VAX TDMS Forms Manual

Order No. AA-GS13B-TE

August 1986

This manual describes the use of the TDMS Form Definition Utility (FDU), including the use of the form editor.

OPERATING SYSTEM:	**VMS**
	MicroVMS
SOFTWARE VERSION:	**TDMS V1.6**

digital equipment corporation, maynard, massachusetts

Contents

Tables

How to Use This Manual

This manual describes how to create and use forms using the VAX Terminal Data Management System (TDMS). The TDMS software is also referred to in this manual simply as TDMS. In particular, this manual discusses the TDMS Form Definition Utility (FDU), which is used to create, modify, and store forms.

The VAX DATATRIEVE software is also referred to as DATATRIEVE in this manual.

Intended Audience

You should use this manual if you want to:

- Understand the characteristics of TDMS forms and the features available to you in designing forms
- Learn how to use the Form Definition Utility to create forms for use in a TDMS application
- Learn how to use the Form Definition Utility to store and retrieve forms
- Learn how to create a simple TDMS application
- Learn the general concepts of forms and requests
- Learn how to run the TDMS samples optionally installed with the TDMS software.

In order to use this manual to create forms for a TDMS application, you should have at least an elementary understanding of the following:

- The VMS operating system
- The use of VT100 or VT200 family of terminals
- Data management concepts (records, files, data types)
- Programming concepts

Structure

The first eight chapters of this manual describe how to use FDU and its form editor to create forms for use in a TDMS application. Chapter 2 walks you through a simple TDMS application. Chapter 9 discusses how to use VAX TDMS with VAX DATATRIEVE.

Chapter 1	Explains the use of forms in a TDMS application.
Chapter 2	Provides a step-by-step walkthrough that creates a simple TDMS application.
Chapter 3	Discusses the primary uses of the Form Definition Utility and provides an overview of the form editor.
Chapters 4-8	Provides a detailed guide to using the form editor.
Chapter 9	Provides information on how to use VAX TDMS with VAX DATATRIEVE.

Related Manuals

As you use this book, you may find the following manuals helpful:

VAX TDMS Request and Programming Manual

VAX TDMS Reference Manual

VAX/VMS DCL Dictionary

VAX Common Data Dictionary Utilities Reference Manual

VAX Common Data Dictionary Data Definition Language Reference Manual

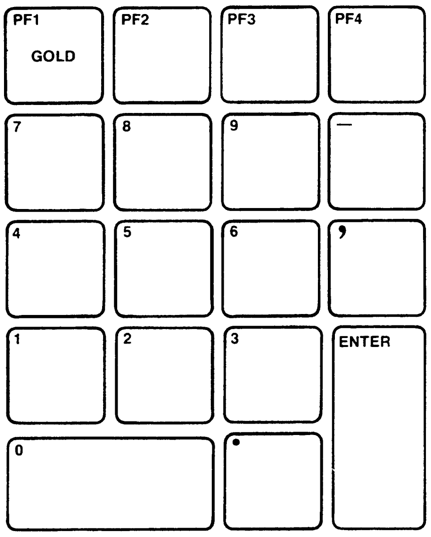
VAX Run-Time Library Reference Manual

Operating System Information

To verify which versions of your operating system are compatible with this version of VAX TDMS, check the most recent copy of the *VAX System Software Order Table/Optional Software Cross Reference Table*, SPD 28.98.xx.

Conventions

The VT100 and VT200 keyboard includes a main keyboard and an auxiliary keypad. The keys on the keypad (KP) are designated by the names shown in the following diagram:



ZK-00094-00

This section explains the special symbols used in this book:

FDU>	The FDU> prompt indicates the utility is at command level and ready to accept FDU commands.
RDU>	The RDU> prompt indicates the utility is at command level and ready to accept RDU commands.
RDUDFN>	The RDUDFN> prompt indicates that the RDU utility is at the instruction level and ready to accept request or request library instructions.
$	The dollar sign prompt indicates that you are at DIGITAL Command Language (DCL) level. (It is possible to change the DCL prompt. However, in this manual the examples use the default prompt, the dollar sign.)

<CTRL/x>	This symbol tells you press both the CTRL (control) key and a specified letter key simultaneously.
<GOLD-x>	This symbol indicates that you press the GOLD key and then a specified letter key consecutively.
Color	Colored text in examples shows what you enter.
<RET>	This symbol indicates the RETURN key. Unless otherwise stated, end all user input lines in examples by pressing the RETURN key.

Technical Changes and New Features

This section summarizes the changes to VAX TDMS that are described in this manual.

- Characters from the DEC Multinational Character Set are now valid for background text and range and choice validators. At run time, DEC Multinational characters are valid for the X, A, and C picture characters. Choice and range string validation is performed using the DEC Multinational Character Set.
- TDMS has a predefined set of run-time function keys that operators can use to perform various operations on the screen such as moving from field to field, refreshing the screen, and getting help. These predefined keys are listed in Table 11-1 in the *VAX TDMS Request and Programming Manual* in the chapter called Program Request Keys.
- FDU can now be used with a VT200 terminal set to VT200 terminal mode. The following function keys from the LK201 keyboard used by VT200-series terminals are supported:

  - The F12 (BS) key performs the BACK SPACE key function.
  - The F13 (LF) key performs the LINE FEED key function.
  - The HELP key performs the PF2 or HELP key function.

