone number with extension would use the following mask in a 13 character field that was left justified:

!!!"!!!!'----	entry control
999-9999X9999	content control

Decimal aligned fields are created when the "." is entered as a content control character.

NOTE: IN ORDER TO STORE A NON-INTEGRAL CALCULATED NUMBER IN A DATAFIELD WITHOUT TRUNCATING THE DECIMAL PORTION, THE DECIMAL ALIGNMENT CHARACTER MUST BE PRESENT IN THE FIELD.

The position of the period in the content control mask determines the precision to which the calculated result will be stored. (See section 2.5, Calculated Datafields, for information on decimal places and rounding.)

It is important to note that in order to use the aligned-decimal feature, the corresponding character position in the entry control mask should NOT be defined as a constant. If the position is defined as a constant or a conditional constant, then the period entered in the content control step will be treated as a constant rather than as an aligned decimal.

Only one decimal point will be permitted in the content control mask.

Additional examples of edit mask usage are available in the ORDER form. Almost every field in the ORDER form uses an edit mask. (See section 2.17, printout #3 for a listing of the edit mask examples in the ORDER form.)

If the edit mask query is answered with a Y response, the "Record Edit Characters" query will be displayed after the edit mask is entered. Unless the edit characters are to be recorded, all constants (conditional or otherwise) will be stripped from the entry before it is stored in the datafile.

2.17 Listing the Form (^W)

At any point in the form design process, the form can be printed on the line printer (if available) by typing a ^W.

NOTE: The CP/M LST: device must be assigned to the correct device in order to list the form on the line printer. If the LST: device is not set correctly, the system will stop (hang), requiring re-booting if ^W is typed.

Typing a ^W will cause the form to be printed in seven different ways, each showing a separate category of information for every field in the form. The form image is printed five times, showing field numbers, upper and lower range limits and edit masks, followed by a field attribute table and a list of calculations. A sample ^W printout, using the example form design provided on your distribution diskette (ORDER) is shown on the following pages.

If you do not want all of the printouts, you can cause FormGen to skip unwanted printouts by typing a character for each printout to be skipped. For example, if only the list of calculations was desired, typing four characters (any 4) after typing ^W would result in skipping the first four printouts. Type the extra characters rapidly to avoid partial printing of unwanted printouts.

1. Printout #1; Field Numbers:

```
Order #: _____1        Date (M/D/Y): _2/_3/_4        Customer #: _____5

   Bill to: _____6                    Ship to: _____11
   Address: _____7                    Address: _____12
   City:    _____8                    City:    _____13
   State:  _9   Zipcode: __10         State:  14   Zipcode: __15

P. O. #: _____16   Ship via: _____17        Terms: _____18

Quantity    Product      Description          Unit Cost  Total Cost

  _19         __20      _____21               ____22     _____23
  _24         __25      _____26               ____27     _____28

      Tax rate: _29%    Sales tax: _____30          Total: _____31
```

The field number of the datafield is given in the last position of each individual field. The number shown is according to the field order; which may be either the order in the form or the order specified through the field definition process. In datafields that are only one character long,fields 10-245 print as "*". In two-character datafields, field numbers 100-199 print as "A0"-"J9" and field numbers 200-245 print as "a0"-"e5".

2. Printout #2, Range Check

The low range check printout lists the lowest and highest acceptable values in each position of a datafield where a range check was defined. This printout will not be present if there are no range checks in the form.

```
                            RANGE CHECK, LOW

Order #: ______          Date (M/D/Y): Ø1/Ø1/__      Customer #: ______

   Bill to: _________________           Ship to: _________________
   Address: _________________           Address: _________________
   City:    _________________           City:    _________________
   State:   __    Zipcode: ____         State:   __    Zipcode: ____

P. O. #: __________     Ship via: _________          Terms: _________

Quantity   Product       Description              Unit Cost  Total Cost

  ___       ____     ______________________        ______    ________
  ___       ____     ______________________        ______    ________

      Tax rate: ___%    Sales Tax: _______           Total: _________
```

```
                          RANGE CHECK, HIGH

Order #: ______        Date (M/D/Y): 12/31/__     Customer #: ______

   Bill to: __________________         Ship to: __________________
   Address: __________________         Address: __________________
   City:    __________________         City:    __________________
   State:   __   Zipcode: _____        State:   __   Zipcode: _____

P. O. #: __________    Ship via: __________        Terms: __________

Quantity    Product         Description            Unit Cost  Total Cost

  ___        _____    ________________________      ______    ________
  ___        _____    ________________________      ______    ________

      Tax rate: __%    Sales Tax: ________         Total: ________
```

3. Printout #3, EDIT MASK

The edit mask printout shows each datafield position where an edit mask has been specified. The specific control character is displayed at the appropriate location. This printout will not be produced if no edit masks were specified in the form.

```
                          ENTRY CONTROL MASK

Order #: ______        Date (M/D/Y): _1/!!/"X     Customer #: ______

   Bill to: __________________         Ship to: __________________
   Address: __________________         Address: __________________
   City:    __________________         City:    __________________
   State:   __   Zipcode: _____        State:   __   Zipcode: _____

P. O. #: __________    Ship via: __________        Terms: __________

Quantity    Product         Description            Unit Cost  Total Cost

  ___        __/_     ________________________      ______    ___'_____
  ___        __/__    ________________________      ______    ___'_____

      Tax rate: __%    Sales Tax: __'_____         Total: ___'_____
```

```
                        CONTENT CONTROL MASK

Order #: 9999999          Date (M/D/Y): 99/99/89        Customer #: 9999999

   Bill to: _____._____                  Ship to: Cddddddddddddddddddd
   Address: _____________                Address: _____________________
   City:    _____________                City:    Cddddddddddddddddddd
   State:   __   Zipcode: _____          State:   CC   Zipcode: 99999

P. O. #: HHHHHHHHHHHHH  Ship via: DDDDDDDDDDD          Terms: HHHHHHHHHHH

Quantity    Product        Description                 Unit Cost  Total Cost

  999         CC-GC    _________________________      ____.__   ____,___.__
  999         CC-GC    _________________________      ____.__   ____,___.__

                                                                 Total: ____,___.__
        Tax rate: ___%    Sales Tax: __,___.__
```

4. Printout #4, Field Attribute Definitions

The Field Attribute printout lists the attributes of every datafield in the form according to the individual field numbers. Background text is not shown. The discussion following the example below will be in five parts: field identification; entry characteristics; added characters (pad & float); derived (list/calculated); and verification. The discussion of the example will proceed from left to right, beginning with the field identification area.

FIELD ATTRIBUTE DEFINITIONS

FLD	LEN	LIN	COL	KEY	Q=required C=check dgt J=right just W=write ed c O=oper entry R=range chk E=edit mask	PAD/ FLOAT	* * D E R I V E D * LIST INDEX FIELD	LIST ITEM NUM	CALC ORDER	******VERIFICATION**** ORDER	LIST VERIFY FILE NAME
001	007	000	009	001	JWO E	P0			001 N		
002	002	000	036		Q JW RE	P0					
003	002	000	039		Q RE						
004	002	000	042		Q W E						
005	007	000	063		Q J E					L 002	CUSTOMER
006	020	002	012				005	001			
007	020	003	012				005	002			
008	020	004	012				005	003			
009	002	005	012				005	004		L 003	OKSTATES
010	005	005	027				005	005			
011	020	002	047		Q E						
012	020	003	047		E						
013	020	004	047		E						
014	002	005	047		E					L 004	OKSTATES
015	005	005	062		E						
016	012	007	009		Q					R 005	
017	012	007	034		Q E						
018	011	007	059		Q E						
019	003	011	002		Q J E					S 006	
020	005	011	013		Q W E					L 007	PRODUCTS
021	025	011	022				020	002			
022	007	011	050		J E	P F$	020	003			
023	011	011	059		J E	P F$			008 N		
024	003	012	002		J E					S 009	
025	005	012	013		W E					L 010	PRODUCTS
026	025	012	022				025	002			
027	007	012	050		J E	P F$	025	003			
028	011	012	059		J E	P F$			011 N		
029	003	014	017		E		009	002			
030	009	014	036		E				012 N		
031	011	014	059		J E	P F$			013 N		

Field Identification:

The leftmost area of the printout identifies the various fields of the form by number, beginning with the first field of the form. For example, the sample shown above lists field #1 of the ORDER form, and gives the total field length (007), the line of the form where the field is located (000), the column of the first position of the field, and identifies the field as a key field. Note that the background text associated with field #1 is not considered.

Entry characteristics:

The entry characteristics area of the printout identifies the entry attributes assigned to each field in the form. Fields are listed by number from the left side of the printout. The attributes are identifed by one character codes shown at the top of the entry characteristics area; e.g. Q = required, C = Check digit, etc. There is a separate column for each code, beginning on the left with Q and proceeding as follows: Q-C-J-W-O-R-E. If the code is not listed in the appropriate column for a given field, the field does not have that attribute. For example, in the above sample printout, field #1 is identified as having the following entry characteristics: J (right justified), W (the edit characters -pad & floating- are to be stored in the datafile together with the actual entry), O (the operator may enter characters in this field) and E (edit masked).

Added characters - Pad/Floating:

The next area of the printout (at approximately the middle column of the page) is devoted to the added characters that may be assigned to a datafield, e.g., pad or floating characters. The format for these characters is as follows: P (for Pad) followed by the pad character assigned; F (for Floating) followed by the assigned floating character. Remember that the pad character may be used more than once in a field while the floating character only appears once. If the pad character is a blank space, no character will be shown after the P in the printout. The character used for a floating character will appear in the column after the F.

Derived datafield characteristics:

The derived datafield characteristics area is divided into two subsections: one for file derived datafields and the other for calculated datafields. The first category lists the associated index field number, followed by the item

number of the value in the datafile record. (See section 2.6 File Derived Datafields.)

The next category in the derived datafield characteristics area is the Calculation Order. Only fields whose values are the result of numeric or string calculations will have entries in this area. The type of calculation is identified immediately after the order number, e.g. N or S for numeric or string calcuations. Note that the order numbers may or may not be in logical progression (i.e. 001,002,003 etc); this is because the calculation order is combined with the verification order (given in the last area of the printout).

Verification:

The last area of the printout, Verification, identifies the characteristics of datafields whose values are verified in some manner. The first column of this area identifies the verification mode assigned to the datafield, e.g., S/R/L. The ORDER example does not show the column for the "B" which will appear after the S/R/L of every verified field if the form is to be batch verified. The order of verification is given next. Remember that the order of verification is combined with the calculation order given in the area to the left.

The rightmost column of the printout identifies the file names for list verified datafields. If a disk drive has been specified, it will precede the file name.

5. Printout #5, Calculations

The calculations printout lists the algebraic and/or string expressions that were entered during the field definition process. The format of this printout is as follows:

[field number] = [operand] [operator] [operand] [operator] etc.

CALCULATIONS:

#001 = #1 + 1
#023 = #19*#22
#028 = #24*#27
#030 = #29*(#23 + #28)/100
#031 = #23 + #28 + #30

2.18 Error Conditions

^_ Unimplemented control character. Hit ESC key:

The wrong control character was entered. Press ESC and retry.

Cursor is not in a field. ^_ ignored. Hit ESC key:

Either ^K, ^R or ^Z was typed when the cursor was not located within a field. Press ESC, move the cursor within the desired field, and retry.

At line limit. ^N ignored. Hit ESC key:

The maximum number of lines allowed is 255. Remove some blank lines in the form and retry.

Cannot delete last line. Hit ESC key:

A minimum of one line must exist at all times. This message will be displayed when ^Y is entered to delete the last line.

At column limit. ^B ignored. Hit ESC key:

The maximum number of columns allowed is 255. Remove some blank columns and retry.

Cannot delete last column. Hit ESC key:

A minimum of one column must exist at all times. This message will be displayed when ^T is entered with only one column remaining.

Not enough memory, ^_ ignored. Hit ESC key:

This message will occur if ^B, ^N, ^Q or ^R is entered and there is not enough memory available to execute the specified operation.

Possible fatal program error.
Please report occurrence, ^_ ignored. Hit ESC key:

This message should never appear. If it should be displayed, and you can re-create the occurrence again, please contact your dealer or MicroPro International. If the error is specific to your form, please include a copy of the form.

***** WARNING: Edit mask or range limits may need revision. Hit ESC key:**

If the field size of a datafield with an edit mask or range check is changed, this message will be displayed as a warning.

The maximum number of fields is 245, ^_ ignored. Hit ESC key:

This message may occur if ^Q is entered to create another field after field #245, or if an attempt is made to split an existing field where the maximum number of fields has already been reached.

Can't read form definition file.
Replace system disk, type RETURN:

Indicates hardware problem or serious program bug.

Insufficient memory.
Replace system disk, type RETURN:

There is not enough memory present in the system to read the form definition file into memory. This should only happen if memory has been removed from your system since the form was created, or if you are using a version of FormGen that is different than the one used to create the form. This could also occur if you change operating system versions. Replace the system to its original configuration, and use FormGen to reduce the form definition size.

Directory full.
Replace system disk, type RETURN:

The maximum number of files already exists on the disk. Move some files off to another disk to make room. This is usually not a problem since the number of files on a disk usually doesn't change during FormGen operation. The exception here occurs when creating a new form or when increasing the size of a form beyond 16K.

Disk full.
Replace system disk, type RETURN:

Move some files off to another disk to make room. Watch your disk space! If you use FormGen with a full disk, many hours of work may be lost as there is NO WAY to recover from this condition other than re-booting the system. Use the Form Save command frequently (See section 1.11 Completing the Form) to avoid losing everything.

The following error messages are only displayed during the field definition query sequence. The first five are messages for illegal field calculation expressions. The last message can occur on any query.

Illegal operand. Item must be
a field between #001 and #---, or
a literal enclosed in quotes. Hit ESC key:

Occurs when an incorrect field number is entered for a string calculation. The cursor will be positioned at the incorrect operand. This may occur when:

1. field number > number of fields or = 0
2. invalid subfield representations
3. character string not enclosed in quotes
4. missing operand (two "joins" in a row, or first character of an expression is the join operator.)

Illegal operator. Item must be &. Hit ESC key:

The only legal operator for string expressions is &. Change the incorrect operator and retry. This condition occurs whenever the first character following a legal string operand is not &.

Illegal operand. Item must be
a field between #001 and #---, or
a constant using only the digits 0 through 9, and optional decimal
point and leading minus sign. Hit ESC key:

Illegal numeric operand. This will occur when:

1. field number > the number of fields, or = 0.
2. illegal constant used.
3. missing operand (two operands in a row, or leading character if an expression is an operator.)

Illegal operator. Item must be +-*/^. Hit ESC key:

The only legal numeric operations are add (+), subtract (-), multiply (*), divide (/) and exponentiate (^). This will occur whenever the first character following a legal numeric operand is not one of the allowed symbols.

Unclosed left parentheses. Hit ESC key:

The right parentheses is missing, and must be inserted or the left parentheses removed. This should be easy to spot and correct by rechecking the parentheses. If there are more right parentheses than left, the illegal operator message will be displayed.

__ Illegal character. Hit ESC key:

Most of the responses to the field definition queries are limited to certain characters; not all characters are acccepted in the query responses. In many cases, the range of valid responses is given in parentheses after the query; otherwise, the set of legal characters is given in the section describing the particular attribute and query sequence.

FormGen exit error messages:

The following error messages may be displayed on the screen after ^C has been entered. All are fatal; the form may not be used for data entry until the errors are corrected. Explanations of the various error messages will be given following the list.

[form definition filename] ERRORS AND INCOMPLETE FIELDS:

Field Error description (position cursor at field to perform corrections)

??? No key field has been established.
(Choose at least 1 field as sort field and use ^K to assign key status.)

000 Illegal character(s) in the content control word.
(^R, RETURN key until the CCW is reached, correct the CCW.)

000 No file name specified for verify list.
(^R, RETURN key until "Enter list file name:" is reached, enter name.)

000 Field must be defined as "list verify" since field 000 references it.
(^R, RETURN key until "Verify field?" is reached, enter Y,L.)

000 Unspecified index for a list derived field.
(^R, RETURN key until "Index Field number:" is reached, enter index field or type ^J for additional help.)

000 Incomplete expression for calculated field.
(^R, RETURN key until expression entry is reached, then complete expression entry, substituting valid field numbers for all "?".)

000 Invalid range check limits.
(^R, RETURN key until range check limits are reached, then edit range limits so that the minimum value is less than the maximum.)

??? Key length is greater than 120 characters.
(Position cursor at key field(s), use ^K to remove "*".)

000 Unspecified Verify/Calculation order.
(^R, RETURN key until order is reached, then enter order.)

Enter exit command:
A = Abort without saving form **B = save form & Boot operating system**
C = save form and Continue **L = List errors on LST: device**
SPACE = continue without saving form
(A/B/C/L/SPACE):

??? No key field has been established.

This message will appear if the form has no key field. To correct for this condition, position the cursor in the chosen field and type ^K.

000 Illegal character(s) in the content control word.

The content control word is assigned during the edit mask queries (see section 2.16). This condition can arise in the following sequence of events: a constant was specified in response to the entry control query; an otherwise illegal character was specified to be used as a constant in the content control query step; the constant is removed from the ECW AND the constant character is not removed. To correct this condition, position the cursor in the field, type ^R and proceed through the query sequence, either removing the constant or replacing the quotation marks.

000 No file name specified for verify list

This message will appear when a datafield has been assigned the list verify attribute, and the file containing the list of legal values was not named in the definition process. As a consequence, DataStar will not be able to verify the values entered into the field. To correct, position the cursor in the field, type ^R and proceed through the query sequence and enter the file name. See section 2.12 for more information.

000 Field must be defined as "list verify" since field #xxx references it.

This condition will occur when a datafield is defined as file derived, and the associated index field has NOT been assigned the list verify attribute. See section 2.6 for more information about the requirements of file derived datafields.

000 Unspecified index for a list derived field.

Every file derived datafield must be associated with an index field. The purpose of the index field is to name the file to be used in deriving the values for the file derived datafield. If there is no index field, DataStar will not be able to derive any values for the datafield. See section 2.6 for more information.

000 Incomplete expression for calculated field.

This message will be displayed if the calculation expression was never entered or if a datafield referenced by the expression was subsequently deleted. To correct, either enter the missing expression or replace any "???" in the expression with a valid field reference.

000 Invalid range check limits.

If the maximum value allowed is smaller than the minimum allowed value, this message will be displayed. To correct, position the cursor in the field, type ^R and proceed to the range check query. Re-enter the minimum and/or the maximum allowed values.

??? Key length is greater than 120 characters

The maximum key length is 120 characters. This condition will occur when too many fields in the form have been designated as key fields or the designated key fields are too long. To correct this condition, use ^K to remove the key field designations or reduce one or more key field lengths.

000 Unspecified Verify/Calculate Order

This condition may occur in one of two ways:

A Y answer was given for the "Verify?" query, and the designer did not go far enough through the query sequence to specify a verify/calculate order; or there are more than 255 verifications or calculations to be performed.

If no order was specified, type ^R and RETURN until the Verify/Calculate Order query is reached and enter a proper response. If there are more than 255 verifications or calculations to be performed, reduce the total number to less than 255.

Chapter 3
Data Entry

This chapter describes the data entry process for the operator. It is advisable, though not required, to read the preceding chapters to get a general understanding of how a form is designed and created before attempting to use the data entry program.

The various functions of the data entry program are defined for the user together with procedures for operation. Data retrieval and updating are described in Chapter 4, and a complete description of the data verification process is given in Chapter 5.

This Chapter, together with Chapters 4 and 5, covers everything that the data entry operator will need to know in order to effectively operate the data entry program.

3.1 The Data Entry Process

The data entry process begins by invoking DataStar and entering the name of a previously created form definition file. DataStar reads the form definition file and displays the form on the CRT screen.

The form, created and defined through FormGen, appears much like common government or business forms with labels and lines or boxes to be filled out by the user. However, in this case the form appears on the CRT screen and the keyboard is used instead of a pencil to enter the requested information items. The information, or data, that is entered in the appropriate spaces on the form will be recorded in a data file on the disk.

When the form is displayed on the screen, the operator will quickly notice that the cursor can only be moved within the boxes, or datafields, and can't be moved anywhere else on the screen. The text that appears next to and around the datafields is called background text, and its purpose is to indicate what specific data should be entered into each field. There may be datafields in the form where the cursor will not be allowed to go; these are called derived or calculated datafields, and will be explained later. (See section 3.8.)

It is important to note that the characters typed into the datafields are the only data that will be stored in the datafile. The background text appearing on the screen is not entered into the datafile. For example: if the form shown below were to be used for data entry, the operator would only be able to move the cursor within the dotted lines.

```
Last Name:.............  First Name: .............
```

If the operator typed "Jones" and "Jack" into the fields, the screen would appear as below:

```
Last Name: Jones.....  First Name: Jack.........
```

The datafile for this form would look like this:

```
Jones,Jack                                                    <
```

after the data was entered into the file. All of the information about "Jack Jones" together is called a *record*. The above example is very simple, in actual practice there would usually be many more information items to be entered about Jack Jones, such as address, telephone number, etc. The datafile record would probably look more as shown below:

```
Jones,Jack,209 Carob Lane,Alameda,California,94501,4155551212          <
```

The end result of the data entry process is records in a datafile, entered by an operator according to the requirements of a form displayed on the CRT screen.

3.2 Invoking DataStar

To begin the data entry process, turn your system on and load the operating system by inserting the system disc and pressing the RESET or "boot" key. When the system prompt (A> or B>) appears on the screen, type one of the following:

DATASTAR

or

DATASTAR filename

and press RETURN. "filename" represents the name of the form definition file where the specifications for the form to be used for data entry are stored. If no file name is entered, DataStar will prompt you to enter the name after the signon message is displayed. If a file name is entered, DataStar will briefly display only the signon message (shown below) before proceeding.

Signon message:

```
MicroPro DataStar  release #.##  serial # XXXXXX
COPYRIGHT (C) 1980  MicroPro International Corporation

   (your terminal name will appear here)
```

If no file name was entered, the following prompt will also be displayed:

```
Enter name of form definition file (or press RETURN)
```

Press RETURN to display the following help message and prompt:

```
The form definition file is where the specifications for your form are
stored.  If you have not yet designed a form, choose a name for the form
definition file and enter it here.

Press RETURN key to enter form name or enter ^C to exit:
```

If ^C is entered, DataStar will exit to the operating system. If RETURN is pressed again, the "Enter name of form definition file (or press RETURN)" prompt will be displayed, allowing you to enter the form name.

Enter 1 to 8 letters and/or numbers (optionally preceded by disk drive letter and a colon). You may want to change disks first, in case the form is on a different disk than the program.

If the named form definition file is not found on the indicated (or current, if no drive was specified) drive, the following message and prompt will appear:

```
The form definition file, (name), does not exist yet.
Do you want to create it (A) or enter a different form name (B)?

Enter A or B:
```

When the name of the form definition file is entered, DataStar checks either the specified or the current disk to find the form. If the form is not found, the user is offered the option of transferring control to the FormGen program to create the form definition file, or re-entering the name.

Enter A to invoke FormGen, or B to enter a different form name. This prompt will appear if, for example, you typed the desired name incorrectly.

Once a form name has been entered and accepted, the next step is opening the associated data and index files. If the form has not been previously used for data entry, or the datafile and index file are not present on the current disk, the following prompts will be displayed. When a response is entered for the first prompt, the second prompt is automatically displayed.

Enter disk drive to use for the data file (filename.DTA) (A/B...)
Enter disk drive to use for the index file (filename.NDX) (A/B...)

If the data and/or index files are not found on the current disk, the user is prompted to either enter the disk to use for creating the files, or to enter the disk drive where the files already exist. Enter a letter (A-P) representing the disk drive. It is not necessary to press RETURN. For example, if the form definition file SAMPLE was located on the current disk (A), and the data and index files SAMPLE.DTA and SAMPLE.NDX were located on drive B, DataStar would offer the option of either creating new files on disk A (SAMPLE.DTA, SAMPLE.NDX) or using the existing files on drive B. However, if the above files had already existed on drive A, DataStar would automatically access the files. If other files are referenced and are not present on the current disk, the user will be prompted to enter the appropriate drive letter.

If a character is typed before one of these messages appears, the message will not be displayed. This allows experienced users to start typing before the messages are presented. Inexperienced users, however, may find themselves with a blank screen and no text. If this condition should occur, just press the RETURN key, and the form name message will be repeated.

3.3 DataStar menu displays.

There are two main displays, or menus to be used in the DataStar program: the Mode Selector Screen and the Mode Operation Menu. The Mode Selector Screen allows the user to select the desired mode of operation from a range of choices. When a form name is entered that has not been previously used for data entry, or for which there are no data records present on the current disk, the Mode Selector Screen is bypassed and the Add Mode is automatically selected.

The purpose of the Mode Operation Menu is to display the operating commands that are available in each mode, such as cursor motion, character insertion or deletion, etc. There are minor variations in the appearance of the Mode Operation Menu from mode to mode; some modes of operation do not require the presence of the full range of commands.

The Mode Selector Screen:

```
ADD MODE                    current form = form name
Enter character to select new mode:

A = Add new records         K = select records by KEY     E = Exit current form
I = SCAN in Index order     D = SCAN in Data file order   M = edit scan Mask
B = select Batch file       V = Verify batch file
F = File maintenance        J = Help                      SPACE = current mode
```

(Your data entry form appears immediately below the Mode Selector Screen.)

ADD mode:	see Section 3.8
K (Select by KEY):	see Chapter 4.
E (Exit current form):	see Section 3.7
I (SCAN Index order):	see Chapter 4.
D (SCAN Data file order):	see Chapter 4.
M (edit SCAN Mask):	see Chapter 4.
B (select Batch file):	see Chapter 5.
V (Verify batch file):	see Chapter 5.
F (File maintenance):	see Section 3.16
J (Help):	see below.

Typing ^J at the Mode Selector Screen will display a menu which prompts the operator to type the appropriate command for the mode to be explained. After the explanation is displayed, typing any other key will redisplay the mode selector screen.

The Mode Operation Menu:

The Mode Operation Menu will be described as it applies for each mode in the subsection descriptions of the various modes.

Typing ^J at any Mode Operation Menu will cause DataStar to remove the menu (except for the status line). The menu may be restored by typing another ^J.

3.4 The Status Line

The top line of both the Mode Selector Screen and the Mode Operation Menu is called the status line. The status line lists the current mode of operation together with the name of the batch file, if one is selected, and the name of the form being used. For example:

```
ADD MODE    /BATCH = filename    current form = NAME
CURSOR:     ^A = prev field      ^S = left char      ^D = right char      ^F = next field
            ^T = first field     ^L = last field
FIELD EDIT: ^G = delete char     ^V = insert hole    ^C = copy from previous record
OTHER:      ^Z = restore screen  ^U = print form     ^O = print data      ^J = help on/off
END/EXIT:   ^B = end entry       ^E = exit current mode
-------------------------------------------------------------------------------------------
```

The form appears in this area.

Unless a batch file has been selected for use, the name of the current datafile and the "/BATCH = " message will not be displayed. (See Chapter 5 on batch files.) The status line remains even if the rest of the Mode Operation Menu is removed by typing ^J for help on/off.

If ^E is entered to change modes, the status line will not be removed; the Mode Selector Screen will be displayed beneath the status line.

3.5 Window positioning

DataStar uses the CRT screen as a moveable window to display form layouts that are larger than the size of the screen. As datafields are filled or skipped through use of cursor motion commands, the screen display is moved automatically as required by the movement of the cursor.

Normal data entry will sometimes not require use of the cursor motion commands as DataStar automatically moves the cursor to the next datafield after the last position of a datafield is filled.

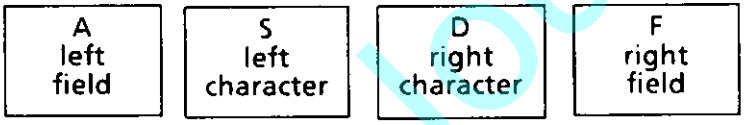
The experienced operator may use more of the screen area for displaying the data entry form by typing ^J to turn off the Mode Operation Menu. The status line will remain while the rest of the screen is used for

displaying the form. When the Mode Operation Menu is displayed, the area available for form display is 79 columns wide by 17 lines long (depending upon your terminal type). Turning the Menu off adds 6 lines to the form display area.

3.6 Cursor Motion

Cursor motion commands are used to place the cursor at a desired point on the screen to edit existing or enter new characters within a datafield. There are two types of cursor motion commands; moving the cursor field by field or character by character.

Note that the basic cursor motion keys ^A, ^S, ^D, and ^F, are arranged in a line on your keyboard. The position of the keys corresponds to the direction of cursor motion.



All of the cursor motion commands may skip over entire fields or certain positions within a field. These positions or datafields have been set up by the form designer. At the appropriate time, the positions will be filled automatically by DataStar. In the table below, references to datafields or positions within datafields should be understood as the first position where data entry can occur.

Table 3-1: Cursor Motion Commands

Command	Function	Description
^A	cursor left field	Moves the cursor to the beginning of the field, or to the first position of the previous field if it is already at the start of the field. If the cursor is at the start of the first field, the command will be ignored.
^S, ^H	cursor left column	Moves the cursor one column to the left. If the cursor is in the first column of a field, ^S will be ignored.
^D	cursor right column	Moves cursor one column to the right. If the cursor is in the last column of a field, ^D will move the cursor column to the first column of the next datafield.
^F, ^I	cursor right field	Moves cursor to the next embedded "hole", data end or start of next datafield –whichever is encountered field first. A hole is an empty character position with data on either side; data end is an empty character position with data only to the left of the position.
RETURN	cursor right field	Moves cursor to start of next datafield.
^T	cursor 1st field	Moves the cursor to the beginning field of the form.
^L	cursor last field	Moves the cursor to the last operator entered field of the form.

If any of the forward cursor commands (^D, ^F, ^I, RETURN) are used to leave the last datafield of the form, an end-entry command (see section 3.14, end entry) is assumed and end-of-record processing is automatically begun. If this condition occurs by mistake, enter ^T to continue entering data into the form.

3.7 ADD MODE

DataStar's ADD MODE allows the user to enter data records into a datafile according to the specifications of the data entry form created by FormGen. It is important to note that the background text of the form is NOT copied into the datafile during data entry. Only the characters entered into the datafields are stored in the datafile.

ADD MODE may be selected either by the user, or by DataStar under special conditions. The user may select ADD MODE by typing an A when the Mode Selector Screen is displayed, or by pressing the SPACE bar when

the current mode listed in the status line of the Mode Selector Screen is ADD. DataStar will automatically select ADD MODE whenever there are no existing records in the datafile associated with the current form. DataStar will also select ADD MODE automatically when a new batch file is selected by the user that does not contain any records. (See Chapter 5, Batch Processing.)

In addition, some forms may require that a batch file be selected for entering records. In this case, it will be impossible to enter ADD MODE without first specifying a batch file. If so, a disk drive and an 8 letter (or less) batch file name must be entered.

When ADD MODE is selected by either the user or by DataStar, the Mode Operation Menu (below) is displayed:

```
ADD MODE    /BATCH = filename    current form = NAME
CURSOR:     ^A = prev field     ^S = left char     ^D = right char     ^F = next field
            ^T = first field    ^L = last field
FIELD EDIT: ^G = delete char    ^V = insert hole   ^C = copy from previous record
OTHER:      ^Z = restore screen ^U = print form    ^O = print data     ^J = help on/off
END/EXIT:   ^B = end entry      ^E = exit current mode
-------------------------------------------------------------------------------

Order #: 0000000      Date (M/D/Y): 00/__/8      Customer #: ______

 Bill to:  ----------------------      Ship to:  ----------------------
 Address:  ----------------------      Address:  ----------------------
 City:     ----------------------      City:     ----------------------
 State:    --  Zipcode: -----         State:    --  Zipcode: -----

P. O. #: ----------  Ship via: ----------      Terms: ----------

Quantity  Product  Description                Unit Cost  Total Cost

---       -----    ---------------------------  $-------   $-------
---       -----    ---------------------------  $-------   $-------

   Tax rate: ---%  Sales Tax: --------          Total: $----------
```

The form shown below the Mode Operation Menu display (ORDER) is the example form provided on the DataStar Distribution diskette. This form will be used to illustrate the description of the various DataStar modes. Note that the form layout will be different if a different form name is used, but that the Mode Operation Menu will remain basically the same for any form used in ADD MODE.

If you invoke DataStar, and specify the ORDER form, the above display will appear on your CRT screen. We recommend using this form to familiarize yourself with using the various operating modes and commands.

Instructions for invoking DataStar are given in Section 3.2, and the cursor motion commands are described fully in Section 3.6. Please refer to section 2.17, Listing the Form, for a listing of the ORDER form giving the field numbers. The discussion on the next pages will refer to fields by their numbers.

As you enter characters and use the cursor motion commands, you will quickly discover several things about the ADD MODE and about the ORDER form. First, the cursor can only be moved within the datafields appearing on the screen, and secondly, that the cursor cannot be moved into all of the displayed datafields. In addition, the cursor is positioned differently in the various fields as you move through the form using ^A and ^F. Some fields may already have data in them even though you have not entered any characters, and you will discover that some characters are acceptable for entry into some datafields and not in others. These differences are described in the next sections.

3.8 Datafield Entry Types

There are three types of datafields as far as the data entry operator is concerned: entered, derived, and combined entered/derived.

Entered fields may be freely entered and the data within modified by the operator. The cursor may be positioned as desired within an entered field. This is the most common type of datafield. The second field of the example (ORDER) form is an entered field.

A derived field is filled by DataStar according to instructions provided by the form designer. The data for the field may be derived by calculations based on the values of other fields in the form, or by reference to other datafiles, copied from previous records or by operations involving constant values, or any combination of such methods. Regardless of the method used to derive the value of the field, the cursor cannot be moved into the field in ADD MODE. Fields 6 through 10 of the ORDER example are all derived fields.

The third type of field is a combination of the first two. The values are normally entered by DataStar, but the operator also may enter or modify data within this type of field. In normal data entry, the cursor will bypass combination fields; the operator may move the cursor within the field by first bypassing the field and then backing up by using ^A. A further characteristic of combination datafields is that once a value has been entered by the operator, the derived portion of the field's definition

instructions is lost for the current record. Another type of combination entered/derived field is one where the operator enters the value for the field for the first record of the file and DataStar enters all the rest. The first field of the ORDER form is an example of both types of entered/derived combinations. The operator must enter the first order number in the file; thereafter, DataStar will supply the order number. In addition, the operator is free to move the cursor into the order number field whenever a different order number is required.

3.9 Datafield Entry Characteristics

Datafields behave differently according to which entry characteristic is assigned to them: left-justified, right-justified, or decimal aligned.

In a left justified field, the cursor is positioned in the leftmost character position when first moved into the field. As data is entered, the cursor moves one column to the right after each character is entered. Alphabetic fields are most commonly left justified. Field #3 of the ORDER example is left justified.

In a right justified datafield, the cursor is positioned one column to the right of the rightmost character position when the field is entered. During data entry, the cursor does not move. As characters are typed, they appear to the left of the cursor position and are "pushed" to the left as more characters are entered. Numeric fields are often right justified. Fields #1, #2, and #5 of the ORDER example are all in this category.

Fields to be used for monetary amounts, or other numbers involving decimal points are commonly decimal aligned. The cursor is positioned at the decimal place and does not move until a period is typed. Characters appear to the left of the decimal place and are pushed left as more characters are typed. When a period is typed, the cursor is moved one place to the right and continues to move normally as the last characters of the field are entered.

3.10 Edit Characters

There are a number of different types of characters, called edit characters, that may be present when the form is first presented for data entry, or may appear only after data is entered into the various fields. Edit characters include pad characters, floating characters, constant characters and auto-duplicated characters.

A pad character is a single character that is used to fill un-entered positions on one end or another of a field. A field may be padded with any character that the form designer chooses. Commonly used pad characters are zeroes and spaces. The order number field in the ORDER form is padded with zeroes.

A floating character is a character on one end of the field or the other that moves as data is entered. Any character can be used as a floating character by the form designer. A dollar sign ($) is used as a floating character in fields #22, #23, #27, #28 and #31 of the ORDER example form.

Constant characters are any other characters that may be present within a datafield, or that may appear as data is typed into the form. A constant character may be any character appearing any place within the field; such characters are set up by the form designer. Operator entry is usually not permitted in positions occupied by constants. The cursor will automatically skip over such positions as data is entered. In some cases, however, it will be possible to move the cursor back over a constant position and change the character. An example of a constant character within a field is the "8" in the first character position of field #4 of the ORDER form.

The last type of edit character, the auto-duplicated character, is different than the other types in that it requires operator participation in order to work. When the auto-dup character is present in a form, the operator must set the value of the character when entering the first record in a session. Thereafter, the character will be automatically copied from record to record as data is entered. As is the case with the constant characters, the cursor will skip over positions containing auto-dup characters once such characters have been entered. It will usually not be possible to move the cursor back over these positions. In order to change such characters, it is necessary to exit from ADD MODE (see section 4.6 on entering auto-dup characters with the scan mask). In some forms, the auto-duplicated character will be defined so that it is possible to change it without leaving ADD MODE. In that case, the ^S command must be used to put the cursor at the auto-duplicated position. An example of an auto-duplicated character is the second character position of field #4 in the ORDER form.

3.11 Field Content Control

One last form of control may be exercised as data is entered; the actual content of the entered data may be monitored on a character by character basis. Field content control may or may not be present in a given form. It restricts the characters that may be entered at a given position within a field to a specified set. If a character is entered that does not belong to the approved set, a message will be displayed at the top of the screen that gives the allowed characters. The usual use of content control is to restrict entry to numbers-only for numeric fields, or letters-only for alphabetic fields. Some character positions in alphabetic fields may be automatically capitalized; others may restrict entry to lower case letters only. In the ORDER form, every entered datafield except the second address field is content-controlled.

3.12 Field Edit Commands

The ^G, ^V and RUB keys are used to delete characters and insert characters and/or spaces within datafields. The cursor positioning will vary between left and right justified and/or decimal aligned datafields when characters are inserted and/or deleted. For character insertion, as well as data entry, a decimal aligned field acts exactly like a right or left justified field, depending upon which side of the decimal point the cursor is positioned. When the cursor is to the left, the field acts right-justified; when the cursor is to the right, the field acts left-justified. When the cursor is at the decimal point, the field functions as if right justified with the cursor at the starting position. In the following discussion, decimal aligned fields should be included with right or left justified fields, depending upon the position of the cursor.

^G = delete character

The ^G command is used to delete individual characters from a datafield. When ^G is entered, the character at the cursor position is deleted and the other characters are automatically moved to fill in the place of the deleted character. In a right justified field, characters are pulled from the left to fill in the place of the deleted character. In a left justified field, characters are pulled from the right. If the cursor is positioned at the decimal point in a decimal aligned field (or at the start position of a right justified field) there is no character deletion.

^V = insert hole

The ^V command inserts an empty space, or hole, at the cursor position within a datafield by moving any other characters present in the field to the left or right depending upon the type of field justification in effect for that particular field. In a left justified field, characters are pushed to the right to make room for the hole. In a right justified field, characters are pushed to the left. If the cursor is at the decimal point, or at the starting position in a right justified field, no hole is inserted.

It is important to note that if ^V is used to insert holes into a datafield, those holes must be filled by characters or an error message will be displayed when the operator attempts to write the record to the datafile. ^V inserts holes, and holes are not counted as characters.

RUB

The RUB key (labeled RUBOUT or DELETE on some terminals) functions so as to erase the previously entered character; if the wrong key is accidentally pressed, RUB will remove the character and restore the field to its previous condition. If the last character entered was a control character, or if RUB is typed in the middle of some previously entered data, the effect will vary depending upon the justification of the field. In a left justified field, the character immediately to the left of the cursor is deleted, and the cursor is moved one column left to fill the hole together with the character under the cursor and all characters to the right of the cursor. If there is no character position to the left of the cursor position, RUB will function like ^G and delete the character under the cursor. If the cursor is positioned to the right of the decimal in a decimal aligned field, RUB will be ignored since the decimal point cannot be deleted.