  In addition, the operator can use the following keys on the LK201 keyboard in the form editor:

  - The SELECT key performs the KP-period (SELECT) key function in the form editor

  - The REMOVE key performs the KP6 (CUT) key function in the form editor
  - The INSERT key performs the GOLD-KP6 (PASTE) function in the form editor

  Information about these keys has been included where appropriate throughout the documentation set.

- A date form field of spaces is now considered a valid date by TDMS. When mapping a date from a date form field to a record field with the data type ADT, TDMS stores the base system date, Nov 17, 1858. When outputting a record field of data type ADT to a date form field, TDMS displays the base system date as spaces instead of Nov 17, 1858.

  Spaces input from a date form field to a field of data type TEXT are stored and displayed as spaces. Any valid TDMS date input from a form field to a record field of data type TEXT, including Nov 17, 1858, are stored and displayed as the text entered.

- TDMS no longer assumes the base year of 1900 when mapping between record fields with the ADT data type and two-digit year date form fields. The last two digits of the year are used for output mapping. For input mapping, TDMS attempts to use the century in your record. If there in no date in your record, or if using the century in your record would result in an invalid date, TDMS uses the current century.

- In addition to its regular date formats, TDMS now includes the standard VMS date format: DD-MMM-YYYY.

- The LIST FORM command in FDU now lists all field attributes, including Must Fill and Response Required.

- In earlier versions of TDMS, a scrolled region whose first field was Display Only was placed at the end of the order list. TDMS now correctly orders this type of scrolled region.

- The default listing file name now contains up to 39 characters and can include the dollar sign ($) and underscore (_). This file name is generated in FDU by use of the /OUTPUT qualifier and in RDU by use of the /OUTPUT and /LIST qualifiers. The name is not truncated and includes any dollar signs and underscores.

Introduction to VAX TDMS 1

VAX Terminal Data Management System (TDMS) is an information management product that lets you use forms to collect and display information on a terminal. It offers a wide range of features that make it easy to display and collect information and to realize significant savings in developing and maintaining an application.

This chapter discusses:

- The parts of TDMS, including form and record definitions, requests, and request library definitions
- The sequence of events in designing a TDMS application
- How to run the sample applications

1.1 A VAX TDMS Form Definition

A **form definition** describes a form, which can be displayed on a terminal at run time in a TDMS application. You create a form definition using the **Form Definition Utility (FDU)**.

When you create a form definition using the Form Definition Utility, FDU stores the definition in a central storage facility, the VAX Common Data Dictionary (CDD).

1.2 Using a Form Definition

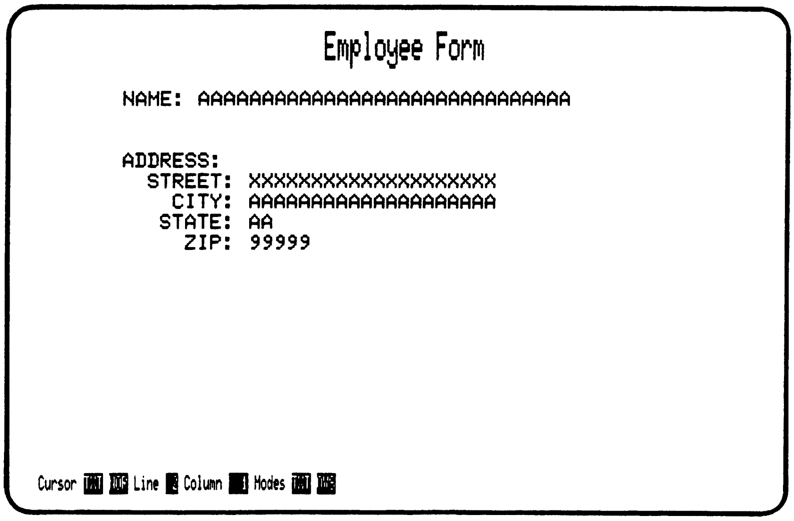
In order to use a form definition in a TDMS application to collect or display data, you must first do the following:

- Create a form and store it in the CDD.
- Create one or more record definition(s) and store them in the CDD. Record definitions define the data type, structure, and length of the data used in an application.
- Create a request that:
  - Identifies the form definition and contains an instruction to display it.
  - Identifies the record definition.
  - Contains instructions that allow data to be entered or displayed on the form. These instructions in a request map, or associate, fields on the form to fields in the record.
- Create a request library definition that names the request.
- Use the request library definition to build a request library file.
- Write a program that opens the request library file and executes the request.

1.3 A Simple Form Definition

Figure 1-1 shows a simple form definition as it appears when you create the form.

EMPLOYEE_FORM



```
                         Employee Form

      NAME: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA


      ADDRESS:
        STREET: XXXXXXXXXXXXXXXXXXXX
          CITY: AAAAAAAAAAAAAAAAAAAA
         STATE: AA
           ZIP: 99999
```

Figure 1-1: A Simple Form Definition

This form definition includes background text (NAME:, ADDRESS:, STATE:, and so on) and **picture characters** (9, A, X). Picture characters determine the location, length, and picture type of fields. Picture characters indicate the type of data that can be entered in a field. (For example, A indicates that only alphabetic characters can be entered, 9 indicates that only the digits 0-9 can be entered, and X means that any printable character can be entered.) A group of one or more picture characters that make up a single field (for example, 99999, AA) is called a **picture string**. The picture string is not displayed at run time; rather, fields are reserved for data, which is collected or displayed according to instructions in the request.

1.4 A Simple Record Definition

Figure 1-2 shows a simple record definition in CDDL syntax.