In a right justified field, the cursor does not move with the RUB key. In this case, the character immediately to the left of the cursor is deleted and the other characters to the left are shifted one column right to fill the hole. If the cursor is in the first position of the field, the character under the cursor will be deleted and the cursor moved one column to the right.

^C = Copy from previous field

The last field edit command, ^C, may be used to copy the field value of the previous record into the current record. This command is used to avoid having to retype values that are identical from record to record.

The previous record is always the last record displayed on the screen, not necessarily the last record entered. If the screen was clear prior to typing ^C, and no previous record existed, ^C will clear the field.

If ^C is entered in error, and the resulting value is not correct, the characters may be deleted by use of ^G or RUB and the correct value entered afterwards, or the correct value may simply be typed over the incorrect characters.

3.13 Other Commands

^U = Print Form, ^O = Print Data

These commands both cause data currently on the screen to be printed on the line printer. The line printer must be set as the CP/M LST: device in order to use the ^U or the ^O commands. If the line printer is not set correctly, the entire system will hang, or stop operating when either ^U or ^O is entered.

The ^U command prints the form along with the data; the ^O command prints only the data.

With the ^O command, the background text of the form and/or the lines of the datafields are NOT printed; only the values appearing on the screen in the datafields will be printed. The ^O command is useful for filling out pre-printed forms.

Typing any character while printing is in progress will abort the printout in both cases (^U, ^O). Whether or not it is aborted by typing a character, printing is terminated automatically by a form feed. If pre-printed forms are used, the LST: device driver or the hardware should be set up to translate the form feed character to a skip n lines command (where n is the appropriate number of lines to get to the top of the next form).

^Z = Restore screen

The ^Z command is used to "reset" the form display on the CRT screen whenever the display becomes jumbled or garbaged with extra or unwanted characters. This can occur, for instance, due to static electricity or messages from the operating system. ^Z command causes DataStar to read the form specifications from memory and restore the form display on the screen.

The form specifications are stored in the system memory, and the form definition file is not used when ^Z is typed to restore the screen display.

^Z has no effect upon the characters that have been typed into the various datafields of the form.

^J = Help ON/OFF

The ^J command can be used by experienced operators to increase the area of display for forms that are longer than 17 lines. (The actual number may differ on some terminals.) If the form is 17 lines or less, there is no advantage to using ^J. Typing ^J removes the help display from the screen; thereby adding an extra six lines to the area available for screen display. Typing a second ^J will restore the help display.

3.14 End of Entry

When all of the required datafields (if any) have been filled, and all other desired fields are complete, the end-of-record processing is begun. This can occur in one of several ways:

- entering ^B at any position on the form, or
- filling the last position of the last datafield on the form, or
- using a control character (^D, ^F, ^I, RETURN) to leave the last field of the form.

Regardless of how it is initiated, end-of-record processing consists of three steps:

1. entry, range, and check digit tests are performed;

2. verifications and/or calculations are completed;

3. operator approval is given.

If these three steps are completed successfully, the data on the form is recorded in the file on the disk as a single record. If the steps are not completed, ADD MODE is resumed and the operator may continue entering and/or correcting data into the form. In some forms, a test for duplicate keys may occur (see section 4.3 on key fields) before the record is written to the file. In such forms, an error message will be displayed and ADD MODE resumed if a record with the same key already exists.

Entry, range and check digit tests:

The first step at end-of-record processing is a check that all required entry positions have been filled, and that there is no field with data entered that is beyond its allowed range of values, and that certain numeric fields with redundant digits have been correctly filled. Unless a mistake has been made, this step is totally invisible to the operator; if the step seems to have been omitted, it is because the entered data passed its first test. If a mistake is discovered, a message with:

"Field is not complete..."

"Field value must lie between..." or

"Illegal field..."

may be displayed at the top of the screen.

The "Field is not complete..." message is displayed whenever a required-entryposition is empty or a field contains a hole. Certain datafield positions in the form may be designated as required; the record cannot be completed until such positions are filled properly. In the ORDER example, fields 2-5 are required entry fields. A hole is an empty space in a string of entered data. In addition, unless the field is entirely empty, a left justified field must start with a data character; a right justified field must end with a data character.

The "Field value must lie between..." message is presented whenever the data in a field lies outside the range of values given by the form designer. Not all fields will have range checks associated with them. When the error message is displayed, the acceptable range for values is given along with the message. In the example form (ORDER), fields 2 and 3 are range checked: #2, representing the month, must have a value between 01 and 12; field #3, representing the day of the month, must have a value between 01 and 31.

The "Illegal field" message is presented whenever a field with an extra check digit has been illegally entered. A check digit is an extra digit in a numeric field whose only purpose is to make the data in the field obey some rule. In fields that are NOT required-entry, the check digit requirement may be avoided by clearing the field; if the field IS required-entry, there is no way to avoid the check digit requirement other than entering an acceptable value.

For all three error messages, the operator must press the ESC key in order to clear the error. When the ESC key is pressed, the cursor will be positioned at the character position within the field that is in error. In the case of an entry check, the cursor will be placed at a hole or unfilled required-entry position. For range check errors, the cursor will be placed at the first position in the field that doesn't satisfy the range requirements. For check digit errors, the cursor will be positioned at the field start position.

Verification and Calculation:

When the form contains no entry, range check or check digit errors, Verify Mode is automatically entered unless batch processing has been selected. (See Chapter 5, Batch Processing.) If a batch file was specified when ADD MODE was entered, the verification and/or calculation step will be omitted from the end of record processing. This step will also be omitted if there are no fields to be verified or calculated on the form. Otherwise, if any fields are to be verified, a Verify Mode message will replace the ADD MODE message in the status line, and the cursor will be positioned in the first field in the form requiring verification.

There are two ways in which an operator will be asked to verify a field: by sight or by retyping the field. If by sight, the cursor will be moved into the field so the operator can look at the entered data one more time to make sure it is correct. After checking the field, the operator must press RETURN or TAB to indicate that the field has been checked. If verification is done by retyping the field, the entered data will be cleared and the operator

will be required to retype the data exactly as it was typed before. If the two entries don't match, the field will again be cleared and the operator asked to enter the data two more times. In the ORDER example, field #16 is retype verified, fields #19 and #24 are sight verified.

Calculations and/or list verified errors are also included in verify mode. At different points in the verification process, field calculations may be suddenly displayed. The ORDER example has 5 calculated fields, #1, 23, 28, 30 and 31. In addition to calculation results, any list verification errors that occur will be brought to the operator's attention at this time. A list verify error may occur where a field's data is only accepted after it is compared against a list of valid entries. The ORDER form contains five such fields; field numbers 6, 9, 14, 20 and 25. If a list verify error occurs, the following message will appear:

"Illegal field. Please re-enter. Hit ESC key."

After pressing the ESC key, the operator will be required to enter an acceptable value into the field.

Operator Approval:

When the verification process, if any, is complete, the following prompt will be displayed:

```
ADD MODE                    current form = EXAMPLE

   Hit RETURN to file entered data or RUB to erase all items on screen.
   Type any other character to return to top of form and continue:

-------------------------------------------------------------------------------
```

If RETURN is pressed, the completed record is written into the datafile and the cursor is positioned at the first position of the first field to begin entering the next record (unless the record contains a key that duplicates one already in the file and the file specifications require unique keys). RUB causes all of the typed data on the screen to be deleted, and the cursor is positioned at the first field to allow re-entry. Any other character or the duplicate key error will cause the cursor to be positioned at the first field to re-enter any incorrect data. If any data in the form is changed, when end of record processing is again entered, any calculated fields will be re-displayed; any verified fields that were altered will also have to be re-verified.

3.15 Changing modes, forms; exiting DataStar

The ^E command allows the user to change from one mode of operation to another, or to change forms, or exit from DataStar.

Changing Modes:

Enter ^E at any point in the form. It is important to note that if you enter ^E before writing the current record to the disk, the values entered into the datafields for that record will be lost. Be sure to complete the entry before entering ^E by first typing ^B and performing any end of record corrections or verifications. Typing ^E will cause the Mode Selector Screen to be displayed. Type the letter of the desired mode, or press the SPACE bar to return to the last mode (listed in the upper right corner of the Status Line). Chapters 4 and 5 document the use of these modes; section 3.3 shows the mode selector screen and lists the various modes.

Changing Forms:

To change forms, type ^E from the current Mode to display the Mode Selector Screen. Type a second ^E after the Selector Screen appears to display the following prompt:

Enter name of form definition file (or press RETURN):

Enter the name of the new form and press RETURN. If the datafile and index file for the new form are present on the current drive, and there are records in the datafile, the Mode Selector Screen will be redisplayed for the new form. If the data and index files are not present on the current drive, you will be prompted to enter a disk drive label to use for the files. If there are no records in the datafile, DataStar will automatically select ADD MODE. Before entering a form name and optional disk drive, the operator may change disks if necessary.

Exiting DataStar:

To exit from DataStar, follow the above procedure for changing forms, but instead of entering a form name, type ^C. The three control characters (^E,^E, and ^C) may be typed rapidly to avoid waiting for the screen display.

3.16 File Maintenance

It is recommended that periodic maintenance be performed on DataStar data and index files by using the ^EF command. There are three reasons for this recommendation:

Efficiency:

When a new record is added to a DataStar file, unless the file is being entered in sort order, the new record's position in the file is determined and record pointer chains are constructed for that record. As the file grows, if no maintenance is performed, the proportion of chained out-of-position records to chained in-position records grows too. This condition will result in slower operating speeds. On the average, when a file has doubled in size since it was last maintained, DataStar can be speeded up considerably by performing file maintenance. The factor of the increase in speed in this case will be the ratio of chained to unchained records.

Obviously, if a file was entered in random order and has never been sorted, both record entry and retrieval speeds can be dramatically improved by using the ^EF command. DataStar was not designed to operate on large datafiles without periodic maintenance.

Disk Space:

When a record is deleted, or updated in such a way that a new record must be created, the record is marked as deleted by writing a delete indicator (the byte OFFH) in the first byte of the record. It is important to note that the deleted record is not erased and the disk space is not reclaimed for re-use. In order to reclaim the "dead" space, the file maintenance step must be performed.

File/Disk Integrity:

If the file is not maintained properly at periodic intervals, the amount of disk accesses needed to locate records is increased. This condition results in greater wear on the surface of the disk and will reduce the overall life expectancy of the disk. Regular performance

of the file maintenance step will decrease the chance of losing data due to a disk failure.

To perform the file maintance step, type F at the Mode Selector Screen. The batch file selector screen is displayed since DataStar uses a separate file for work space. (See Section 5.2 on batch file selection.) Enter the disk drive and the name of the file to use for temporary work space. The named file should not already exist on the disk. DataStar will display the following message when the maintenance operation is complete:

File maintenance is complete. Hit ESC key.

If the disk full message appears instead, the work file may be deleted without losing any data. Delete enough files on the disk to create a space as large as the data and index files to be maintained, and then retry the operation.

Chapter 4
Data Retrieval and Modification

Before attempting to read this chapter, we strongly recommend that you first read Chapter 3 -Data Entry- in order to gain a basic understanding of the operation of DataStar.

This chapter describes the retrieval and modification procedures for previously entered data records, including scanning the data and/or index files, selecting individual records by keys and modification and/or deletion of records as desired.

Operations involving the selection and verification of batch files are described in Chapter 5.

4.1 The Data Retrieval Process

There are three basic methods of retrieving data records through DataStar; scanning either the data or the index files record-by-record, and/or selecting individual records by key. The two SCAN modes operate by accessing the specified data or index file, and then displaying records in sequence as they are encountered. The two files, data and index, are usually in different orders. Scanning the data file displays records in the order of entry/modification, whereas scanning the index file displays records in order by the key field(s). The operation of the scan modes may be altered by using a scan mask (see Section 4.6 Edit Scan Mask) so that only those records which fit the current scan mask will be displayed. The scan modes examine all of the records in either data or index file order.

The Select by Key mode provides rapid retrieval of individual records. When a key value is entered, DataStar searches the index file to find a matching value. If a value is found that exactly matches the specified key, DataStar then reads the associated record from the datafile and displays the entire record on the screen. The index file is organized for very rapid retrieval in the select by key mode. Since it is in key field order, it is not necessary to read the file sequentially in order to find the specified key; instead, DataStar skips through the file using a process similiar to locating a number in a telephone directory.

DataStar uses "pointers" during the retrieval process in much the same way as a reader uses book or place markers when reading; the pointer

marks the place in the file where DataStar stopped scanning. When a command is given to resume scanning, DataStar goes directly to the pointer and resumes scanning at the indicated place in the file.

Since the file pointer is set by any file access, the select by key mode can be used to set the file pointer for scanning. For example, if a file was keyed on a date field, all of the entries for a given date could be quickly examined by using the select by key mode to find that date, then switching to SCAN in INDEX to look at the rest of the records.

In other words, the retrieval process involves searching the data records to locate either a specified record (Select-by-Key mode) or any record that is encountered in sequence (Scan in Data file order or Scan in Index file order). Note that either scan mode can be made to display records selectively by using the scan mask.

The four modes described in this chapter (Select by Key, SCAN DATA, SCAN INDEX and EDIT SCAN MASK) are all entered by typing ^E to bring up the mode selector screen, followed by a letter indicating the mode to be selected (K, D, I or M). Mode selection is therefore a two-letter command; the letters may be typed quickly so as to avoid displaying the mode selector screen.

Either the primary file or a batch file may be current when one of these modes is selected. If the current file is empty, however, these modes are neither useful nor permitted. If an attempt is made to select one of these modes, the following message will be displayed:

File is empty so add mode is the only legal mode.
Hit ESC key.

4.2 Record Modification and/or Deletion

When data records are retrieved through either Scan mode or by Select mode, the user may modify or edit the record as desired, or delete the record entirely. The edited record may be deleted (retaining the original), or used to replace the original record. When the modified record is used to replace the original record, the modified record is written at the end of the data and the original record is marked as deleted. (The space occupied by the deleted record is NOT automatically available for re-use, so if a lot of records are modified or deleted, the file maintenance mode should be used to remove the unusable space in the file. See Section 3.16 File Maintenance.)

When a record is retrieved and displayed on the screen, the operator may use the editing commands displayed on the Mode Operation Menu to insert or delete characters in the datafields. After all the desired corrections have been made, type ^B to end entry. The following prompt will be displayed:

> **Hit RETURN to file updated data or RUB to leave filed data as it was and begin updating another form.**
> **Type any other character to return to top of form and continue:**

To delete a record without making any modifications, type ^B after the record has been displayed. The following prompt will appear on the screen:

> **Hit RETURN to go to next form or RUB to remove data from file.**
> **Type any other character to return to top of form and continue:**

Pressing RUB causes DataStar to mark the record as deleted. Subsequent scanning of the file will not display the deleted record; although the record technically still exists within the file, DataStar will not access nor display the data contained in the record. The deleted record will remain in the file until the file is rewritten through use of the File Maintenance mode. Note that once a retrieved record is edited, typing ^B will display the record update message instead of the record deletion message; the only way to enable record deletion at this point is to re-retrieve the record.

Verification

If the current form has verified datafields, and one or more of the verified datafields has been altered, the modified record must be verified before it is written into the file unless a batch file is selected. The verification phase proceeds according to the type of verification required (Sight/Retype) and in the order specified by the form designer. See Chapter 5 for more information about verification processing.

4.3 Select By KEY Mode

```
LOCATE KEY MODE, PLEASE ENTER KEY     current form = EXAMPLE
CURSOR:      ^A = prev field    ^S = left char     ^D = right char    ^F = next field
             ^T = first field   ^L = last field
FIELD EDIT:  ^G = delete char   ^V = insert hole   ^C = Copy from previous record
OTHER:       ^Z = restore screen ^U = print form   ^O = print data    ^J = help on/off
END/EXIT:    ^B = end entry     ^E = exit current mode
-------------------------------------------------------------------------------------------

Order #: 0000000     Date (M/D/Y): 00/../8.     Customer #: .......
```

Keys

When a form is designed through FormGen, certain datafields (at least one) are designated as "Key Fields". Key fields are treated differently than the other fields in the form. When values are entered into key fields, a copy of the data is stored in the index file together with a data pointer and a chain pointer. The data pointer indicates the location of the complete record in the data file, while the chain pointer is used to keep the file sorted in key field order even though it may be entered in another order. The purpose of the index file is to provide a means of rapidly accessing selected data records without having to sequentially read the entire datafile record by record. The Select by Key mode allows the user to enter the key for a specific record; DataStar then searches the index file for a matching value. If a matching value is encountered, the pointer is read and the entire record is displayed on the screen. If no matching value is found, the following message is displayed:

```
Key not found. Hit ESC key
```

Note that even though the record was not found, the file pointer has still been set. The file (on either side of the file pointer) may be examined by switching to SCAN INDEX mode, and using the ^N and/or ^P commands to scan forward and backwards. This is useful in cases where the exact wording or spelling of the key fields is not remembered or in cases where the key field is long and only a few characters are entered to save time.

In other cases, there may be more than one record with the same key values. DataStar will display the first matching record encountered. To examine other records having the same key, the operator may switch to SCAN INDEX mode and use the ^N and ^P commands to search forwards and/or backwards.

When Select By Key mode is used, the cursor may only be moved within key fields to enter the key. All of the editing rules described in chapter 3

apply here also. If the key fields on the form have embedded constants or are restricted to a certain set of characters during data entry, these conditions will also apply during key field entry. In contrast, if there is more than one field designated as a key, the order of entry in Select By Key mode may be different than the order used for original data entry. This is because key fields are entered in this mode in order of precedence of sort order. The primary field is entered first, followed by the secondary, etc.
The end entry command, ^B, is used to end the key field entry. DataStar will clear the

```
Please enter key
```

in the status line, and display the record if located. After the key is located, and the complete record is displayed, the operator may update the record as desired and write the new record into the datafile to replace the original record or delete the original as desired.

In the example form (ORDER) partially displayed above, the first field is the key field.

4.4 SCAN in Data file order

```
     SCAN MODE (D)                    current form = EXAMPLE
CURSOR:     ^A = prev field   ^S = left char    ^D = right char    ^F = next field
            ^T = first field  ^L = last field
FIELD EDIT: ^G = delete char  ^V = insert hole  ^C = Copy from previous record
OTHER:      ^Z = restore screen ^U = print form ^O = print data    ^J = help on/off
END/EXIT:   ^B = end entry    ^N = next record  ^P = prev record   ^E = exit mode
 ------------------------------------------------------------------------------

Order #: ------      Date (M/D/Y): __/__/__      Customer #: -------

     Bill to:  ----------------        Ship to:  ----------------
     Address:  ----------------        Address:  ----------------
     City:     ----------------        City:     ----------------
     State:    --  Zipcode: -----      State:    --  Zipcode: -----

P. O. #: ----------   Ship via: ----------      Terms: ----------

Quantity    Product    Description              Unit Cost    Total Cost
  ---         -----    -------------------------  -------      ----------
  ---         -----    -------------------------  -------      ----------

        Tax rate: ___%    Sales tax:  ---------   Total:  ----------
```

The SCAN in Data file order mode allows you to scan through the datafile in sequential order, displaying each record in the file as it is encountered. You may scan either forwards or backwards by typing successive ^P's or ^N's to display either the previous or the next record in the file. Each record may be modified or deleted as desired (See 4.2 Modification/Deletion).

The operator may change the scan mask so that only certain records will be displayed while scanning through the file (see section 4.6 Edit Scan Mask). If either the end of the file or the beginning of the file is encountered while scanning in either direction, the following message is displayed on the screen:

End of file.
Hit ESC key to continue scan.

After pressing the ESC key, the first or last record, depending upon the direction of scanning, in the file (that fits the scan mask) will be displayed on the screen. If no matching record is found, the following message will appear:

The file does not contain any records to match the scan mask.
Hit ESC key to continue in EDIT SCAN MASK mode.

After pressing the ESC key, the current scan mask will be displayed for editing. (Note that this message will also appear if all the records in the file have been deleted.) At this point, the scan mask may be edited or a new mode selected in order to continue.