EMPLOYEE_RECORD

```
DEFINE RECORD EMPLOYEE_RECORD.
  EMPLOYEE STRUCTURE.
       NAME     DATATYPE TEXT    30.
    ADDRESS STRUCTURE.
       STREET   DATATYPE TEXT    20.
       CITY     DATATYPE TEXT    20.
       STATE    DATATYPE TEXT     2.
       ZIP      DATATYPE TEXT     5.
    END ADDRESS STRUCTURE.
  END EMPLOYEE STRUCTURE.
END EMPLOYEE_RECORD.
```

Figure 1-2: A Simple Record Definition

This definition describes a record with five fields named NAME, STREET, CITY, STATE, and ZIP. The fields named STREET, CITY, STATE, and ZIP make up a **group field,** named ADDRESS. The data type and length of each record field are also included in the record definition.

Record definitions are stored in the CDD using one of the following:

- VAX CDD Data Definition Language (CDDL)
- VAX DATATRIEVE
- VAX DBMS
- VAX Rdb/VMS

1.5 A Simple Request

Figure 1-3 shows a simple request.

EMPLOYEE_REQUEST

```
FORM IS EMPLOYEE_FORM;
RECORD IS EMPLOYEE_RECORD;

CLEAR SCREEN;
DISPLAY FORM EMPLOYEE_FORM;

OUTPUT NAME   TO NAME,
       STREET TO STREET,
       CITY   TO CITY,
       STATE  TO STATE,
       ZIP    TO ZIP;
WAIT;

END DEFINITION;
```

Figure 1-3: A Simple Request

This request contains a series of instructions that:

- Identify the form definition named EMPLOYEE_FORM (shown in Figure 1-1) and the record definition named EMPLOYEE_RECORD (shown in Figure 1-2)
- Clear the terminal screen
- Display the form EMPLOYEE_FORM
- Display, or output, the information in five record fields in the corresponding form fields
- Wait for operator acknowledgment

This request, when named in a request library definition and built in a request library file, can be executed by a TDMS application program.

The *VAX TDMS Request and Programming Manual* provides complete information about the use of requests, request library definitions, and request library files.

1.6 A Simple Request Library Definition

Figure 1-4 shows a simple request library definition.

```
REQUEST IS EMPLOYEE_REQUEST;

FILE IS "EMPLOYLIB.RLB";
END DEFINITION;
```

Figure 1-4: A Simple Request Library Definition

The request library definition identifies, in one place, all the requests that are used in an application or a portion of an application. It contains only two types of instructions:

- The REQUEST IS instruction, which identifies all the requests you want to list in the request library
- The FILE IS instruction, which names the VMS request library file (RLB) that you create

During the build operation, RDU searches this request library definition, extracts the requests and forms, and places them in the request library file EMPLOYLIB.RLB. The application program then accesses this file.

1.7 Sequence of Events Within TDMS

Now you know the pieces you need to build a TDMS application. Until you are more familiar with TDMS, you can follow a step-by-step procedure in working with these pieces. Figure 1-5: Suggested TDMS Design Sequence gives an overview of this procedure.

You should always start by planning your TDMS application. You need to consider questions such as:

- Who will use this application?
- What records do you have or need? What record fields?
- How will your forms look?
- What is the structure of the application program itself?

Note that you can create either the records or the forms first. You might begin by:

- Looking at the names of records and the names, lengths, and data types of record fields if you use existing records
- Planning the records and record fields if you use new records and already know how the forms should look
- Designing a prototype form for your intended audience if you want a nontechnical user to approve the form before you set up your database

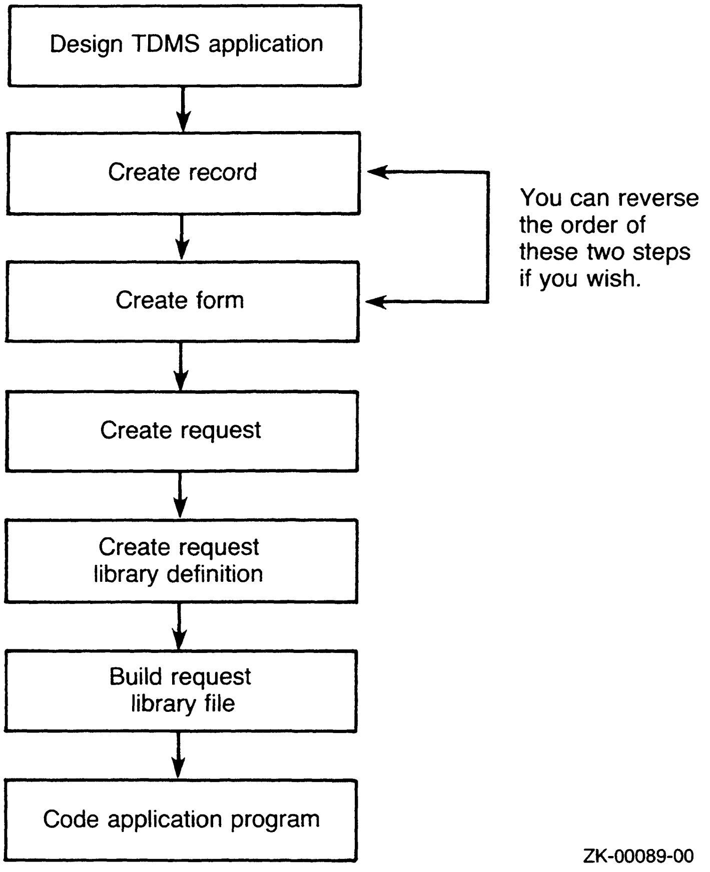

Design TDMS application
Create record
Create form
You can reverse the order of these two steps if you wish.
Create request
Create request library definition
Build request library file
Code application program
ZK-00089-00

Figure 1-5: Suggested TDMS Design Sequence

1.8 Running the TDMS Sample Applications

The TDMS software includes several online sample applications illustrating various TDMS features. These sample applications can be installed as part of the TDMS installation.