If the line printer has been set up as the CP/M LST: device, ^U or ^O may be used to print the data in the file on the line printer. ^U prints the form as well as the data; ^O prints just the data. ^O is especially useful for filling out pre-printed forms. Unlike the select by key mode or ADD MODE, typing ^U or ^O in SCAN MODE (D) will cause the remainder of the file to be printed in datafile order, beginning at the current record on the screen. Printing stops when the end of the file is encountered, or a key is pressed on the terminal. Note that printing in scan mode is subject to the scan mask; only records that match the scan mask will be printed.

4.5 SCAN in Index file order

```
     SCAN MODE (I)                              current form = EXAMPLE
CURSOR:     ^A = prev field    ^S = left char     ^D = right char    ^F = next field
            ^T = first field   ^L = last field
FIELD EDIT: ^G = delete char   ^V = insert hole   ^C = Copy from previous record
OTHER:      ^Z = restore screen ^U = print form   ^O = print data    ^J = help on/off
END/EXIT:   ^B = end entry     ^N = next record   ^P = prev record   ^E = exit mode
 ------------------------------------------------------------------------------

Order #: _______          Date (M/D/Y): __/__/__          Customer #: _______

     Bill to:  ____________________          Ship to:  ____________________
     Address:  ____________________          Address:  ____________________
     City:     ____________________          City:     ____________________
     State:    __  Zipcode: _____            State:    __  Zipcode: _____

P. O. #: ___________     Ship via: ___________          Terms: ___________

Quantity    Product    Description              Unit Cost    Total Cost

  ---         -----    -------------------------  -------    ----------
  ---         -----    -------------------------  -------    ----------

        Tax rate: ___%     Sales tax:  _________   Total:  ___________
```

The SCAN MODE (I) functions exactly the same way as the Scan in Data file order mode, except that the Index file order instead of the Data file order is used.

The records are presented when scanning in sort order (by key field) instead of entry order. When printing the file with ^U or ^O, records will also be printed in sort order. If the printing did not begin at the start of the file, a different subset of the file may be printed depending on whether data or index scan modes were used.

4.6 Edit Scan Mask

The scan mask determines which records will be displayed by either of the two scan modes, and which records will not be displayed. Until the scan mask is edited, the scan modes will display all records in the files as they are encountered. If the scan mask is altered, or edited, then only those records that match the scan mask will be displayed; all others will be ignored during scanning. The initial state of the scan mask is displayed below:

```
     EDIT SCAN MASK                    current form = EXAMPLE
CURSOR:      ^A = prev field    ^S = left char     ^D = right char     ^F = next field
             ^T = first field   ^L = last field
FIELD EDIT:  ^G = delete char   ^V = insert hole   ^C = Copy from previous record
OTHER:       ^Z = restore screen ^U = print form   ^O = print data     ^J = help on/off
END/EXIT:    ^B = end entry     ^E = exit current mode
 ------------------------------------------------------------------------------

Order #:*******          Date (M/D/Y): **/**/8*       Customer #:*******

     Bill to:      ******************  Ship to:      ******************
     Address:      ******************  Address:      ******************
     City:         ******************  City:         ******************
     State:        **  Zipcode: *****  State:        **  Zipcode:*****

P. O. #:**********       Ship via: ************          Terms: **********

Quantity     Product     Description      Unit Cost      Total Cost

   ***        *****      ***************   *****         ********.**
   ***        *****      ***************   *****         ********.**

        Tax rate: *.*%        Sales tax:    ********Total:    ********.**
```

Note that all of the datafields are filled with asterisks. The asterisks indicate nonspecific values (i.e., "wild cards" that accept any values) that will match any record in either the datafile or the index file. Also note that any constants, such as decimal points and the "8" in field #4, are shown in the appropriate places within the fields. Conditional constants (requiring data to be on one or both sides of the constant) are only displayed when the associated condition is satisfied. An example of a conditional constant is a comma in the total cost fields.

To edit the scan mask so that only certain records will be displayed by the scan modes, use the cursor motion commands to position the cursor within the desired datafields, and enter the appropriate changes. Unlike add mode, entry will be allowed at all datafields whether or not they are derived. As far as possible, EDIT SCAN MASK mode operates so as to prevent creation of scan masks that are impossible to fill. Constant characters (conditional or otherwise) in the datafield cannot be replaced with other characters; no entry is permitted in these positions, and the shift commands, ^G and ^V, simply shift character strings around these positions.

In addition, all of the conditions and restrictions present in ADD MODE are also in force here.

It is important to note that DataStar compares fields from left to right, regardless of whether they are left or right justified or decimal aligned. This may create some confusion at first when using the scan mask in right justified and decimally aligned fields when all character positions are not entered.

For example: A form has a four character, right-justified field. A scan mask which had a 2 in the first position would appear as follows:

```
2***
```

This does not select all records that have the values 2000-2999 in that field. Instead, all records that have a 2 in the first character position are selected (e.g., 2, 20-29, 200-299, 2000-2999)

If the above example had a 2 in the last character position:

```
***2
```

then the scan mask would NOT display all records that end in a 2. Instead, only those records greater than 1000 that also ended in a 2 would be displayed. The confusion may be created because a right justified field is entered on the right side of the field, but for the purposes of comparison, is checked from the left side of the field.

Note: if right justified fields are padded and given the record-edit-character attribute, this confusion will not occur.)

The important thing to remember is that if the scan mask doesn't seem to be working properly, the first thing to check is if any right justified or decimally aligned fields have been entered incorrectly in the scan mask.

When using left justified fields or when fields are completely entered, operation of the scan mask is very simple. For example, if "CA" is entered into the state field, then only those records that have CA in the state will be displayed. For an address form, the zip code field might be set so that only records with a particular zip code would be displayed. Such a change would allow the operator to print all of the addresses for a given zip code area together on the line printer, while ignoring all other records with different zip codes.

It is important to note that any changes made to the scan mask will last until you exit from the current form, or until the scan mask is re-edited. You may return the scan mask to its original state by replacing any

changes with asterisks, by using ^G or ^V to clear individual fields or by changing forms and then restoring the original form. The third method is not recommended due to the amount of time it takes.

There is one final use of the scan mask. If your form contains any auto-duplicated character positions (See section 3.9), their contents can be changed by entering the value in the scan mask.

Chapter 5
Batch Processing

This chapter was designed to be read after Chapters 3 and 4 were read and the reader was familiar with the concepts and procedures of data entry, retrieval and modification.

This chapter describes batch file selection and verification.

5.1 Batch Files

In many large scale data entry applications, data is entered into the data base in groups of records called batches rather than as individual records. This method of operation is called batch processing and is provided by DataStar. In batch processing, records are entered as a batch, verified as a batch, and then entered into the main file in a process called pooling. DataStar accomplishes this by allowing entered records to be grouped in a separate file called a batch file. A DataStar batch file is a file containing records that have been entered but not verified or calculated.

Batch processing is useful in the following ways:

- It provides a means of monitoring the data entry process by keeping totals of numbers and amounts by batch.
- It allows for simultaneous entry of data by more than one operator.
- It allows the data to be verified by a different operator than the one that entered it, thereby reducing the chances of the same mistake being repeated.

Batch processing in DataStar consists of two steps: selecting a batch file for data entry and verifying the batch. The pooling step is omitted since records are entered into the primary file as soon as they are verified.

5.2 Selecting a Batch File

To select a batch file, type ^E to bring up the mode selector screen, followed by the letter B. In order to select batch files, the BATCH.OVR file

must be present on the logged drive. If this file is not present, the following prompt will be displayed:

```
BATCH.OVR not found.
Replace system disk, type RETURN
```

After the RETURN key is pressed, DataStar will exit to the operating system.

If the BATCH.OVR file is found, the following screen will appear:

```
ADD MODE                                    current form = ORDER
CURSOR:      ^A = prev field    ^S = left char     ^D = right char    ^F = next field
             ^T = first field   ^L = last field
FIELD EDIT:  ^G = delete char   ^V = insert hole   ^C = Copy from previous record
OTHER:       ^Z = restore screen ^U = print form   ^O = print data    ^J = help on/off
END/EXIT:    ^B = end entry     ^E = exit current mode
-------------------------------------------------------------------------------------------

                Enter disk drive (A/P...): _
                Enter name of batch file: --------
```

Type the letter of the disk drive (A-P), followed by the name of the batch file. DataStar will automatically create the named file (if it does not already exist) with the type .DTA, and also create an index file (name.NDX). If it is a new file, DataStar will automatically select ADD MODE; otherwise the mode in force prior to the batch file selection will be maintained. The disk drive and batch file name are displayed in the status line.

```
ADD MODE   /BATCH A:filename          current form = EXAMPLE
CURSOR:      ^A = prev field    ^S = left char     ^D = right char    ^F = next field
             ^T = first field   ^L = last field
FIELD EDIT:  ^G = delete char   ^V = insert hole   ^C = Copy from previous record
OTHER:       ^Z = restore screen ^U = print form   ^O = print data    ^J = help on/off
END/EXIT:    ^B = end entry     ^E = exit current mode
-------------------------------------------------------------------------------------------
```

Records are added to a batch file in the same way as to the main file, and may be scanned or selected by keys in order to modify or delete records as desired. To end data entry when using a batch file, complete entry of the current record, then type ^E to change modes. The Mode Selector Screen will be displayed with a new option:

```
ADD MODE    /BATCH  A:filename          current form = formn name
 Enter character to select new mode:

A =  Add new records          K =  select records by Key    E =  Exit current form
I =  SCAN in Index Order      D =  SCAN in Data file order  M =  edit scan Mask
B =  select Batch file        V =  Verify batch file        R =  Restore main file
F =  File maintenance         J =  Help                     SPACE = current mode
```

The R command allows you to return to the main file without verifying the records entered into the batch file. If the R command is used, the batch file is not verified, and the records are not merged into the main file.

If the batch selector screen is again invoked by typing ^EB, it will still contain the name of the last batch file entered. Type ^B to re-select the batch file.

5.3 Mandatory Batch File Selection

There are three cases where batch file selection is mandatory:

- whenever add mode is selected for forms that have the batch verify attribute,
- whenever verify mode is selected, and
- whenever file maintenance mode is selected.

In these cases DataStar will automatically switch to the batch file selector screen without any operator intervention; it is not necessary to type ^EB to initiate batch file selection.

The first case only occurs in forms that have the batch verify attribute. Some forms will require batch processing. For these forms, it will not be possible to add records directly into the main file; whenever ADD MODE is selected the batch file selector screen will be automatically displayed. Since ADD MODE is pre-selected for new forms, the first screen displayed when a batch processed form is brought up for the first time is the batch selector screen.

In the second case, a batch file must be selected since the verify process involves moving records from one file to another.

In the third case, a batch file is selected for use as a temporary work file while file maintenance is being performed. It will be removed from the

disk when the maintenance is complete. If the maintenance does not go to completion for some reason, (such as the disk becoming full) you may erase the batch file without losing any data.

5.4 Batch Verification

The batch verify process entails selecting verify mode, selecting a batch file and verifying the records in the batch file one by one. As each record in the batch is verified, it is removed from the batch file and transferred to the main file. The verification operation can be interrupted at any point and resumed later. When the batch has been verified, a message is displayed and another batch or different mode may be selected.

To enter verify mode, ^E is typed to bring up the mode selector screen, followed by the letter V. The batch file selector screen will be automatically displayed.

```
VERIFY MODE                                    current form = form name
 Enter character to select new mode:

A =  Add new records          K =  select records by Key    E =  Exit current form
I =  SCAN in Index Order      D =  SCAN in Data file order  M =  edit scan Mask
B =  select Batch file        V =  Verify batch file        R =  Restore main file
F =  File maintenance         J =  Help                     SPACE = current mode
 ------------------------------------------------------------------------------

                    Enter disk drive (A/B...): _
                    Enter name of batch file: --------
```

Enter the disk drive where the batch file resides, followed by the name of the batch file to be verified.

If the batch file is not located, a new batch file will be created and ADD MODE automatically invoked. If the batch file is found, the following screen will be displayed:

```
 VERIFY MODE /BATCH A:filename   current form = ORDER
CURSOR:      ^A = prev field     ^S = left char      ^D = right char     ^F = next field
FIELD EDIT:  ^G = delete char    ^V = insert hole
OTHER:       ^Z = restore screen ^U = print form     ^O = print data     ^J = help on/off
END/EXIT:    ^B = end entry      ^E = exit current mode
 ------------------------------------------------------------------------------
```

The cursor will be positioned at the first field in the form that requires verification. Not all fields are necessarily verified; in some cases only a few fields may require verification.

The ^A key function cannot be used to move to the previous field as is the case with other modes. Instead, ^A will cause the cursor to be moved to the start of the current field. If the cursor is already at the start of the field, ^A will be ignored.

There are two ways in which an operator will be asked to verify a field: by sight and by retyping the field. If by sight, the cursor will be moved into the field so the operator can look at the entered data one more time to be sure it is correct. After checking the field, the operator must press RETURN to indicate that the field has been checked. If retyping the field is required, the data in the field will be cleared and the operator required to re-enter the data exactly as it was the first time. If the two entries don't match, the field will again be cleared and the operator asked to enter the data twice more.

Included in verify mode are calculations and list verify errors. At various points in the verification process, field calculations may be suddenly displayed. In addition, if there are any list verify errors, they will be brought to the operator's attention at this time. A list verify error may occur for fields whose data is only accepted after it is compared against a list of valid entries. If a list verified field has a value entered that is not found on the verification list, the following message will appear:

Illegal field. Please re-enter. Hit ESC key.

The operator will be required to re-enter the field data.

After each field is verified, the cursor will move automatically to the next field requiring verification and/or correction. When all of the fields have been verified, the record is written into the main file and removed from the batch file. The next record is then automatically displayed.

In files that require unique keys, if the record being verified has the same key as a record already in the main file, the following message may be displayed:

Key already exists. Hit ESC key.

In this case, the operator must correct the key field or remove the duplicate entry from the main file before the verification process can continue.

When all of the records have been verified, the message:

The batch file has been verified.
Hit ESC key to enter another batch name.

will be displayed at the top of the screen. After pressing ESC, the current mode can be changed or another batch name entered for verification.

Appendix A
Field Definition Summary

This section is a summary of the recording, verification and display options available in DataStar. These options are assigned as field attributes in a question and answer session in FormGen called the field definition phase. This phase can be entered in FormGen whenever the cursor is postioned over a field and R is typed.

The field definition phase is a tree-structured questionaire, with many questions only being included conditionally, depending on the response to one or more previous questions.

This section starts out with a diagram of the questionnaire structure. In the remainder of the section, there is a page for each field definition option. On each page, the text following "Field Attribute:" is the prompt used for that option. For most options, a set of legal responses will follow in parentheses. The default or last entered value of the attribute will be displayed on the screen during each query.

The text following "Help Text:" is the help message that is displayed if a J is typed at any time in response to the field attribute prompt.

The text following "Additional Comments:" is additional information not available directly from FormGen.

Field Attribute Diagram

This is a diagram of the options available in the field definition phase of FormGen. The bold-faced text presented here is the text of the prompt for each item. The current attribute value is not shown. Whenever there are a limited number of responses, they are listed in parentheses. Conditional branches are indicated by

--(X)-->

where (X) is the condition for the branch. Unconditional branches are indicated by

------>

If no branch is taken, the item on the next line will be next. Indented items are reached if no branch is taken at a fork. Page numbers for each field attribute are listed at the right.

Field Attribute Diagram

Field Attribute:

Field order:

Help Text:

You have entered the field definition phase by typing a R. The field definition phase is a questionnaire which allows you to set up controls over what data will be accepted during the data entry portion of this program.

In the questions that follow, the answers you give will apply to the current field only. To return to background definition, type C.

This item allows you to specify the order in which fields will be processed during the data entry phase.

Additional Comments:

An order may be assigned to the various fields independent of their position on the screen. The order assigned here will affect:

1. the order that A and F move through the fields in FormGen.
2. the order in which fields will be presented for data entry in DataStar.
3. the order that the fields will have when the record is written to the disk.

The number assigned here is actually a field identification number. It is used whenever the field is referenced, either as part of a calculation or as the index field for a list derived field. If you change the field number, all references to the field will be automatically adjusted. In addition, whenever a field is added or removed, or its position in the order changed, the field numbers of the other fields and their references are also changed. For example, if field number two was removed from its position in the order (either by deleting it entirely or by moving it to the end of the order), fields three through n (where n was the number of fields) would now have the numbers two through n-1 and all references to them would be adjusted accordingly. If that is too confusing, just remember that you can change field orders freely and calculations, etc., will not be affected.

Field Attribute:

Key order:

Help Text:

The key field(s) is/are the field(s) on which the file is sorted. If there is more than one key field, they are combined for sorting. The field with key order of 001 will be the most significant field in the sort.

Additional Comments:

This option is only presented if the field being defined has been assigned key status. Key status can be assigned/removed by positioning the cursor at a field and typing K. Key status is indicated in FormGen by asterisks in the field. At least one field must be assigned key status in order to complete the form. If more than one field has been assigned key status then there is significance to the key order option. Key field order is initially assigned in the order that the fields were given key status. For example, if there already were three key fields and a fourth field was given key status, it would be assigned a key order of four.

Key fields are used at data entry time in two ways:

1. To impose an order on the data file.
2. To provide a means of rapid retrieval of data records.

For both purposes, a key is constructed (in the data entry program) from the key fields as follows:

1. The key fields are padded with spaces out to their full lengths.
2. The key fields are then combined in the specified key order with key field one being the most significant.

Once a key is constructed, it can be used for sorting the record or selecting a record quickly from the file.

Field Attribute:

Tie breaker field? (Y/N)

Help Text:

A tie breaker field is a key field entered by DataStar instead of the data entry operator, to create a unique key.

Enter a Y here to have DataStar maintain the file by entering the lowest numeric value that will create a unique key whenever necessary.

Additional Comments:

This query is only presented for the least significant key field in the form when there are more than one key fields in the form. If the tie breaker option is selected, the only other attributes that may be assigned to the field are:

Right Justify
Pad Field
Record Edit Characters.

Field Attribute:

Refuse duplicate keys? (Y/N):

Help Text:

This item is used to create files with unique keys. If the data entry operator attempts to enter a record with the same key as a record already in the file, an error message will be generated and the entry refused.

Additional Comments:

None.

Field Attribute:

Copy attributes of field:

Help Text:

This item is used to save time when entering field attributes. If another field is defined similarly, entering its field number here will cause its attributes to be used by this field also.

Additional Comments:

If the number here is changed from its default value (equal to the field number of the datafield being defined), the attributes of the specified field are automatically copied as soon as the entry is completed. The values displayed as the field definition process continues will be those of the field that was copied; all previous field attributes of the current field are lost. The exception to the above is field order and key-related attributes which remain unchanged. All items are copied exactly. Verify and calculate orders become those of the copied field; the order of the copied field is increased by one. If there was a range check or edit mask and the field lengths of the two fields are different, characters are removed or inserted at the cursor position within the field when R was typed. This position is given in the status line.

Field Attribute:

Field derived? (Y/N)

Help Text:

A field may be derived from others or from a list or file instead of being entered. Normally a field will simply be entered by the operator. A Y answer here will make this a derived field. The cursor will not stop here during data entry. Instead, the field value will be defined in the next few questions.

Additional Comments:

Fields are either entered or derived. If a field is to be entered, it will be presented to the operator for entry when its turn comes up in the field order sequence. If a field is to be derived, it will be skipped when its turn comes up during data entry.

There are two ways to derive a value for a field:

1. By calculating its value from other fields and constants.
2. By looking up its value in an external reference file.

If the first method is used, the field is called a calculated field. You will have to enter an expression that represents how the field is to be calculated and specify where in the calculation order the calculation for this field should occur. These items will be explained further in the following pages (A-10 through A-13).

If the second method is used, the field is called a list derived field. Its total specification will require an index field number and a field offset within record. These items will be explained further in the following pages. Note that use of the list derived option implies that an additional file exists and that its structure is compatible.