1.8.1 The Employee Sample

The Employee sample demonstrates a record-keeping system for employees. The system maintains records, or basic information, about each employee, including the employee's number, name, address, sex, and date of birth. The Employee sample application allows you to add, modify, display, or delete these employee records. Sample records have been provided for employee numbers 1001, 1002, 1003, and 1004.

Before you run the Employee sample, you must copy the file TDMS$EXAMPLES:EMPLOYEE.DAT into your default directory and then execute a command procedure to set up process logical names. Run the Employee sample to understand how a TDMS application looks at run time.

There are two versions of the Employee sample, one version using forms that require a terminal with the Advanced Video Option (AVO), the other version using forms that do not require an AVO terminal.

To copy the data file, type:

```
$ COPY TDMS$EXAMPLES:EMPLOYEE.DAT EMPLOYEE.DAT
```

To execute the command procedure that defines logical names, type:

```
$ @TDMS$EXAMPLES:TDMSSAMP
```

To run the Employee sample on a terminal with AVO, type:

```
$ RUN TDMS$EXAMPLES:EMPBASAVO
```
(To run the sample coded in BASIC)

```
$ RUN TDMS$EXAMPLES:EMPCOBAVO
```
(To run the sample coded in COBOL)

```
$ RUN TDMS$EXAMPLES:EMPFORAVO
```
(To run the sample coded in FORTRAN)

To run the Employee sample on a terminal that does not have AVO, type:

```
$ RUN TDMS$EXAMPLES:EMPLOYBAS
```
(To run the sample coded in BASIC)

```
$ RUN TDMS$EXAMPLES:EMPLOYCOB
```
(To run the sample coded in COBOL)

```
$ RUN TDMS$EXAMPLES:EMPLOYFOR
```
(To run the sample coded in FORTRAN)

1.8.2 Extended Sample Applications

There are two extended sample applications installed with the TDMS software. The Personnel sample application shows a solution to the administrative problem of maintaining a complete set of employee records. The Department sample application shows one solution to the administrative problems of updating dates of an employee's performance appraisal and salary history and weekly reporting of an employee's hours per project. Each of these sample applications demonstrates some advanced features of VAX TDMS such as scrolling and use of program request keys to move between functions.

1.8.2.1 Running the Personnel Sample Application — The Personnel sample is coded in two languages:

- BASIC
- COBOL

Before you run the Personnel sample, you should copy the data files that are in TDMS$EXAMPLES to your default directory and you must execute a command procedure to set up process logical names. There are nine sample records in the data files. The personnel numbers associated with the sample records are 1001 to 1010. To copy the data files, type:

```
$ COPY TDMS$EXAMPLES:PERMAIN.DAT *
$ COPY TDMS$EXAMPLES:PEREDUC.DAT *
$ COPY TDMS$EXAMPLES:PERFAML.DAT *
$ COPY TDMS$EXAMPLES:PERHIST.DAT *
```

To execute the command procedure, type:

```
$ @TDMS$EXAMPLES:TDMSSAMP
```

To run the Personnel sample coded in BASIC, enter:

```
$ RUN TDMS$EXAMPLES:PERSONBAS <RET>
```

To run the Personnel sample coded in COBOL, enter:

```
$ RUN TDMS$EXAMPLES:PERSONCOB <RET>
```

1.8.2.2 Running the Department Sample Application — The Department sample is coded in two languages:

- BASIC
- COBOL

Before you run the Department sample, you should copy the data files that are in TDMS$EXAMPLES to your default directory and you must execute a command procedure to set up process logical names. There are nine sample records in the data files. The personnel numbers associated with the sample records are 1001 to 1010. To copy the data files, type:

```
$ COPY TDMS$EXAMPLES:DEPLABR.DAT *
$ COPY TDMS$EXAMPLES:DEPPRIV.DAT *
$ COPY TDMS$EXAMPLES:PERMAIN.DAT *
$ COPY TDMS$EXAMPLES:PERHIST.DAT *
```

To execute the command procedure, type:

```
$ @TDMS$EXAMPLES:TDMSSAMP
```

Note

Only one person at a time can access a single copy of the data files used by the sample application. The reason is that the labor file is organized sequentially, and sequential files do not allow concurrent access. If more than one person needs to run the Department sample at the same time, you can copy the data files to a different VMS directory and run the department sample with the copied files.

To run the Department sample coded in BASIC, enter:

```
$ RUN TDMS$EXAMPLES:DEPARTBAS <RET>
```

To run the Department sample coded in COBOL, enter:

```
$ RUN TDMS$EXAMPLES:DEPARTCOB <RET>
```

1.8.3 Creating the CDD Directory for Sample CDD Objects

Installing the sample applications is an optional part of the TDMS installation procedure. If you choose to install the samples, TDMS only links the samples to produce executable images and provides the sample application source programs and request library files. You can then run the samples but the CDD directory containing the forms, records, requests, and request library definitions used by the samples is not created.

To create the CDD directory containing the objects used by the samples, you must run the following command file.

```
$ @TDMS$EXAMPLES:TDMSBLDSAMPLE.COM
```

In addition to storing the objects used by the samples, this command file rebuilds the sample request library files. The objects are stored in CDD$TOP.TDMS$SAMPLES.

Walking Through a Simple TDMS Application 2

A major function in any business enterprise is to collect information. Generally, the best way to do this is to use forms. Forms vary in clarity and complexity, as anyone who has filled out a government form knows. TDMS allow lets you only design the best form for your needs, but customize how you want to organize, display, and collect that information.

A common business task is to keep track of employee data. Very simply, you need to know the employee's name, birth date, and identification number. In addition, if the employee is married, you may want to collect information about the spouse.

This chapter gives you step-by-step instructions on creating a simple TDMS application to collect employee information. In addition, the application lets you update records and display information. Feel free to adapt the application to your particular needs. The employee number, for example, can easily be changed to a social security number. This helps you to create a database that has meaning to you.

You should allow approximately two hours to complete this walkthrough, which is organized in the following main parts:

- Creating a form definition
- Creating records
- Creating requests
- Creating a request library definition
- Building a request library file
- Writing the application program

2.1 Getting Started

Before you can create a form definition, you must have an area in the CDD where you want to store your form definition (and any other TDMS definitions you create or use). If you do not have a CDD directory, (or are unsure of what a CDD directory is), refer to the *VAX Common Data Dictionary Utilities Reference Manual* before continuing with this walkthrough.

For example, you might want to store the definitions that you create or modify in a CDD subdirectory using your last name. In that case, type the following logical assignment (using your own last name) at DCL level:

```
$ DEFINE CDD$DEFAULT CDD$TOP.(your_last_name)
```

When you have defined CDD$DEFAULT in this manner, CDD precedes any partial references that you make to a CDD directory or object with CDD$TOP.(your_last_name).

The name you assign the form is the CDD given name as it is stored in the CDD.

In this walkthrough, the name you assign the form is FAMILY_FORM. Because you defined CDD$DEFAULT to point to your personal CDD directory earlier, FDU will store the form in that directory:

CDD$TOP.(name_of_your_directory).FAMILY_FORM

If you get an error when you enter the CREATE FORM command, then probably:

- You did not define CDD$DEFAULT to point to your personal CDD directory. Change your default CDD directory and try to create the request again.
- You already have a request named FAMILY_FORM in your personal CDD directory. Delete the existing form.

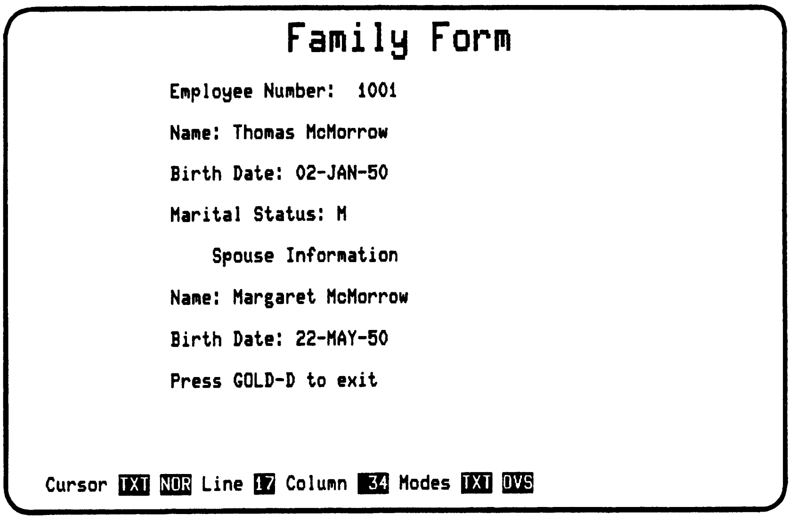
If neither of these corrects the problem, see your system manager.

2.2 Creating a Form Definition

Figure 2-1 shows the form that you create in this walkthrough as it might appear after the operator enters information for a new employee. The form definition that you create includes:

- **Background text** (characters that TDMS always displays when the form is on the screen)
- **Fields** (locations on the form in which TDMS displays data or where the operator can enter data)

Your goal in creating this form is to make it easy to collect accurate information. You want to give the form a title, indicate the particular data you want to collect, and give instructions on how to exit the application at any point during its use.



```
                    Family Form

        Employee Number:  1001

        Name: Thomas McMorrow

        Birth Date: 02-JAN-50

        Marital Status: M

            Spouse Information

        Name: Margaret McMorrow

        Birth Date: 22-MAY-50

        Press GOLD-D to exit


 Cursor TXT NOR Line 17 Column 34 Modes TXT OVS
```

Figure 2-1: Run-Time Form (with Sample Data)

You create this form by following the steps that are explained in the following sections:

1. Entering FDU
2. Assigning a form name and entering the form editor
3. Assigning formwide attributes
4. Creating a screen image for the form
5. Assigning field attributes and validators
6. Saving the form definition and storing it in the CDD

Follow the instructions in this walkthrough carefully to create the form definition. Enter text that is shown in colored ink. Unless otherwise stated, always follow each step by pressing the RETURN key.

2.2.1 Entering the Form Definition Utility (FDU)

To use the Form Definition Utility (FDU), you must use a VT100 or VT200 terminal. Make sure that the VMS operating system recognizes the terminal by issuing the following command at DCL level:

```
$ SET TERMINAL/INQUIRE
```

Now, enter FDU by issuing the command:

```
$ RUN SYS$SYSTEM:FDU.EXE
```

The system returns the FDU> prompt. You are now ready to begin using FDU.