Field Attribute:

Allow operator entry? (Y/N)

Help Text:

Normally, a derived field will need no operator entry. Use of this option will allow the operator to position the cursor in the field after it has been derived in order to make adjustments.

Enter a Y to permit operator modifications to this field.

Additional Comments:

Once an operator has altered the contents of a derived field, it loses its derived status until the current record is disposed of. That is, the field will no longer respond to changes in other fields either by being recalculated or by getting looked up again when the contents of the index field changes. However, this option is useful for fields that can be "almost" derived. For instance, if a customer usually receives a certain discount, but can make special arrangements, this option can be used. The discount rate as a list derived field would be part of a customer file. In case of special arrangement, a different rate could be entered by the operator.

Field Attribute:

List/Calculated? (L/C)

Help Text:

A list derived field uses another field to index into a list or file and extract a value for this field. For example, you could use vendor name (entered elsewhere on the form) to pull vendor address into this field from a file of vendor name and addresses. A calculated field uses other fields and constants to create a value for this field.

Additional Comments:

None.

Field Attribute:

Index field number:

Help Text:

Enter the field number that you want to be used as an index into the list or file. For the example of the previous question, you would enter the field number for vendor name. If you don't remember the number of the index field, you can use A or F to move to the index field and get its number from the status line.

Additional Comments:

This item is part of the specification of a list derived field. Associated with every list derived field is an index field. Its field number is the number assigned in the "Field order:" attribute. An index field is the field in the current form that is to be used to select a record from the external reference file during the data entry process. A value for a list derived field is obtained during data entry by:

1. entering/deriving a value for the index field.
2. finding a record in the reference file/list that has a matching key field.
3. locating which field in the selected record is desired here.
4. transferring the contents of that field to the list derived field.

Note that:

1. Unlike key fields for the main file, the index for a reference file can only be a single field. There is no ordering of keys for the reference file.
2. The index field will be padded with spaces before the reference index file is searched for a matching entry. If an exact match in the index file key field is not found (including being padded to a different length with spaces) the file reference will fail.
3. The field chosen as an index field must be defined as a list verify field in order to complete the specification.

Field Attribute:

Item number in list:

Help Text:

Enter the number of the item that you wish to extract from the list or file. For the example of the previous two questions, if the vendor file had for each vendor:

name, address, city and state

you would enter "002" to extract address or "003" to get city/state.

Additional Comments:

Reference files always have at least two fields per record:

1. the key field (used to select a particular record from the file)
2. at least one data field

These fields can be positioned in the record in any order. They are numbered starting with one.

Item number in list is the second specification for a list derived field. It allows you to select which data item in the record belongs in a particular list derived field.

Field Attribute:

Verify/calculate order:

Help Text:

Field processing in verify mode includes verifications and calculations. In general, calculations should follow verifications unless the result of a calculation is being used as a file index field. The fields can be calculated in any order. The order the calculations are performed in will make a difference only when one calculation depends on the results of another.

Additonal Comments:

This is the first specification for a calculated field. It is used to specify the order in which calculations are performed. The reason it is called verify/calculate order instead of just calculation order, is that the verify operation is not performed as a separate step from the calculation operation. This is by design, to enable a calculated field to be used as an index field for a list derived field, since a list derived field is completed as part of the verify step.

If this is all very complicated, you can start out using calculated fields by knowing the following:

1. Calculation order is important only when one field is calculated using the results of another calculated field. For instance, if

   #005 = #1 + #2 (field 5 is calculated as the sum of fields 1 & 2)

   and if field 1 itself is calculated, you should have field 5 later in the calculation order than field 1.

2. If any of the fields used in the calculation are verified, you should have them verified before the calculation is performed. For the example above,

   #005 = #1 + #2,

   if field 2 is to be verified, you should have field 5 later in the verify/calculate order than field 2.

Field Attribute:

Numeric/String? (N/S)

Help Text:

A numeric calculated field uses an algebraic expression to derive a value for the field. The algebraic expression consists of fields and numeric constants combined arithmetically, using the operations:

add(+), subtract(-), multiply(*), divide(/), exponentiation(^)

A string expression consists of fields, sub-fields and string constants combined using the join (&) operation.

Additional Comments:

Choose a numeric calculation if you wish to manipulate numbers as numbers. Choose a string calculation when you wish to manipulate character strings. String operations are used to combine bits and pieces of data from one field or another.

In both numeric and string calculations:

The expression is not examined for legality until RETURN is typed. If any errors are detected at that time, the cursor is positioned at the illegal item after ESC is typed.

Changes to the expression are not recorded until RETURN is typed.

Field Attribute:

Enter string expression for field:
#000 =

Help Text:

A string expression may contain fields, subfields or string constants (literals) combined using the join (&) operation.Fields are represented by field number preceded by # sign: #7. Subfields are used to extract a portion of a field. The form is: field number (first character number, number of characters): #7(1,2). Literals are represented by characters enclosed in "s: "A01". The expression, #6(1,2)&#7(3), combines the first two characters of #6 with the rest of #7. Note that the number of characters is optional.

Additional Comments:

There is only one string operator, join, sometimes called concatenation. The operator, "join", joins separate characters or strings of characters (called operands) together to form a single string of characters. There are three types of string operands: fields, subfields and string constants.

Fields: Fields in a string expression are represented by a number sign followed by the field number. Field number is the number assigned in the "Field order:" attribute. Examples of valid fields are:

#5, #124, #002

assuming that there are at least 5, 124, or 2 fields in your form.

Sub-fields: Sub-fields are portions of fields. They are represented by the field number followed by starting character position and optional substring length. Examples of valid sub-fields are:

#5(1,2)	the first two characters of field five
#5(2,1)	the second character of field five
#5(3)	all but the first two characters of field five

Substrings are not padded with spaces if the field is not long enough to supply the requested number of characters.

String constants: A string constant, also called a literal, is represented by a

string of characters enclosed in quotes. Examples of valid string constants are:

"1", "2h9", "*^(^&"

A string expression is formed by combining string operands with the string operator, "join", so that there is one operator between every operand. Examples of valid string expressions are:

#5(1,2)	the first two characters of field five
#5(1,2)&" "	the first two characters of field five followed by a space
#5(1,2)&" "&5(3)	field five with a space inserted between the second and third characters.

Field Attribute:

Enter algebraic expression for field calculation:
#000 =

Help Text:

An algebraic expression may contain fields and numeric constants combined with the arithmetic operations: add(+), subtract(-), multiply(*),divide(/) and exponentiation(^). Operations are performed from left to right in the usual manner with parentheses recognized. Fields are represented by field number preceded by # sign: #007. Constants as well as field contents should contain only digits and optional minus sign and decimal point: 123, -1.005. Before arithmetic is performed, edit characters are removed from operands.

Additional Comments:

Operands for an algebraic expression are fields and numeric constants.

Fields: Fields in an algebraic expression are represented by a number sign followed by the field number. Field number is the number assigned in the "Field order:" attribute. Examples of valid fields are:

#5, #124, #002:

assuming that there are at least 5, 124, or 2 fields in your form.

Constants: Numeric constants may contain digits, a leading minus sign, or an embedded decimal point.

Examples of valid algebraic expressions:

100*(#5 + #6)
-.23 -(26.2*#4)/((1.001*#123^2 + 10) - 17.6*#15)

Field Attribute:

Required? (Y/N)

Help Text:

A Y answer here will force the data entry operator to enter something into this field. A N answer will allow the field to be skipped.

Additional Comments:

This item is actually redundant since the entry control mask has an edit character that requires entry in a given character position. This field attribute has been included since it is easier to use than the edit mask.

Field Attribute:

Right justify? (Y/N)

Help Text:

Right/left justify determines on which side of the field the data will be placed when it does not fill the field. Also if the field is padded, it will determine which side it will be padded from.

A Y answer will align the data entered with the right side of the field.

Additional Comments:

This item is purely for display purposes unless the field is padded or has a floating character and the record edit character attribute has been selected. It has no effect on the actual data recorded. Right justification is useful with numeric fields for lining up numbers in a column. Also, if a pad or floating character is used, the justification chosen will determine on which side of the data the pad or floating character(s) is/are inserted.

Field Attribute:

Pad field? (Y/N)

Help Text:

A field can be extended to its full length by padding on the right or left with some character.

A Y answer here will pad the field with some character if it is not full.

Additional Comments:

This item is purely for display purposes unless the Record Edit Characters attribute is assigned. Otherwise, it has no effect on the actual data recorded. It can be used to change the field designation character in the data entry program. On some terminals the underline character is annoying as a field designation character since it isn't a true underline. It may be desired in those cases to replace it with a space character or a period. It can also be used in printing checks to fill out the field with asterisks.

Field Attribute:

Enter pad character:

Help Text:

Any character may be used to extend the field to its maximum size. Commonly used characters are space and zero.

Additional Comments:

This character is not recorded unless the Record Edit Characters attribute has been assigned.

Field Attribute:

Floating character? (Y/N)

Help Text:

A 'floating character' can be inserted into the field, either on the left of the data for right justified fields or vice versa. For example if '$' is used as a floating character for a right justified field, a dollar sign will be added to the left of the data entered.

A Y answer will provide a floating character for the field.

Additional Comments:

This item is purely for display purposes unless the Record Edit Characters attribute has been assigned. Otherwise, it has no effect on the actual data recorded.

Field Attribute:

Enter floating character:

Help Text:

Any character may be used as a floating character. Commonly used characters are '$', and ' + ' or '-'.

Additional Comments:

None.

Field Attribute:

Verify field? (Y/N)

Help Text:

After the data entry for a form is complete, the program will enter a verify phase. In this phase some or all of the fields will be checked for accuracy.

A Y answer here will cause this field to be checked during the verification phase.

Additional Comments:

None.

Field Attribute:

Sight/Retype/List? (S/R/L)

Help Text:

There are 3 types of verification available: sight, retype and list. Sight verify just moves the cursor to each field allowing the operator to take another look at what has been entered. Retype verify will clear the field before moving the cursor there and force the operator to type the same information again. List verify will check the field against a list of correct field entries. In list verify, the list is kept in a separate file which may be prepared by any means.

Additional Comments:

The simplest and least accurate means of verification is by inspection. The danger is, of course, that the sight verification can appear to be a nuisance rather than a valuable procedural step.

Retype verification is very good, especially if a different operator performs the verification. In many applications, it is not unusual to have the entire record re-entered in order to improve accuracy.

List verification may be the best method of verification in cases where the list of legal entries is very much smaller than the list of possible entries. Examples of this are:

- a list of permitted zip codes
- a list containing the abbreviations for state
- a list containing the letters M and F

An additional consideration is that the list verify attribute is required for any field referenced as an index field by a list derived field. This is so that the specification of the file look-up can be completed.

Field Attribute:

Keep file in memory during data entry? (Y/N)

Help Text:

If the file is small enough (less than 500 characters) it will actually save space to keep it all in memory during data entry. It will always speed up processing to keep it in memory.

A Y answer here will cause the entire list file to be brought into memory before data entry, if it fits. A N answer, however, may be required here to leave enough room for other files to be loaded.

Additional Comments:

None.

Field Attribute:

Enter list file name:

Help Text:

A file name is 1-8 letters/digits.

Enter the name of the file that has a list of correct entries for this field.

Additional Comments:

None.

Field Attribute:

Enter list file disk drive (-/A/B...):

Help Text:

Enter A-P to specify a disk drive to use for the verification file. Enter "N" to use the current drive. In both cases, the operator will be asked to supply the disk drive if the file is not found on that drive.

Additional Comments:

The index file is assumed to reside on the same drive as the data file.

Field Attribute:

Item number of field:

Help Text:

The file record may contain more than just the item you wish to verify. This avoids unnecessary data duplication in data base systems as well as providing a means of deriving a field by retrieving it from another file. For example, a separate file having for each vendor:

name, address, city & state RETURN

might be used to verify vendor name.

Enter the item number within record that you wish to match against. The file will have to be sorted using this item as the sort key.

Additional Comments:

In the example above, the "Item number of field:" to enter would be 001.

Field Attribute:

Batch verify? (Y/N)

Help Text:

Batch verify fields do not get verified immediately after data entry. Instead, they are marked as un-verified and filed. This allows them to be verified as a batch on a different day or by a different operator in order to lessen the chances of a mistake in the entry being repeated in the verification.

Additional Comments:

If a Yes answer is given for this option, the data entry operator is forced to used batch entry all the time instead of having that as an entry-time option.

Field Attribute:

Verify/calculate order:

Help Text:

Field processing in verify mode includes verifications and calculations. In general, verifications should precede calculations unless the result of a calculation is being used as a file index field. The fields can be verified in any order. The field with verify order 001 will be verified first.

Additional Comments:

None.

Field Attribute:

Check Digit? (Y/N):

Help Text:

A check digit is a digit added on to the end of a numeric field in such a way so as to make the digits of the field obey some rule. It is usedto improve the accuracy of entered data, since incorrect data is not likely to obey the rule. The rule in DataStar is that the field must be divisible by eleven.

Enter Y here to require that the data in this field be divisible by 11.

Additional Comments:

None.

Field Attribute:

Range check? (Y/N)

Help Text:

A range check is useful for data that is only valid within a certain range. An example of this is a two character field called month. It can only have values between 01 and 12.

A Y answer here will allow you to specify the minimum and maximum values that the data in this field will be allowed to take on.

Additional Comments:

The range check can be used for alphanumeric data as well as numeric, since a character string comparison used for range check.

If a field value at data entry time does not fall within the specified range, the operator shown the required range and asked to correct the entry.

Field Attribute:

Enter/change the minimum field value:

Help Text:

A field is compared on a character by character basis to determine acceptablity. Characters are compared according to the following sequence:

space !"#$%&''()* + ,-./0-9:;< = >?@A-Z[\]^_`a-z{|}~ (highest)

Enter the smallest acceptable value for this field.

Additional Comments:

The field designation character (underline) will accept any character in that position. For example, if:

___63

was the minimum field value for a field of length four, the smallest legal value for the field would be " 63".

An empty character position in a field (at data entry time) always passes the range test for that character position. In other words, the underline character (or an empty character position) is treated like a wild card in the range test.

Field Attribute:

Enter/change the maximum field value:

Help Text:

A field is compared on a character by character basis to determine acceptablity. Characters are compared according to the following sequence:

space !"#$%&''()* + ,-./0-9:;< = >?@A-Z[\]^_'a-z{|}~
(highest)

Enter the largest acceptable value for this field.

Additional Comments:

The field designation character (underline) will accept any character in that position. For example, if:

___63

is the maximum field value in a field of length four, the largest legal value for the field would be ~~63.

An empty character position in a field (at data entry time) always passes the range test for that character position. In other words, the underline character (or an empty character position) is treated like a wild card in the range test.

Field Attribute:

Edit mask? (Y/N)

Help Text:

An edit mask allows you to control the entry and content of a field on a character by character basis. This includes inserting constants, copying data from the previous form, requiring entry at certain positions, as well as providing control over the actual characters that may be entered at each position.

A Y answer will allow you to enter an edit mask in two steps: entry and content control.

Additional Comments:

None.

Field Attribute:

Entry Control Character Codes:

! = must enter a character	**_ = may enter or leave blank**
X = automatic copy	**$\overline{Y}$ = auto-copy/may enter**
" = constant in this position	**~ = constant/may enter**
' = constant if data on both sides of constant	**/ = constant if data on one side only**

Enter/change the entry control mask:

Help Text:

The entry control mask uses one of the following characters: ! _ $\overline{X}$ Y"'~/ in each position of the field. This allows you to control whether or not there must be data there, as well as where it comes from. The constants are simply characters that you will supply in the next question. These characters will be embedded in the data each time the field is entered, unless the "'/" control codes are used. "'/" are conditional constants that are only inserted if enough characters are entered. For example, __'___'___ would allow insertion of commas into a seven digit number.

Additional Comments:

!	Required entry in this position.
_	Optional entry in this position.
$\overline{X}$	Will automatically copy a character from the previous record. This will permit, for instance, entering today's date at the beginning of an entry session, then having it copied automatically into subsequent forms as records are added.
Y	This is an automatic copy that permits operator adjustment afterwards .
"	Inserts the corresponding character from the content control mask (next page) into the data field.
~	Inserts the content control character while allowing operator override.
'	This inserts the content control character only if the character positions on both sides are filled.
/	This inserts the content control character only if the character position on one side or the other is filled.

Field Attribute:

Content Control Character Codes:

A = A-Z only	**a = a-z only**
B = A-Z, space	**b = a-z, space**
C = A-Z, a-z -> A-Z	**c = A-Z, a-z**
D = A-Z, a-z -> A-Z, space	**d = A-Z, a-z, space**
E = A-Z, 0-9	**e = a-z, 0-9**
F = A-Z, 0-9, space	**f = a-z, 0-9, space**
G = A-Z, 0-9, a-z -> A-Z	**g = A-Z, 0-9, a-z**
H = same as G with space	**h = A-Z, 0-9, a-z, space**
9 = 0-9 only	**8 = 0-9, #$%()* + -.**
. = align decimal point	**_ = any character allowed**

Enter/change the content control mask:

Help Text:

Each character in the content mask controls the data characters that will be allowed in that position in the data to be entered. For instance, a content and entry control mask for a phone number is:

999-9999	(content mask)
!!!"!!!!	(entry mask)

The "9"s will permit only digits in those positions and a "-" will automatically be inserted where it belongs.

Additional Comments:

There are three points that might not be clear.

1. The C, D, G and H formats automatically convert lower to upper case.

2. The content control character codes, AaBbCcDdEeFfGgHh98._, are not enforced when one of the entry control constant codes, "~'/, is present in that character position. If one of the constant codes are used, then any character can be entered here and it will be used as a literal character and inserted into the field if conditions permit.

3. The period format is unique in several ways. First, only one period

can be present in a content control mask. Second, the only character that can be typed at that position is a period, so it serves to align the field with the decimal point. And finally, in a calculated field, unless there is a decimal character in the content mask, non-integral values cannot be stored.

Field Attribute:

Record Edit Characters? (Y/N):

Help Text:

Edit characters are pad/float characters and constants from the edit mask. Usually they are removed from the field before the data is recorded.

A Y answer here will include edit characters as well as data in the disk file.

Additional Comments:

This query is only presented if the pad, float or edit mask attributes have been selected.

Appendix B

INSTALLATION AND MODIFICATION

The "Installation" procedure sets up DataStar to work with your particular terminal and printer. This procedure enables one distributed DataStar version to work with a wide variety of computer equipment. For most cases, installation is accomplished by entering choices from menus, as described in this section. For exceptional terminals and printers, and for achieving refinements in installation, there are provisions for "patching" by a programmer as described at the end of this section.

Terminal Installation Considerations

DataStar's installation dialog can automatically supply the necessary control codes for many of the most common terminals, including the following:

ADDS REGENT 20/25
ADDS REGENT 40/45/60/65
ADDS VIEWPOINT
BEEHIVE 150 / CROMEMCO 3100
COMPUCOLOR 8001G
FLASHWRITER II
HAZELTINE 1420
HAZELTINE 1500
HEATH or ZENITH H89/H19
HEWLETT-PACKARD 2621 A/P
IBM 3101
IMSAI VIO
INFOTON I-10
LEAR-SIEGLER ADM-3A
LEAR-SIEGLER ADM-31
MICROTERM ACT-IV
MICROTERM ACT-V
PERKIN-ELMER 550 (BANTAM)
SOROC IQ-120
TEC MODEL 571
TELEVIDEO 912/920
TELEVIDEO 950
VISUAL 200

If your terminal is not shown, you will need to determine its control codes for cursor positioning and other (optional) functions from the manufacturer's manual, then "patch" them in as described later in this section. If this is the case, skip the sections describing the use of the DINSTALL program and turn immediately to the section describing DataStar modfication.

Some terminals have option switches which must be correctly set:

CURSOR CONTROL, if present, must be ON;
AUTO LINE FEED, if present, must be OFF.

Usually such switches are hidden under a cover, on the back of the terminal, or inside the case.