2.2.2 Assigning a Form Name and Entering the Form Editor

To create a form definition, you must first assign a **form name**. The form name identifies a unique CDD location where you want to store your form definition. To create the form named FAMILY_FORM, type the following at the FDU> prompt:

```
FDU> CREATE FORM FAMILY_FORM
```

FDU does not accept this command if a form is stored already in the CDD with the name FAMILY_FORM. If FDU issues an error message because there is already a form entitled FAMILY_FORM, choose another form name and issue the CREATE FORM command using a new form name. If FAMILY_FORM already exists because you have partially completed this walkthrough and you want to continue, enter FDU and use the MODIFY FORM command. For example, enter the following text:

```
$ RUN SYS$SYSTEM:FDU.EXE
FDU> MODIFY FORM FAMILY_FORM
```

If you want to end the form editing session at any time during this walkthrough, press the sequence GOLD-KP7. If the form does not contain errors, TDMS displays the Phase Selection menu. If there are errors in the form, TDMS displays an error message at the bottom of the screen.

To save your form definition and return to FDU command level, type E and press RETURN twice. You can then return to DCL level by typing EXIT and pressing RETURN or CTRL/Z at the FDU> prompt.

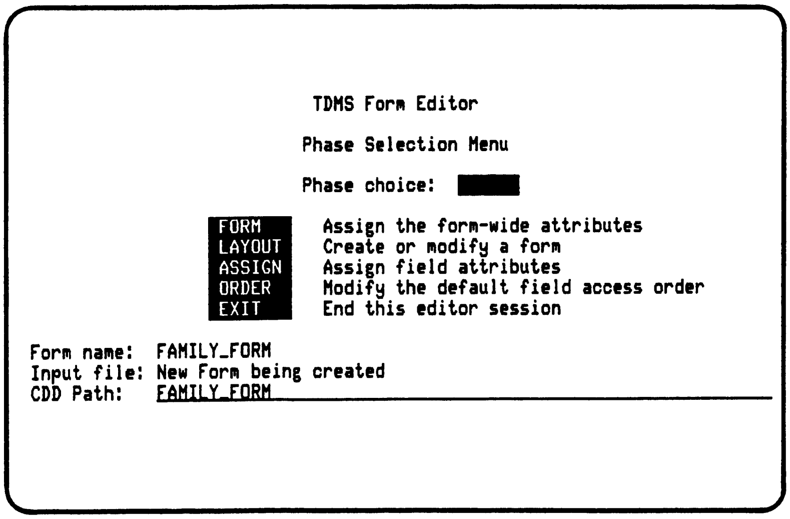
Later, to continue creating your form definition, issue the MODIFY FORM command, reenter the appropriate form editor phase from the Phase Selection menu, and continue the walkthrough from the point where you left off.

If you do not want to save your form definition, press RETURN and type N at the prompt "Do you want to save this form?". You can then return to the FDU> prompt by pressing RETURN.

2.2.3 Assigning Formwide Attributes

The first step in creating a form is to determine how you want the form to look. FDU allows you to give the entire form certain visual characteristics. You assign these formwide attributes during the **Form phase**. In a subsequent phase, you assign attributes to individual fields.

When you issue the CREATE FORM (or the MODIFY FORM) command and press RETURN, the form editor displays the Phase Selection menu on your screen as shown in Figure 2-2.



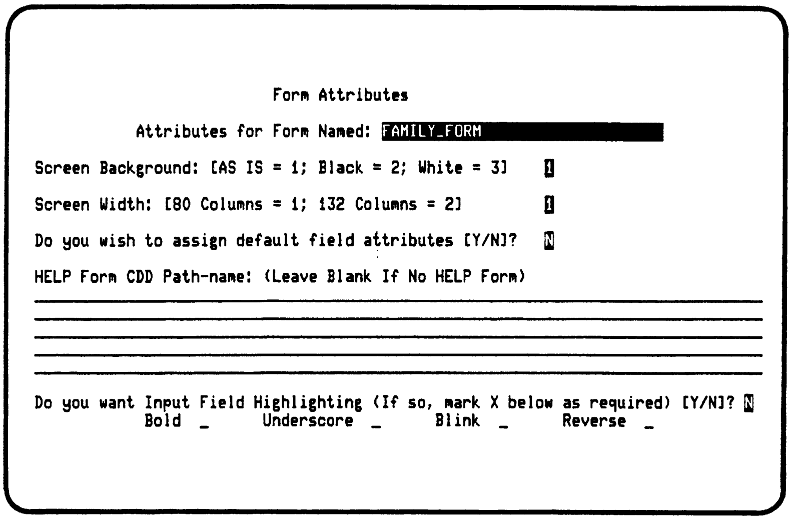
```
                    TDMS Form Editor

                  Phase Selection Menu

                  Phase choice:

            FORM      Assign the form-wide attributes
            LAYOUT    Create or modify a form
            ASSIGN    Assign field attributes
            ORDER     Modify the default field access order
            EXIT      End this editor session

Form name:  FAMILY_FORM
Input file: New Form being created
CDD Path:   FAMILY_FORM
```

Figure 2-2: Phase Selection Menu

The cursor is positioned at the Phase Choice prompt. To enter the Form phase, type:

```
FORM
```