The Distribution Diskette

The diskette you receive upon purchasing DataStar contains the following files:

DATASTAR.COM	Uninstalled DataStar; installation for your terminal is achieved by running DINSTALL as described later in the section.
FORMGEN.COM	Uninstalled FormGen; installation for your terminal is achieved by running DINSTALL as described later in the section.
DINSTALL.COM	DataStar installation program; used to customize (install) DataStar for a particular hardware configuration.
BATCH.OVR	This is an auxillfiary program required for batch file selection.
ORDER ORDER.DTA ORDER.NDX	This is a sample form demonstrating many of DataStar's features. It may be useful to examine the ORDER form using FormGen. All of the non-trivial options are exercised in this form.

The following files are used as reference files by the ORDER form:

PRODUCTS.DTA PRODUCTS.NDX	This is a sample file with records containing product code, description and price.
OKSTATES.DTA OKSTATES.NDX	This is a sample file containing a two digit postal abbreviation of state, followed by the sales tax in that state.

CUSTOMER.DTA CUSTOMER.NDX	This is a sample file with records containing customer name, address, city, state, zipcode and customer number.

Installation Procedure

Before installing DataStar, we recommend that you copy the files from the DataStar distribution disk to a system disk prepared as required for your system. Refer to your operating system documentation for the procedure to format a disk (if required) and to transfer the operating system's "system image" to the disk (this transfer is accomplished in many systems with a program called SYSGEN). Then copy the files from the DataStar distribution disk to this prepared disk.

To initiate DataStar installation, insert the disk containing DATASTAR.COM, FORMGEN.COM, BATCH.OVR, and DINSTALL.COM, type control-C (if your system is already up, or else cold-start (boot) the system), and type the command

DINSTALL

and a carriage return. DINSTALL will take several seconds to load, after which it will display a sign-on message and ask its first question. At this point, the screen should appear as follows:

```
A>DINSTALL

COPYRIGHT (C) 1982, MicroPro International Corporation

INSTALL version 3.81 for MicroPro DataStar release 1.4
```

Selecting the Terminal Type

The terminal choice "menu" appears next, as follows:

```
MicroPro DataStar release n.nn serial 000000

          *****DataStar TERMINAL MENU #1*****

A  Lear-Siegler ADM-3A              C  Lear-Siegler ADM-31
D  Hazeltine 1500                   E  Microterm ACT-IV
F  Beehive 150/Cromenco 3100        G  IMSAI VIO
H  Hewlett-Packard 2621 A/P         I  Infoton I-100
J  Processor Tech SOL/VDM           K  Soroc IQ-120
L  Perkin-Elmer 550 (Bantam)        Z  none of the above
2  Terminal menu #2

PLEASE ENTER SELECTION (1 LETTER):
```

Enter the letter corresponding to your terminal type. If your terminal type is not shown on the menu and is listed at the start of this section, enter 2 or 3 to bring up the second or third menu.

After the terminal type letter is entered, DINSTALL will display a brief message confirming the terminal type and ask

OK (Y/N):

If you made an error or wish to change your selection, type an N, and the TERMINAL TYPE question will be asked again. Upon repetition, a choice U for "NO CHANGE" will appear on the menu. When you answer Y to OK(Y/N):, DINSTALL will patch the DataStar and FormGen file. The installation is now complete.

MODIFICATION

Modification by patching is used in installing DataStar for terminals not included in the installation menus. Since basic installation is achieved with the INSTALL program ("installation dialog"), most users should have little or no need to patch.

"Patching" means modifying the programs (DATASTAR.COM and FORMGEN.COM) by changing the contents of bytes in it, with the debugger (DDT in most systems; DEBUG in some; must be purchased separately from certain operating system suppliers) or with the front panel switches. The rest of this section is written for the reader who already understands patching, usually an assembly language programmer. We will not attempt to explain patching technique or to describe the background concepts.

Listings in Appendix: Appendix C of this manual contains an absolute assembly listing of the user-patchable portion of DataStar, USER. USER is the module containing all terminal-installation related patch areas.

The listing is commented heavily; read it for a detailed description of each item. The following paragraphs give general descriptions of the modules, and hit a few high points of the details, but do not attempt to duplicate the detailed information contained in the program comments.

The module whose listing is supplied was used with a linking loader after assembly; it contains external references whose values are not shown in the listing. Generally, these are in locations there will be no need to change, but if you do need to move or use one of them, examine the locations with your debugger (DDT, or whatever your version is called) to determine the value. These values will change with each release of DataStar.

Installation for a Terminal Not Shown on Terminal Menu

Before patching DataStar for a terminal not on the installation dialog terminal selection menu, it is necessary to determine that the terminal is suitable for use with DataStar, and to determine the terminal's control codes from the terminal manufacturer's manual.

Terminal Requirements: To be used with DataStar, the terminal must have a screen at least 23 lines high by 80 columns wide. The terminal must have the ability to position the cursor at any position on the screen.

Determining Control Codes: study the terminal manufacturer's manual for your terminal to determine the screen size and the control codes which DataStar must transmit to the terminal to accomplish the cursor positioning.

Position Cursor: Positioning the cursor is the most complex terminal function used by DataStar, since it involves transmitting the row and column at which the cursor is to be placed. The form of the control string sent to the terminal to position the cursor varies widely from terminal to terminal; DataStar's cursor positioning patch items are designed to accomodate almost all commonly used forms. For these reasons, it is necessary to analyze your terminal's cursor positioning sequence to derive the information needed to patch DataStar.

Some terminals have a single control sequence to position the cursor at a line (row) and column (character) number; others have separate control sequences to position the cursor on a line and at a column; either will work with DataStar. In the former case, the terminal may require either the row or column transmitted first. For some terminals, the line and column numbers are sent with no offset, that is, a zero is sent for the top line on the screen (or left column), a one for the next line down (next column over), etc. For others, an "offset" is added. For example, several terminals require 20 (hexadecimal) for the top line or left column, 21 for the second line (or column), etc.

Analyze the cursor positioning codes for your terminal and determine:

1. Number of characters and character values that must be sent before either the line or column is transmitted.

2. Whether the row or column is transmitted first.

3. Number of characters and character values that must be sent after the row (or column, whichever is sent first), and before the other dimension is sent. For many terminals, this is no characters.

4. Number and values of characters to send after both row and column, if any–usually none.

5. Offset to add to row number–usually zero or 20 hex.

6. Offset to add to column number–usually same as for row.

Cursor Example A: terminal requires, in order, 1B hex, 3D hex, line number plus 20 hex, column number plus 20 hex, to set cursor position. The answers to the above questions are:

1.	2, 1B, 3D
2.	row is first
3.	none
4.	none
5 and 6.	20 hex

Cursor Example B: terminal requires 1B hex, 01 hex, line number to set cursor to line, and 1B hex, 02 hex, column number to set cursor to column. The above questions may be answered as follows:

1.	2, 1B, 01
2.	row is first
3.	2, 1B, 02
4.	none.
5 and 6.	0

If you have an unusual terminal, you may find that it is not possible to analyze its required cursor positioning commands into answers to the above six questions -- e.g., it may be necessary to transmit the row and column numbers as strings of ASCII digits. For such cases, you must write a subroutine to generate the necessary cursor positioning command sequence.

Highlighting (Optional): you also may want to determine the codes to invoke highlighting, if available, by inverse video display, bright/dim display, or some other method. If both bright/dim and character-by-character inverse video are available, we suggest inverse video, as it will highlight blanks as well as other characters, and stands out strongly. The highlighting method must operate on a character-by-character basis, take no screen positions, and not be tied to field protection. Determine the code or code sequence to turn high-lighting on, and the sequence to turn highlighting off. The "Highlighting on" sequence should cause subsequently transmitted characters to be highlighted until a "Highlighting off" sequence is transmitted. DataStar always turns highlighting off before transmitting a carriage return, line feed, or cursor positioning sequence; therefore, it does not matter whether these functions cancel highlighting on your terminal or not.