You can type an abbreviation of any phase choice that makes the name unique. For example, you can type F, FO, or FOR to enter the Form phase. Figure 2-3 shows the Form phase screen.



```
                         Form Attributes

        Attributes for Form Named: FAMILY_FORM

Screen Background: [AS IS = 1; Black = 2; White = 3]   1

Screen Width: [80 Columns = 1; 132 Columns = 2]        1

Do you wish to assign default field attributes [Y/N]?  N

HELP Form CDD Path-name: (Leave Blank If No HELP Form)
______________________________________________________________
______________________________________________________________
______________________________________________________________
______________________________________________________________
______________________________________________________________

Do you want Input Field Highlighting (If so, mark X below as required) [Y/N]? N
         Bold _      Underscore _      Blink _      Reverse _
```

Figure 2-3: Form Phase Screen

The goal in this phase is to create a form with the following attributes:

- Black (dark) background
- 80-column width
- No input field highlighting
- No Help form

While you are filling out a form, if you want to move back to any field to change your response, press the BACK SPACE key. Each time you press the BACK SPACE key, the cursor moves back one field. If you want to move forward to any field, press the TAB key. Each time you press the TAB key, the cursor moves forward one field. Pressing the RETURN key tells FDU to save your choices. If you wish to change your choices after you press RETURN, you must reenter the phase.

The cursor is at the Screen Background prompt; the Screen Background attribute determines the shading of the screen background at run time. The form editor displays the default value 1, which leaves the screen background unchanged. Type the following number:

```
2
```

By typing 2, you assign a black (dark) screen background. After you type 2, the cursor automatically moves to the Screen Width prompt. Accept the rest of the default values at each prompt (an 80-column screen, no Input Field highlighting, no Help form, and no default field attributes) by pressing:

```
<RET>
```

FDU returns you to the Phase Selection menu.

2.2.4 Creating a Screen Image of the Form

By following the steps in this section, you create the screen image of the form. You do this in the Layout phase.

While creating the form, you also learn how to use function keys to:

- Distinguish between background text and fields
- Create a double-size line
- Center text on a line
- Move the cursor
- Return to the Phase Selection menu

To enter the Layout phase of the form editor, at the Phase Choice prompt, type:

```
LAYOUT
```

The top 23 lines of your terminal screen clear, and the **cursor status line** (the bottom, or 24th line) shows the following:

```
Cursor TXT NOR Line 1 Column 1 Modes TXT OVS
```

In Chapter 5 of this manual, you learn about the cursor status line and each of its elements. For this walkthrough, you should know that the first block after the word Modes displays either TXT or FLD, distinguishing Text mode (TXT) from Field mode (FLD). You are creating background text when you see TXT, and you are creating a field when you see FLD.

For example, in Figure 2-1, Name is background text. Background text notes the kind of information needed and serves as a prompt on the run-time form. The name Thomas McMorrow occupies a field, or area on the form where actual data is entered or displayed.

The cursor is now at the upper left corner of the screen at line 1, column 1. Note that the cursor status line shows the current cursor location. To insert a title for the form, move the cursor to line 2, column 1 by pressing:

```
<RET>
```

Then type:

```
Family Form
```

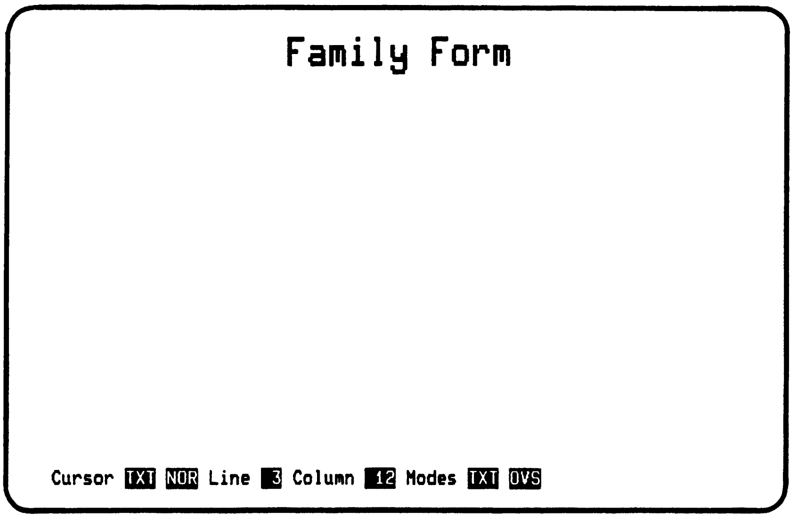
Now, change the line to a double-size (double-height and double-wide) line by pressing the sequence:

```
<GOLD-S>
```

You can then center the title, Family Form, by pressing:

```
<KP1>
```

The cursor is on line 3, and your form looks like this:



```
                    Family Form









Cursor TXT NOR Line 3 Column 12 Modes TXT OVS
```

The title is double-size and it is centered on line 3. To move the cursor to line 5, column 1, press:

`<RET>` (two times)

The background text Employee Number begins at line 5, column 13. To move to column 13, you can press space 12 times or use the repeat function by entering the following key sequence:

```
<GOLD>-12-<space>
```

The cursor moves 12 spaces, to line 5 column 13. You can use the repeat function (GOLD-n-space) for any character except the digits 0-9.

Enter the background text:

```
Employee Number:
```

Leave one space after the colon for readability.

Now you want to create the field used to collect or display the employee number. Creating a field is different from what you have done so far. You have to define acceptable characteristics for what a user can enter. You need to control the data that is entered to make sure it is correct and can be processed as intended.