APPENDIX C

LISTINGS OF USER MODIFIABLE AREAS

A listing of the DataStar user modifiable area follows this page. This assembly listing is supplied for the convenience of the occasional user who may need to install DataStar for an unusual terminal. See Appendix B for a discussion of these modifications.

```
                        TITLE USER    9 / 30 /80   TERMINAL PATCH AREA


                        ENTRY HITE,WID,CLRSCR
                        ENTRY CLEAD1,CLEAD2,CTRAIL,CB4LFG,LINOFF,COLOFF,UCRPOS,ERAEOL
                        ENTRY IVON,IVOFF,TRMINI,TRMUNI,INISUB,UNISUB,USELST
                        ENTRY DELCLR,DELCUS,DELMIS,MEMAPV,MEMADR,HIBIV,HIBCUR,CRBLIV
                        ENTRY MORPAT,PBGMEM,ITHELP

                                EXT OUTCHR, MEMORY, MSTART

0100                    TBASE   EQU     100H    ;WHERE TO LOAD PROGRAM

0000'                           ASEG            ;ABSOLUTE CODE, FOR LISTING

                                ORG TBASE+140H  ;BEGINS AT 240 HEX

                                PAGE 63         ;SETS PAGE LENGTH


                        ; ************************************************
                        ; *                                              *
                        ; *   USER-MODIFIABLE CONSTANTS AND ROUTINES FOR  *
                        ; *   HARDWARE-DEPENDENT TERMINAL CHARACTERISTICS  *
                        ; *   AND FUNCTIONS USED BY EDITOR                *
                        ; *                                              *
                        ; ************************************************

                        ;NOTE: THIS AREA IS NORMALLY PATCHED FOR YOUR TERMINAL
                        ;TYPE VIA THE INTERACTIVE INSTALL PROGRAM.  ADDITIONAL
                        ;PATCHING TO THIS AREA IS NEEDED ONLY FOR UNUSUAL TERMINALS
                        ;OR UNUSUAL VIDEO BOARDS, OR TO MEET SPECIAL REQUIREMENTS
                        ;OR ENHANCE OR PERSONALIZE PERFORMANCE.

                        ;
                        ; PATCHES ALWAYS NEEDED: SCREEN HITE & WIDTH, CURSOR.
                        ;
                        ; ALL OTHERS ARE OPTIONS FOR SPECIAL CASES OR ENHANCEMENT.
                        ;
```

```
                                ;
                                ;    SCREEN SIZE
                                ;
                                ;    SCREEN HEIGHT AND WIDTH PATCHES ARE MANDATORY.
                                ;
0240    18                      HITE:   DB 24   ;MUST BE EXACT SCREEN HEIGHT IN LINES
0241    50                      WID:    DB 80   ;MUST BE <= EXACT SCREEN WIDTH


                                ;
                                ; IN ALL CHARACTER STRINGS TO BE SENT TO TERMINAL,
                                ; FIRST BYTE IS NUMBER OF CHARACTERS, CHARACTERS FOLLOW.
                                ;


                                ;
                                ; CLEAR SCREEN SEQUENCE
                                ;  SHOWN SET TO SEND CONTROL-Z AS AN EXAMPLE.
                                ;
                                CLRSCR:
0242    01                              DB 1            ;LENGTH: 1 CHARACTER
0243    1A                              DB 1AH          ;FIRST CHARACTER: ESCAPE
0244    00                              DB 0            ;SECOND CHARACTER
0245    00 00 00 00                     DB 0,0,0,0      ;3RD THRU 6TH GO HERE
                                                        ; .. IF NEEDED
0249    00 00 00 00                     DB 0,0,0,0      ;ROOM FOR MORE


                                ;
                                ;           PROVISIONS FOR PATCHING
                                ;       CURSOR POSITIONING CONTROL SEQUENCES
                                ;
                                ;       CURSOR POSITIONING PATCH IS MANDATORY.
                                ;
                                ; CURSOR POSITIONING FOR MOST TERMINALS
                                ; IS ACCOMPLISHED BY SENDING:
                                ;
                                ;   1.  A "LEAD-IN" STRING OF ONE OR MORE
                                ;               TERMINAL-SPECIFIC CHARACTERS;
                                ;   2.  THE LINE NUMBER, WITH AN OFFSET (OFTEN 0)
                                ;               ADDED; OR, FOR SOME TERMINALS,
                                ;               THE COLUMN NUMBER IS SENT FIRST;
                                ;   3.  ANOTHER LEAD-IN STRING, FOR SOME TERMINALS;
                                ;   4.  THE COLUMN (OR LINE) NUMBER, WITH OFFSET;
                                ;   5.  ANOTHER STRING (FOR SOME TERMINALS).
                                ;
                                ;   FOR MOST TERMINALS, THE LINE AND COLUMN ARE SENT
                                ;   AS 1-BYTE BINARY NUMBERS; FOR A FEW TERMINALS,
                                ;   TWO- OR THREE-DIGIT ASCII NUMBERS ARE SENT.
                                ;
                                ; FOR TERMINALS THAT DON'T FIT THE ABOVE PATTERNS,
                                ; YOU MUST CODE YOUR OWN SUBROUTINE.

                                      ;CURSOR PATCH AREAS ARE ON NEXT PAGE
```

```
                        ; CURSOR POSITIONING...
                        ;               SEE COMMENTS PRECEDING PAGE

                        ; FOR EXAMPLE, THE CURSOR IS POSITIONED ON THE
                        ; ADM-3A TERMINAL BY SENDING:
                        ;       ESCAPE, =,
                        ;       LINE # PLUS 20 HEX,
                        ;       COLUMN NUMBER PLUS 20 HEX.
                        ; THE FOLLOWING PATCH AREAS ARE SET UP FOR THIS
                        ; TERMINAL, AS AN EXAMPLE.

                        ; CURSOR POSITIONING INITIAL LEAD-IN STRING
024D    02              CLEAD1: DB 2    ;NUMBER OF CHARACTERS
024E    1B                      DB 1BH          ;FIRST CHARACTER
024F    3D                      DB '='          ;SECOND CHARACTER
0250    00 00 00                DB 0,0,0        ;SPACE FOR MORE
0253    00 00 00                DB 0,0,0        ;.. CHARACTERS

                        ; CURSOR POSITIONING STRING SENT BETWEEN
                        ;   LINE AND COLUMN
0256    00              CLEAD2: DB 0    ;NUMBER OF CHARACTERS -
                                        ; NONE IN OUR EXAMPLE
0257    00                      DB 0            ;FIRST CHARACTER
0258    00 00 00                DB 0,0,0        ;SECOND THRU 4TH

                        ; CURSOR POSITIONING STRING SENT AFTER
                        ;   BOTH LINE AND COLUMN HAVE BEEN SENT
025B    00              CTRAIL: DB 0    ;NUMBER OF CHARACTERS (NONE)
025C    00 00 00 00             DB 0,0,0,0

                        ; FLAG NON-ZERO TO SEND COLUMN BEFORE LINE
0260    00              CB4LFG: DB 0    ;LINE GOES BEFORE
                                        ;..COLUMN IN OUR EXAMPLE

                        ; OFFSET TO ADD TO LINE
0261    20              LINOFF: DB 20H  ;ADD THIS TO LINE #
                                        ;(WHERE 0 IS TOP LINE ON
                                        ; SCREEN BEFORE OFFSET)

                        ; OFFSET TO ADD TO COLUMN
0262    20              COLOFF: DB 20H  ;ADD THIS TO COLUMN #
                                        ;(WHERE 0 IS LEFT EDGE OF
                                        ; SCREEN BEFORE OFFSET)

0263    00                      DB 0    ;RESERVED

                        ;SEE NEXT PAGE FOR POSITIONING
                        ;CURSOR VIA USER-CODED SUBROUTINE

                                ;CURSOR POSITIONING...

0264    00 00 00                DB 0,0,0        ;BYTES RESERVED FOR EXPANSION
                        ;
                        ; PROVISION FOR POSITIONING CURSOR BY USER-CODED
                        ; SUBROUTINE INSTEAD OF UNDER CONTROL OF ABOVE ITEMS,
                        ; FOR USE IN EXCEPTIONAL CASES ONLY:
```

```
                                ;
                                ;  PUT A  JMP  INSTRUCTION TO YOUR SUBROUTINE IN
                                ;  FOLLOWING 3 BYTES.  WHENEVER FIRST BYTE IS
                                ;  NON-0, THIS LOCATION WILL BE CALLED TO POSITION
                                ;  CURSOR; ABOVE CURSOR PATCH ITEMS WILL BE DISREGARDED.
                                ;
0267    00                      UCRPOS: NOP     ;NORMALLY 0, OR JMP TO YOUR
0268    00                              NOP     ;SPECIAL CURSOR POSITIONING
0269    C9                              RET     ;ROUTINE.

                                ; SEE "MORPAT" AT THE END OF THIS LISTING
                                ; FOR SPACE TO PUT YOUR SUBROUTINE IN.

                                ;     YOUR SUBROUTINE WILL RECEIVE LINE IN L REGISTER
                                ; (0=TOP SCREEN), COLUMN IN H (0=LEFT EDGE).
                                ;     YOUR SUBROUTINE MAY ALTER ALL REGISTERS.
                                ;     TO OUTPUT A CHARACTER TO THE TERMINAL FROM WITHIN
                                ; YOUR SUBROUTINE, CALL OUTCHR.  OUTCHR IS ASSEMBLED AT
                                ; ABSOLUTE LOCATION 109H AND REPRODUCED HERE.

026A    0000*                           DW OUTCHR       ;ADDRESS OF BYTE OUTPUT
                                                        ;ROUTINE LOADED HERE

026C    00 00 00 00                     DB 0,0,0,0      ;RESERVED BYTES


                                ;
                                ;EVERYTHING THAT FOLLOWS IS OPTIONAL.
                                ;  EACH ITEMS RELATE EITHER TO ENHANCED PERFORM-
                                ;  ANCE (FOR EXAMPLE, USE OF INVERSE VIDEO
                                ;  OR BRIGHT/DIM TO SET OFF BLOCKS OF TEXT),
                                ;  OR TO ACCOMODATING UNUSUAL TERMINALS.
                                ;

                                 ;ERASE TO END OF LINE. OPTIONAL - IF FUNCTION
                                 ;NOT AVAILABLE, LEAVE FIRST BYTE ZERO AND DATASTAR
                                 ;WILL DO THE FUNCTION MORE SLOWLY VIA SOFTWARE.

0270    00                      ERAEOL: DB 0    ;PUT NUMBER OF CHARACTERS HERE
0271    00                              DB 0    ;PUT FIRST CHARACTER HERE
0272    00                              DB 0    ;IF 2-CHAR SEQUENCE, PUT 2ND HERE
0273    00 00 00 00                     DB 0,0,0,0  ;IF 3 TO 6 CHAR SEQUENCE

0277    00 00 00 00                     DB 0,0,0,0      ;RESERVED FOR FUTURE EXPANSION
027B    00 00 00 00                     DB 0,0,0,0
027F    00 00 00 00                     DB 0,0,0,0
0283    00 00 00 00                     DB 0,0,0,0


                                ;TURN ON HIGHLIGHTING (INVERSE VIDEO, BRIGHT/DIM, OR
                                ;OTHER MEANS OF HIGHLIGHTING A SECTION OF TEXT).
                                ;APPLICABLE ONLY IF MEMAPV (NEXT PAGE) IS 0. OPTIONAL.
```

```
0287  00              IVON:   DB 0            ;LENGTH
0288  00 00 00                DB 0,0,0        ;UP TO 6 CHARACTERS
028B  00 00 00                DB 0,0,0

                      ;TURN OFF HIGLHLIGHTING. OPTIONAL.
                      ;IF NO STRING PATCHED IN, IVON STRING WILL BE
                      ;USED TO TURN OFF AS WELL AS ON.

028E  00              IVOFF:  DB 0            ;LENGTH
028F  00 00 00                DB 0,0,0        ;UP TO 6 CHARACTERS
0292  00 00 00                DB 0,0,0

                      ;TERMINAL INITIALIZATION STRING: ANY BYTES YOU
                      ;WANT SENT TO YOUR TERMINAL AT BEGINNING OF
                      ;SESSION. OPTIONAL. SEE ALSO INISUB, NEXT PAGE.
0295  00              TRMINI: DB 0            ;LENGTH
0296  00 00 00 00           DB 0,0,0,0      ;THE BYTES,
029A  00 00 00 00           DB 0,0,0,0      ;MAX LENGTH 8.

                      ;ANY BYTES YOU WANT SENT TO TERMINAL AT END OF
                      ;SESSION.  OPTIONAL. SEE ALSO UNISUB
029E  00              TRMUNI: DB 0            ;LENGTH
029F  00 00 00 00           DB 0,0,0,0      ;THE BYTES,
02A3  00 00 00 00           DB 0,0,0,0      ;UP TO 8.

                      ;
                      ;USER-PATCHABLE INITIALIZATION SUBROUTINE. CALLED
                      ;JUST BEFORE TRMINI (PREVIOUS PAGE) IS SENT, THIS
                      ;SUBROUTINE MAY BE USED FOR SPECIAL CONSOLE INIT-
                      ;IALIZATION OR OTHER PURPOSES.
                      ;
02A7  00              INISUB: NOP     ;ENTRY POINT. PUT DESIRED CODE IN
02A8  00                      NOP     ;"MORPAT" AREA (3 PAGES AHEAD IN
02A9  C9                      RET     ;THIS LISTING) AND PATCH IN A "JMP"
                                      ;INSTRUCTION HERE.  FOR YET MORE
                                      ;SPACE, SEE "PBGMEM" ON SAME PAGE.
                      ;
                      ;USER-PATCHABLE DE-INITIALIZATION SUBROUTINE.
                      ;CALLED AT EXIT (JUST BEFORE TRMUNI IS SENT),
                      ;THIS SUBROUTINE MAY BE USED TO "UNDO" ANY SPECIAL
                      ;TERMINAL STATUS USED IN DATASTAR.
                      ;
02AA  00              UNISUB: NOP     ;PUT DESIRED CODE IN
02AB  00                      NOP     ;"MORPAT" AREA, PATCH IN
02AC  C9                      RET     ;A "JMP" HERE.
```

```
                        ;
                        ; MISCELLANEOUS TERMINAL-RELATED ITEMS
                        ;

                         ;FLAG TO PERMIT DISPLAY IN LAST COLUMN OF LAST LINE.
                         ;INITIALLY DISABLED, AS MANY TERMINALS SCROLL SCREEN
                         ; WHEN A CHARACTER IS DISPLAYED IN THIS POSITION.

02AD    00               USELST: DB 0     ;PATCH NON-0 TO PERMIT LAST CHAR ON
                                          ;LAST LINE TO DISPLAY IF THIS WILL
                                          ;NOT SCROLL SCREEN OF YOUR TERMINAL.
                                          ;NORMALLY LEAVE 0 FOR USE WITH
                                          ;TERMINAL, PATCH NON-0 FOR USE
                                          ;WITH MEMORY-MAPPED VIDEO BOARD.

02AE    00                       DB 0     ;BYTE RESERVED FOR EXPANSION

                         ;
                         ;DELAYS EXECUTED AFTER VARIOUS TERMINAL FUNCTIONS,
                         ;BEFORE NEXT CHAR IS SENT TO TERMINAL, TO ALLOW RESPONSE
                         ;TIME REQUIRED BY CERTAIN TERMINALS WHEN USED AT HIGH
                         ;BAUD RATES.  PATCH LARGER IF YOU EXPERIENCE, FOR
                         ;EXAMPLE, LOSS OF CHARACTERS AFTER CURSOR POSITIONING.
                         ;EACH DELAY IS APPROX NUMBER OF MILLISECONDS ON
                         ;4 MHZ Z-80; DELAY IS ABOUT TWICE AS LONG ON 2 MHZ 8080.
                         ;
02AF    19               DELCLR: DB 25    ;25+ MSEC DELAY AFTER CLEAR SCREEN
02B0    0A               DELCUS: DB 10    ;10+ MSEC DELAY AFTER CURSOR SET
02B1    05               DELMIS: DB 5     ;5+ MSEC DELAY AFTER OTHER FUNCTIONS

                         ;PROVISION FOR MEMORY MAPPED VIDEO BOARD:
                         ; IF YOUR CONSOLE DEVICE IS A MEMORY MAPPED VIDEO
                         ; DISPLAY MEETING THE RESTRICTIONS GIVEN IN THE MANUAL,
                         ; THE FOLLOWING MAY BE PATCHED TO CAUSE DATASTAR TO
                         ; ACCESS IT DIRECTLY INSTEAD OF VIA CP/M. DOING SO
                         ; THIS PROVIDES EXTREMELY FAST DISPLAY, AND MAKES
                         ; MORE MEMORY AVAILABLE FOR THE FORM BEING PROCESSED.
                         ;
02B2    00               MEMAPV: DB 0     ;NON-0 TO USE MEMORY-MAPPED
                                          ;VIDEO DISPLAY BY DIRECT STORAGE
02B3    0000             MEMADR: DW 0     ;ADDRESS VIDEO SCREEN RAM, TOP LINE,
                                          ;LEFTMOST COLUMN.  THE REST OF THE
                                          ;SCREEN MUST BE AT CONTIGUOUS
                                          ;ASCENDING ADDRESSES.
                         ;
                         ;FLAG TO SAY INVERSE VIDEO, HIGH BRIGHTNESS,
                         ;OR OTHER MEANS OF HIGHLIGHTING CHARACTER WILL
                         ;OCCUR IF HIGH ORDER BIT OF CHARACTER IS ON.
                         ;CAUSES HIGH ORDER BIT TO BE USED IN DISPLAYING
                         ;PROMPTS AND TEXT WHICH IS
                         ;"MARKED" TO BE MOVED, DELETED, ETC.
                         ;
02B5    00               HIBIV:  DB 0     ;PATCH NON-0 IF YOU HAVE INVERSE
                                          ;VIDEO ETC. INVOKED BY HI ORDER BIT.
```

```
                        ;NOTE: HIBIV RELATES ONLY TO MEMORY MAPPED
                        ;OPERATION; IVON AND IVOFF RELATE ONLY TO TERMINALS.

                        ;
                        ;FLAG TO INVOKE DISPLAY OF CURSOR ON MEMORY MAPPED
                        ;VIDEO BOARD BY SHOULD BE PRODUCED BY SETTING HIGH
                        ;ORDER BIT OF CHARACTER UNDER CURSOR, RATHER THAN
                        ;VIA "CRPOS" SUBROUTINE.  THIS SAVES TIME AND
                        ;ELIMINATES INTERACTION PROBLEMS THAT CAN ARISE IF
                        ;CHARACTER DISPLAY IS BY DIRECT STORAGE BUT CURSOR
                        ;POSITIONING IS VIA THE EXTERNAL BOARD DRIVER.
                        ;NOTE: CRPOS SHOULD STILL BE PATCHED TO
                        ;INTERFACE TO EXTERNAL CURSOR POSITION FUNCTION;
                        ;IT IS CALLED AT ENTRY AND EXIT TO PLACE INTERNAL
                        ;AND EXTERNAL CURSORS IN SAME PLACE.
                        ;
02B6    00              HIBCUR: DB 0    ;NON-0 TO DISPLAY CRSR BY SETTING
                                        ;HIGH ORDER BIT OF CHARACTER

                        ;
                        ;FLAG TO SAY CURSOR MUST BE BLINKED BY DATASTAR
                        ;IN ORDER TO BE VISIBLE WHEN ON OR ADJACENT TO AN
                        ;INVERSE VIDEO (OR OTHERWISE HIGHLIGHTED)
                        ;CHARACTER.  THIS SHOULD BE OFFH IF YOUR CURSOR
                        ;IS NOT VISUALLY DISTINCT FROM YOUR INVERSE VIDEO
                        ;AND INVERSE VIDEO IS USED (VIA IVON/IVOFF OR HIBIV).
                        ;NORMALLY NEEDED WITH MEMORY MAPPED VIDEO BOARDS
                        ;WITH HIBIV AND HIBCUR BOTH OFFH.
                        ;BLINK METHOD USED FOR A TERMINAL: ALTERNATELY
                        ;SEND IVON, IVOFF STRINGS, WITH DELAYS.
                        ;
02B7    00              CRBLIV: DB 0    ;FF HEX FOR DATASTAR TO BLINK CURSOR
                                        ;WHEN ON INVERSE VIDEO CHARACTER.

                        ;
                        ; RESERVED SPACE
                        ;
02B8    00 00 00 00             DB 0,0,0,0      ;SPACE RESERVED FOR EXPANSION
02BC    00 00 00 00             DB 0,0,0,0      ;SPACE RESERVED FOR EXPANSION
02C0    00 00 00 00             DB 0,0,0,0      ;SPACE RESERVED FOR EXPANSION
02C4    00 00 00 00             DB 0,0,0,0      ;SPACE RESERVED FOR EXPANSION
02C8    00 00 00 00             DB 0,0,0,0      ;SPACE RESERVED FOR EXPANSION
02CC    00 00 00 00             DB 0,0,0,0      ;SPACE RESERVED FOR EXPANSION
02D0    00 00 00 00             DB 0,0,0,0      ;SPACE RESERVED FOR EXPANSION
02D4    00 00 00 00             DB 0,0,0,0      ;SPACE RESERVED FOR EXPANSION
02D8    00 00 00 00             DB 0,0,0,0      ;SPACE RESERVED FOR EXPANSION
```

```
                                ;
                                ; SPACE FOR USER-ADDED SUBROUTINES
                                ;
02DC    00 00 00 00             MORPAT: DB 0,0,0,0      ;ADDITIONAL SPACE THAT MAY
02E0    00 00 00 00                     DB 0,0,0,0      ;BE USED FOR USER PATCHES,
02E4    00 00 00 00                     DB 0,0,0,0      ;FOR EXAMPLE FOR A CURSOR
02E8    00 00 00 00                     DB 0,0,0,0      ;POSITIONING SUBROUTINE, OR
02EC    00 00 00 00                     DB 0,0,0,0      ;FOR "INISUB" OR "UNISUB"
02F0    00 00 00 00                     DB 0,0,0,0      ;SUBROUTINES.
02F4    00 00 00 00                     DB 0,0,0,0      ;FOR YET MORE SPACE,
02F8    00 00 00 00                     DB 0,0,0,0      ;SPACE, SEE "PBGMEM", NEXT.
02FC    00 00 00 00                     DB 0,0,0,0
0300    00 00 00 00                     DB 0,0,0,0
0304    00 00 00 00                     DB 0,0,0,0
0308    00 00 00 00                     DB 0,0,0,0
030C    00 00 00 00                     DB 0,0,0,0
0310    00 00 00 00                     DB 0,0,0,0
0314    00 00 00 00                     DB 0,0,0,0
0318    00 00 00 00                     DB 0,0,0,0
031C    00 00 00 00                     DB 0,0,0,0
0320    00 00 00 00                     DB 0,0,0,0
0324    00 00 00 00                     DB 0,0,0,0
0328    00 00                           DB 0,0

                                ;TEXT POINTS TO THE LOCATION OF THE MESSAGE AREA.  IF
                                ;THE MESSAGE FILE HAS BEEN ALTERED, THE NEW MESSAGE FILE
                                ;SHOULD BE PATCHED AT THIS ADDRESS.  IF THE LENGTH OF THE
                                ;MESSAGE FILE IS DIFFERENT, PBGMEM MUST BE ADJUSTED.  THE
                                ;FIRST TWO BYTES OF THE MESSAGE FILE GIVE THE NEW LENGTH.
                                ;THIS QUANTITY ADDED TO THE ADDRESS AT 'TEXT' IS THE
                                ;VALUE THAT SHOULD BE PATCHED AT PBGMEM.

032A    0000*                   TEXT:   DW MSTART       ;MESSAGE AREA START

                                ;PBGMEM POINTS TO BEGINNING OF MEMORY TO USE FOR WORK-
                                ;ING STORAGE. IF YET MORE SPACE IS NEEDED FOR PATCHES,
                                ;INSTALL FIRST, THEN PUT YOUR ADDED CODE WHERE PBGMEM
                                ;POINTS AND INCREASE PBGMEM TO POINT BEYOND YOUR PATCHES.
                                ;BE SURE TO USE A LARGE ENOUGH "SAVE" COMMAND!

032C    0000*                   PBGMEM: DW MEMORY       ;An indefinite amount of code may
                                                        ;be added here AFTER INSTALLation.
                                                ;Do not add code before INSTALLing,
                                                ;as INSTALL will delete it!

                                ;
                                ; INITIAL HELP LEVEL FOR FORMGEN
                                ;
032E    04                      ITHELP: DB 4            ;1 = NO HELP MESSAGE
                                                        ;2 = ONLY SCREEN TWO
                                                        ;3 = INCLUDE HELP SCREENS 2 & 3
                                                        ;4 = ALL MESSAGES INCLUDED
```

```
                        ;
                        ; UNUSED
                        ;
032F    00              DB 0

                        ; ** END TERMINAL MODIFICATION AREA **

0330                    ALABEL  EQU     $       ;ASSEMBLER SHOULD PRINT 330
                        END     ;NEXT MODULE LOADS AT LOCATION 330 HEX
```

Macros:

Symbols:

ALABEL	0330	CB4LFG	0260I	CLEAD1	024DI	CLEAD2	0256I
CLRSCR	0242I	COLOFF	0262I	CRBLIV	02B7I	CTRAIL	025BI
DELCLR	02AFI	DELCUS	02B0I	DELMIS	02B1I	ERAEOL	0270I
HIBCUR	02B6I	HIBIV	02B5I	HITE	0240I	INISUB	02A7I
ITHELP	032EI	IVOFF	028EI	IVON	0287I	LINOFF	0261I
MEMADR	02B3I	MEMAPV	02B2I	MEMORY	032C*	MORPAT	02DCI
MSTART	032A*	OUTCHR	026A*	PBGMEM	032CI	TBASE	0100
TEXT	032A	TRMINI	0295I	TRMUNI	029EI	UCRPOS	0267I
UNISUB	02AAI	USELST	02ADI	WID	0241I		

No Fatal error(s)

Appendix D

FORMGEN REQUIREMENTS

A minimum of 48K is required in order to run FormGen. This amount should be sufficient for most forms. Very large form designs may require greater amounts of memory. A general rule is that if the form is near maximum size when it is designed through FormGen, the resulting form will probably work properly when used for data entry through DataStar.

The operating system will require 6-8K of memory, while the FormGen program will require 34K. The remaining memory on the system will be used to define the form. In systems with 48K of memory, this will amount to an additional 6-8K, depending upon your system; systems with 64K, 22-24K will be available for form definition.

Memory is used as follows in the form definition process:

1	byte for every character in the form image (multiply form height by form width).
18	bytes for each datafield.
2*	field length for each range-checked datafield.
2*	field length for each edit-masked datafield.
12	bytes for each list verified datafield.
??	A variable amount will be required for each calculated datafield. The requirement is approximated by the number of characters in the expression.

For example: In a 48K system, a form with 50 lines, 100 columns wide and 100 datafields would use 7000 bytes of memory. This would fit with a 6K operating system, leaving 1000 bytes free for calculations, range checks and/or edit masks.

DataStar Requirements

The memory requirements for the DataStar program are as follows:

6-8K for the operating system

26K for the DataStar program

(27K for the DataStar program if batch mode is selected.)

Note that the DataStar program is smaller than the FormGen program by 7-8K (depending on batch mode usage). It would appear, therefore, that any form that fit in FormGen would likewise work in DataStar. This is usually the case; however, it is possible to design a form that can't run under DataStar. This is because DataStar requires additional memory for every list verified file in the form. The space required is the size of the list verify file if it is to be kept in memory, or 1/2K if the file is not read into memory. There is room for 14-16 external reference files maximum with a form that uses the maximum amount of memory in the design process. The important point is that if your form is at or near the FormGen memory limits, it won't all fit in DataStar if you have more than 16 file references for files larger than 500 characters.

The final consideration about memory requirements is that the FormGen and DataStar program sizes are subject to change. In order to stay compatible with future versions, you may want to keep somewhat below the absolute maximum form size.

APPENDIX E

ASCII CODES

ASCII	DECIMAL	HEXA DECIMAL	ASCII CODE	
^@	0	00H	"NUL"	Null Character
^A	1	01H	"SOH"	Start of Heading
^B	2	02H	"STX"	Start of Text
^C	3	03H	"ETX"	End of Text
^D	4	04H	"EOT"	End of Transmission
^E	5	05H	"ENQ"	Enquiry
^F	6	06H	"ACK"	Acknowledge
^G	7	07H	"BEL"	Bell
^H	8	08H	"BS"	Backspace
^I	9	09H	"HT"	Horizontal Tabulation
^J	10	0AH	"LF"	Line Feed
^K	11	0BH	"VT"	Vertical Tabulation
^L	12	0CH	"FF"	Form Feed
^M	13	0DH	"CR"	Carriage Return
^N	14	0EH	"SO"	Shift Out
^O	15	0FH	"SI"	Shift In
^P	16	10H	"DLE"	Data Link Escape
^Q	17	11H	"DC1"	Device Control 1
^R	18	12H	"DC2"	Device Control 2
^S	19	13H	"DC3"	Device Control 3
^T	20	14H	"DC4"	Device Control 4
^U	21	15H	"NAK"	Negative Acknowledge
^V	22	16H	"SYN"	Synchronus Idle
^W	23	17H	"ETB"	End of Transmission Block
^X	24	18H	"CAN"	Cancel
^Y	25	19H	"EM"	End of Medium
^Z	26	1AH	"SUB"	Substitute
^[	27	1BH	"ESC"	Escape
^\	28	1CH	"FS"	File Separator
^]	29	1DH	"GS"	Group Separator
^^	30	1EH	"RS"	Record Separator
^_	31	1FH	"US"	Unit Separator
	32	20H	"SP"	Space

ASCII	DECIMAL	HEXA DECIMAL
!	33	21H
"	34	22H
#	35	23H
$	36	24H
%	37	25H
&	38	26H
'	39	27H
(	40	28H
)	41	29H
*	42	2AH
+	43	2BH
,	44	2CH
-	45	2DH
.	46	2EH
/	47	2FH
0	48	30H
1	49	31H
2	50	32H
3	51	33H
4	52	34H
5	53	35H
6	54	36H
7	55	37H
8	56	38H
9	57	39H
:	58	3AH
;	59	3BH
<	60	3CH
=	61	3DH
>	62	3EH
?	63	3FH
	64	40H
A	65	41H
B	66	42H
C	67	43H
D	68	44H
E	69	45H
F	70	46H
G	71	47H
H	72	48H
I	73	49H
J	74	4AH
K	75	4BH
L	76	4CH
M	77	4DH
N	78	4EH
O	79	4FH

ASCII	DECIMAL	HEXA DECIMAL
P	80	50H
Q	81	51H
R	82	52H
S	83	53H
T	84	54H
U	85	55H
V	86	56H
W	87	57H
X	88	58H
Y	89	59H
Z	90	5AH
[	91	5BH
\	92	5CH
]	93	5DH
^	94	5EH
-	95	5FH
`	96	60H
a	97	61H
b	98	62H
c	99	63H
d	100	64H
e	101	65H
f	102	66H
g	103	67H
h	104	68H
i	105	69H
j	106	6AH
k	107	6BH
l	108	6CH
m	109	6DH
n	110	6EH
o	111	6FH
p	112	70H
q	113	71H
r	114	72H
s	115	73H
t	116	74H
u	117	75H
v	118	76H
w	119	77H
x	120	78H
y	121	79H
z	122	7AH
{	123	7BH
|	124	7CH
}	125	7DH
~	126	7EH
DEL	127	7FH

DATASTAR
Version 1.4

ADDENDUM

The new DataStar 1.4 release has several innovative features that will make DataStar even more flexible and useful. You now have the option of creating intermediate fields and assigning field names to facilitate the handling of large forms and complex field calculations. Additionally, the new DataStar Training Guide and reorganized Help Menus make DataStar easier to learn and use.

Here's what these features do and how they affect other parts of the DataStar program.

FIELD NAMES

This attribute is now presented as the first prompt and help message (shown below) on the FormGen Attribute List. To select a field name you will need to type in the name you want to assign. If the field is not named and you do not wish to name it, simply press RETURN to proceed to the next prompt.

> *Field name:*
>
> You have entered the field definition phase by typing ^R. The field definition phase is a questionnaire which allows you to set up controls over what data will be accepted during the data entry portion of this program. In the questions that follow, the answers you give will apply to the current field only. To return to background definition, type ^C.
>
> Field names are optional. The field name may contain from 1 to 32 characters and must begin with a letter. The remaining characters may be only *letters, numerals,* or *spaces.*

Don't use the same field name twice. If you do, DataStar will present this error message:

> **Duplicate Field Name. Hit Escape Key.**

THE ADDITION OF FIELD NAMES WILL HAVE THE FOLLOWING EFFECTS:

1. The field name, if assigned, will then be used in the field calculation attribute listing on page 5 of the FormGen printout (shown below).

   #005 = PREVIOUS BALANCE + CHARGED

2. In FormGen, the line buffer command will copy a line to a new location on your form, but the field names will be deleted from that line. This feature eliminates the possibility of duplicating field names.

3. When you assign the third attribute, Copy Attributes of Field, the field name will *NOT* be copied. FormGen will not create a duplicate name.

4. If a field is split, after you have assigned a name to that field, the second field (created from the split) will not have a name. Additionally, if two fields are joined, the second field loses its name (even if the first field has no name).

5. Two FormGen Error Messages (page 2-38 of DataStar User's Guide 1.1) will change to reflect the addition of field names. These revised messages are shown below:

   Illegal operand. Item must be
   a field between #001 and #_____,
   a literal enclosed in quotes, or
   a valid field name. Hit ESC key:

   Illegal operand. Item must be
   a field between #001 and #_____, or
   a constant using only the digits 0 through 9, an optional decimal point and leading minus sign, or a valid field name. Hit ESC key.

6. If you have used a previous version of FormGen to create a form without field names, that form will be usable with DataStar 1.4 *unless that form fits the computer's memory completely* and thus allows no memory space for the addition of field names. Field names require a minimum of 1 byte per field (even if no name is assigned).

7. The field name will be listed on Part 4 of the FormGen printout, as this excerpt shows.

```
                    FIELD ATTRIBUTE DEFINITIONS

                  Q=required
                  C=Check dgt
                  J=Right just       * * D E R I V E D *
                  W=Write ed c       *     LIST  CALC  ******VERIFICATION****
                  O=Oper entry       *                 *
FIELD  NUM/NAME R=Range chk  PAD/ * INDEX ITEM      *          LIST VERIFY
LEN LIN COL KEY E=edit mask FLOAT * FIELD NUM ORDER *   ORDER  FILE NAME

001/CUSTOMER
    000 009 001 .    JWO E . PO    .          001 N.
002/STREET
    000 036     . Q JW RE . PO    .                .

                                2
```

8. Field names can be used instead of field numbers when assigning the calculation attribute.

INTERMEDIATE FIELDS

You can now create intermediate fields during the FormGen field definition (attribute) phase. The prompt and help messages are shown below:

Intermediate field:

An intermediate field can keep redundant data from being stored on the disk or hold partial results in complicated calculations. An intermediate field is like any other field except that it is absent from the record that gets stored on the disk.

A Y answer here will designate this field as intermediate.

You must assign the following attributes to an intermeditate field:

- Derived.
- Derived by either File (F) or Calculated (C). See Appendix A for more information on File Derived (formally called List) and Calculated fields.
- Intermediate field attribute.

Remember these facts about intermediate fields:

- You cannot assign the Intermediate Field Attribute to either a Key Field or a self-referenced field.
- You can use intermediate fields for subtotaling if you have a calculation with many steps. DataStar will print or display the subtotals, but will not store these totals on the disk record.
- You cannot assign any Verify Attribute for intermediate fields.
- You cannot assign Operator Entry Attribute for intermediate fields.

OTHER CHANGES/ADDITIONS

1. MEMORY REQUIREMENTS

FormGen's memory requirements have not changed. DataStar's new memory requirements are listed below.

DATASTAR.COM = 26 K (total)
FORMGEN.COM = 35 K (total)

You will also need memory for the form definition file and datafiles. (DataStar's file limit is 8 megabytes.)

2. FORMGEN FILE EXTENSION

The extension .DEF has been added to FormGen file names. If you have old form files without this extension, these files will not be compatible with DataStar 1.4 unless you add the .DEF extension.

Add the extension to your form file name by using your operating system's RENAME command. In CP/M you would use the following command if your old file name was ORDER:

```
A>REN ORDER.DEF = ORDER
```

or

```
A>ren order.def = order,
```

In DataStar's Add and Scan Modes, the word RUB has been replaced by DEL (for delete). After you have edited a record, you can use ^Y as well as DEL to delete your record.

3. NEW INSTALL OPTIONS

a. These terminals are now supported:

******* DataStar TERMINAL MENU #1 *******

A	Lear-Siegler ADM-3A	C	Lear-Siegler ADM-31
D	Hazeltine 1500	E	Microterm ACT-IV
F	Beehive 150/Cromemco 310	G	Imsai VIO
H	Hewlett-Packard 2621 A/P	I	Infoton I-100
K	Soroc IQ-120	L	Perkin-Elmer 550 (Bantam)
M	Microterm ACT-V	N	TeleVideo 912/920
2	Terminal Menu #2	3	Terminal Menu #3
U	no change		

******* DataStar TERMINAL MENU #2 *******

0	Visual 200	P	DEC VT100/ANSI Terminal
Q	Vector Flashwriter II	S	Compucolor 8001G
V	TEC Model 571	W	Xerox 820
=	Intertec Superbrain	!	Sanyo MBC-2000/3000
)	NorthStar Advantage	+	TRS-80/Lifeboat 2.24 CP/M
Y	TeleVideo 910	%	TRS-80/Lifeboat 2.25 CP/M
1	Original Menu	3	Terminal Menu #3

******* DataStar TERMINAL MENU #3 *******

1	Original Menu	2	Terminal Menu #2
@	Hewlett-Packard 87	#	Hewlett-Packard 125
<	ADDS Regent 20/25	>	ADDS Regent 40/45/60/45
$	IBM 3101	(	Hazeltine 1420
&	TeleVideo 950	]	ADDS Viewpoint
X	Heath or Zenith H89/H19	/	Osborne 1

b. **IS THIS INSTALLATION FOR MP/M? (Y/N)**

(Note: This version of DataStar works under CP/M only. The appropriate response is a N.)

c. **DISABLE LINE FEEDS TO PRINTER? (Y/N)**

If you have a printer that automatically provides a line feed with each carriage return, type Y; if not, type N.

4. TERMINAL PATCH CHANGES

To establish whether the cursor positioning control sequence patch will be expressed in ASCII (two or three digit) or binary numbers, the following section of code has been added to the Terminal Patch Area, Appendix E, page 4:

Insert the information below for byte 0263

```
0263  00 ASCUR  DB 0     ;ZERO TO SEND BINARY LINE & COLUMN
                         ;2 TO SEND 2-DIGIT ASCII NUMBERS
                         ;3 TO SEND 3-DIGIT ASCII NUMBERS

                         ;SEE NEXT PAGE FOR POSITIONING
                         ;CURSOR VIA USER-CODED SUBROUTINE
                         ;CURSOR POSITIONING...
                         ;Add RSTFLAG
                         ;Add LFFLAG
```

5. BATCH FILE CONSIDERATIONS

a. Batch File Renamed

BATCH.COM is now BATCH.OVR in DS 1.4. Replace (or rename) your old batch file with BATCH.OVR.

b. Batch Verifying *Very Large* Files

At some point in the batch verify procedure for extremely large files, DataStar may present the following message:

> **End of session. All work saved, but re-start is necessary to continue. (If disk is full, first remove something.) Press RETURN.**

If you get this message simply do the following.

PRESS **RETURN**

Your system prompt will appear. Reinvoke DataStar and the *same* form name you have been using.

TYPE **DataStar** (form name)
PRESS **V** (for batch verify)
TYPE the same disk drive you have been using
TYPE the same batch file name you have been using

DataStar will continue to batch verify your records from where it left off.

6. ATTRIBUTES

a. Edit Mask, Entry Control Character Codes:
2 changes to Appendix A, Attributes, pg. A-37
/ = constant if data on *either side*
. = align decimal point

b. List Derived Fields:
The title of the List Derived Attribute has been changed to File Derived.

c. Calculated Fields:
You can design calculations up to 255 characters long, and more than one line. The calculation listing, page 5 on the form attribute printout, will now list more than one line of calculations when assigned.

d. Copy Previous Record:
Do not use the Copy Previous Record (^C) if the pad or float characters in the record conflict with the Content Control Characters.

e. Range Check:
When assigning the range check, enter the number for *both* the minimum and maximum ranges (as they would appear when entered into DataStar) and follow these guidelines.

- if the field is right justified put the range value on the *right* side of the field.
- if the field is decimal aligned, put the decimal point in the correct position.
- if the field contains a floating character, place that character in the correct position.
- if the field contains pad characters, place these characters in the correct positions.

Examples

For a nine character right justified field containing a floating character:

Minimum Range	Maximum Range
_____$10.00	$1,000.00

For a five character right justified field padded with zeros:

Minimun Range	Maximum Range
00010	01000

7. EDIT SCAN MASK

The procedure for using the Edit Scan Mask with *right justified fields* has been changed. Use the following examples to *replace* the information on pg 4-9.

When searching records, DataStar compares fields in the following way:

Right Justified Fields = right to left

Left Justified Fields = left to right

Decimal Aligned Fields = either of the above depending on the justify assignment.

Example:

For a four character, right justified field, a Scan Mask with a 2 in the first position should appear as follows:

```
2***
```

DataStar will then retrieve all records from 2000-2999.
If the above example had a 2 in the *last* position:

```
***2
```

DataStar will then select all records that *end* in 2.

If you set up a Mask for a right justified field with a 2 in the second position:

```
*2**
```

DataStar will retrieve all records that have a 2 in this position (e.g., 200-299, 1200-1299, 2200-2299...).

If you want only records in the 200 range, enter this Mask for right justified fields:

02**

DataStar will retrieve all records from 200-299.

Note: Left justified decimal aligned fields will continue to work as described on pg 4-9.

8. **FILE MAINTENANCE**

The following options have been added to the file maintenance procedure:

a. To abort file maintenance

PRESS **CTRL E**

b. To stop record display, hit any key (other than ^E). The last record displayed will stay on your screen while file maintenance continues. Hit any key and the record display will resume. NOT displaying records speeds up file maintenance.

9. **DTB/NDY FILES**

DTB and NDY files reserve space in the directory. If this space is required during the course of DataStar operations, .DTB and .NDY will be assigned to your file name. If you find these file names in your disk directory, just erase them.

10. **FORMGEN HELP MESSAGES**

The attribute help messages in FormGen have been rewritten and the new messages are listed below. Remember, list derived fields are now called File Derived fields.

Field name:

You have entered the field definition phase by typing a ^R. The field definition phase is a questionnaire which allows you to set up controls over what data will be accepted during the data entry portion of this program. In the questions that follow, the answers you give will apply to the current field only. To return to background definition, type ^C.

Field names are optional. The field name may contain from 1 to 32 characters and must begin with a letter. The remaining characters may be letters, numerals, or spaces.

Field order:

This attribute allows you to specify the order in which fields will be processed during the data entry phase.

Key order:

The key fields are the fields on which the index file is sorted. These fields are combined for sorting. The field with the key order #001 will be the most significant field in the sort.

Tie breaker field? (Y/N)

A tie breaker field is a key field entered by DataStar to create a unique key.

Enter a Y here to have DataStar maintain the file by entering the lowest numeric value here that will create a unique key whenever necessary.

Refuse duplicate keys? (Y/N)

This attribute is used to create files with unique keys. If the data entry operator attempts to enter a record with the same key as a record already in the file, an error message will be generated and the entry refused.

Copy attributes of field:

This attribute is used to save time when defining fields. If another field has been defined with the desired attributes, entering its field number here will cause its attributes to be used by this field also.

Field derived? (Y/N)

A field may be derived from other fields or from a file. The cursor will not stop at a derived field during data entry.

A Y answer here will make this a derived field. You will define how data is to be derived for this field in the following prompts.

Allow operator entry? (Y/N)

A Y answer here will allow the operator to make adjustments to the data in this field. Otherwise, the cursor will not enter this field. Use ^A to back your cursor into an operator entry field. If the operator entry field is the last field on the form, use ^L to move your cursor into it.

Calculated/File? (C/F)

A C answer here will make this a calculated field. Data for a calculated field is computed from constants and other fields on the form. You may define the calculation in the following prompt.

An F answer here will make this a file derived field. Data for a file derived field is copied from a reference file. Another field on this form is used as an index into that file. For example, to derive vendor address for this field from a file of vendor names .cp2 and addresses, you could use vendor name (entered elsewhere on this form) as the index field.

Index field number:

Enter the number of the field on this form that you want to be used as an index into the reference file. For the example in the previous prompt, you would enter the field number for vendor name. If you don't remember the number of the index field, use ^A or ^F to move to the index field. Its number will be displayed on the status line.

Item number in file:

This attribute designates the item (data) to be copied from the reference file to this field. For the example in the previous two prompts, if the vendor file has "name, address, city, state" for each vendor, you would enter "003" to extract city.

Verify/calculate order:

This attribute allows you to specify the verify/calculate sequence. In general, calculations should follow verifications unless the result of a calculation is being used as an index field. Fields can be calculated in any order. The calculation sequence will only make a difference when one calculation depends on the results of another.

The field with verify/calculate order #001 will be processed first.

Numeric/String? (N/S)

A numeric calculated field uses an algebraic expression to derive a value for the field. The algebraic expression consists of fields and numeric constants combined arithmetically, using the operations:

add(+), subtract(-), multiply(*), divide(/), exponentiation(^)

A string expression consists of fields, sub-fields and string constants combined using the join (&) operation.

Enter string expression for field:

A string expression may contain fields, subfields or string constants (literals) combined using the join (&) operation. Fields are represented by either field number preceded by a "#", or field name. For example, #007 or FIELDNAME. Subfields are used to extract only a portion of a field. The form is: field number (first character number, number of characters). For example, #7(1,2) or fieldname(1,2). Literals are represented by enclosing the characters in double quotation marks. For example, "A01".

Enter algebraic expression for field calculation:

An algebraic expression may contain fields and numeric constants combined with the arithmetic operations: add(+), subtract(-), multiply(*), divide(/) and exponentiation(^). Operations are performed from left to right in the usual manner with parentheses recognized. Fields are represented by either field number preceded by a "#", or field name. For example, #007 or FIELDNAME. All constants and fields should contain only digits and optional minus sign and decimal point.

Intermediate field? (Y/N)

An intermediate field can keep redundant data from being stored on the disk or hold partial results in complicated calculations. An intermediate field is like any other field except that it is absent from the record that gets stored on the disk.

A Y answer here will designate this field as intermediate.

Required? (Y/N)

This attribute determines if data must be entered in this field during data entry. A Y answer here will force the operator to make an entry.

Right justify? (Y/N)

This attribute determines on which side of the field the data will be placed if it does not fill the field.

A Y answer here will align the data with the right side of the field.

Pad field? (Y/N)

A field can be extended to its full length by adding pad characters on the left of the data, for right justified fields, or vice versa.

A Y answer here will pad the field with pad characters if it is not full. The pad character will be specified in the next prompt.

Enter pad character:

Any character may be used to extend the field to its maximum size. Commonly used characters are space and zero.

Floating character? (Y/N)

A "floating character" can be inserted into the field, either on the left of the data for right justified fields or vice versa. For example if '$' is used as a floating character for a right justified field, a dollar sign will be added to the left of the data entered.

A Y answer will provide a floating character in this field. The floating character will be specified in the next prompt.

Enter floating character:

Any character may be used as a floating character. Commonly used characters are '$', and ' + ' or '-'.

Verify field? (Y/N)

After the data entry for a form is complete, DataStar will enter a verify phase. In this phase all of the fields with the verify attribute will be checked for accuracy.

You cannot assign the verify attribute to intermediate fields. A Y answer here will cause this field to be checked during the verify phase. The kind of verification will be specified in the next prompt.

Sight/Retype/File? (S/R/F)

Sight verify means the cursor will move to this field and allow the operator to take another look at the data.

Retype verify means the field will be cleared before the cursor moves to it and the operator will be forced to type the same data again.

File verify means the field will be checked against a reference file. File verify will be further specified in the following prompts.

Keep file in memory? (Y/N)

If the reference file is small enough (fewer than 500 characters), it will save space to keep it in memory during data entry. It will always speed processing to keep it in memory.

A Y answer here will cause the reference file to be kept in memory.

Enter file name:

Enter the name of the reference file that contains the list of acceptable entries for this field.

Enter file disk drive (-/A/B...):

Enter "A-P" to specify the disk drive that holds the reference file. Enter "-" to use the current drive. In both cases, the operator will be asked to specify another disk drive if the file is not found.

Enter file key field number:

This attribute specifies the field in the reference file that DataStar will search to verify or match the data in this field. The field used must be the key field and be exactly the same length as this field.

Enter the field number of the key field in the reference file.

Batch verify? (Y/N)

This attribute determines if the records created in DataStar will be verified immediately after data entry or later as a group.

A Y answer here means the records will be verified later as a group.

Check digit? (Y/N)

A check digit is a digit added on to the end of a numeric field to make data obey some rule. The rule in DataStar is that the field must be divisible by eleven. A check digit is used to improve the accuracy of entered data, since incorrect data is not likely to obey the rule.

Enter a Y to require that the data in this field be divisible by 11.

Range check? (Y/N)

This attribute is used to assure that only data within a certain range will be entered in this field. As an example, the two character field called month can only have values between 01 and 12.

A Y answer here will allow you to specify the minimum and maximum values for this field in the following two prompts.

Enter/change the minimum field value:

A field is compared on a character by character basis to determine acceptablity. Characters are compared according to the following sequence:

space !"#$%&''()* + ,-./0-9:;< = >?@A-Z[\]^_'a-z{|}~ (highest)

Enter the smallest acceptable value for this field.

Enter/change the maximum field value:

A field is compared on a character by character basis to determine acceptablity. Characters are compared according to the following sequence:

space !"#$%&''()* + ,-./0-9:; < = >?@A-Z[\]^_'a-z{|}~ (highest)

Enter the largest acceptable value for this field.

Edit mask? (Y/N)

The edit mask allows you to control the entry and content of a field on a character by character basis. This includes inserting constants, copying data from the previous form, requiring entry at certain positions, as well as providing control over the actual characters that may be entered at each position.

You may *not* copy data from a previous form (selection X and Y below) if the field is derived.

A Y answer will allow you to define the edit mask in two steps: entry and content control.

Entry Control Character Codes:

! = must enter a character	_ = may enter or leave blank
X = automatic copy	Y = auto-copy/may enter
" = constant in this position	~ = constant/may enter
' = constant if data on both sides of constant	/ = constant if data on either side
. = align decimal point	

Enter/change the entry control mask:

This attribute defines the entry control mask. This mask allows you to specify if data must be entered or how it is entered for each field position. Each character in the field may be entered by the operator, copied from the same position and field of the previous record, or be a constant. Do not use X or Y for automatic copy in derived fields. You will get the error message:

Illegal character.
Press escape key.

Constants will be specified in the next prompt.

Content Control Character Codes:

A = A-Z only	a = a-z only
B = A-Z, space	b = a-z, space
C = A-Z, a-z -> A-Z	c = A-Z, a-z
D = A-Z, a-z -> A-Z, space	d = A-Z, a-z, space
E = A-Z, 0-9	e = a-z, 0-9
F = A-Z, 0-9, space	f = a-z, 0-9, space
G = A-Z, 0-9, a-z -> A-Z	g = A-Z, 0-9, a-z
H = same as G with space	h = A-Z, 0-9, a-z, space
9 = 0-9 only	8 = 0-9, #$%()* + -.
. = align decimal point	_ = any character allowed

Enter/change the content control mask:

This attribute defines the content control mask. This mask allows you to specify the constant or type of data to be entered in each field position. For each position specified as a constant in the entry control mask (see the previous prompt), enter the desired constant in the same position in this mask.

Record edit characters? (Y/N)

Edit characters are constants and pad/float characters in the field. Usually they are removed from the field before the data is recorded.

A Y answer here will include the edit characters in the disk record.