Switch to Field mode by pressing:

```
<GOLD-KP8>
```

Notice that the block to the right of Modes on the cursor status line switches from TXT to FLD. When you are in Field mode (FLD), the form editor accepts only **picture characters**, **field constants**, or spaces. Picture characters specify the location, length, and type of fields and determine the type of data that can be entered or displayed in a form field. Field constants are commonly used symbols and punctuation marks displayed in a field whenever TDMS displays the form. You should be in Field mode *only* when typing picture characters or field constants.

In this walkthrough, you use two picture characters:

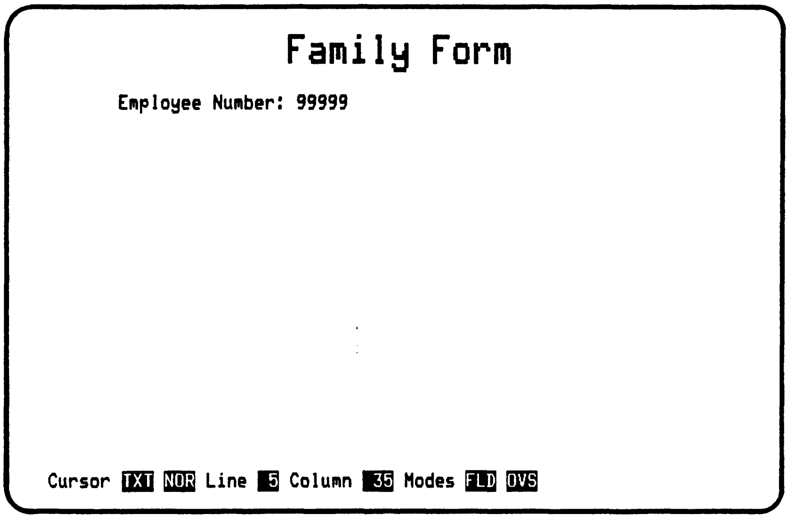
A, denoting an alphabetic character (only the characters A-Z, a-z, space, or alphabetic characters from the DEC Multinational Character Set can be displayed or entered at run time)

9, denoting a numeric character (only the characters 0-9 can be displayed or entered at run time)

Since employee numbers in your company can have up to five digits (but no letters or symbols), you must identify a five-character field that has numeric picture characters by typing:

```
99999
```

Here is an example of the form.



```
                    Family Form

    Employee Number: 99999







 Cursor TXT NOR Line 5 Column 35 Modes FLD OVS
```

Now, change back to Text mode by pressing:

`<KP8>`

After you press KP8, the block to the right of Modes switches from FLD to TXT. To move to line 7, column 1, press:

`<RET>` (two times)

Move 12 columns to the right by entering:

`<GOLD>-12-<space>`

The cursor is at line 7, column 13. Type:

`Name:`

Remember to leave a space after the colon. To switch to Field mode, press:

`<GOLD-KP8>`

To identify a 20-character field representing the employee's name, enter the following sequence:

```
<GOLD>-20-<A>
```

Then, to switch back to Text mode, press:

```
<KP8>
```

Now use the RETURN key and repeat key sequences to position the cursor at line 9, column 12. Then type the background text for the next field:

```
Birth Date:
```

On this form, the format for the employee's birth date is Day-Month-Year (for example, 01-DEC-49). TDMS provides a predefined key sequence to create a date field. Enter the following next to Birth Date:

```
<GOLD-D>
```

The date choices appear on the bottom of the screen:

```
Choice: _ 1(month day, year) 2(dd-mmm-yy) 3(mm/dd/yy) 4(dd-mm-yy) 5(dd-mmm-yyyy)
```

Next to Choice:, type:

```
2
```

For TDMS to accept your choice, type:

```
<RET>
```

By typing 2 and then pressing RETURN, you automatically select your desired date field. Notice that you do not have to select Field mode. TDMS selects it for you and returns you to Text mode after entering the correct date format.

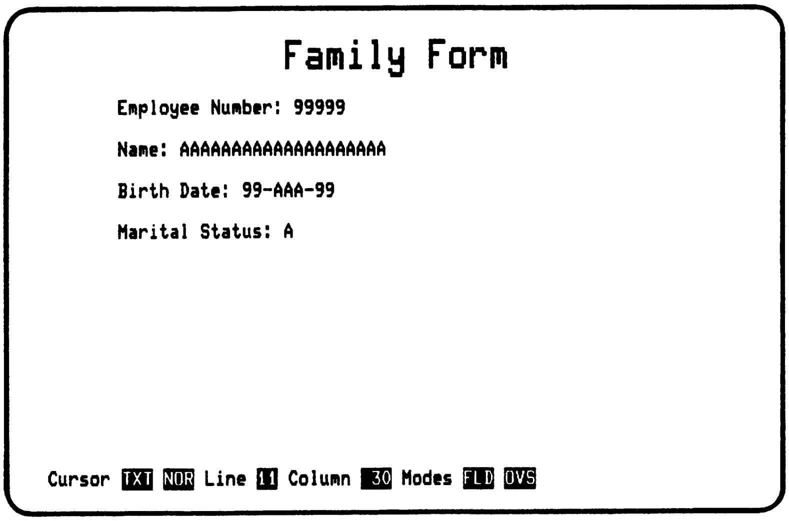
Now move to line 11, column 13 to enter the next field. Type the background text:

```
Marital Status:
```

Then switch back to Field mode and type:

```
A
```

At this point, this is what your form looks like.

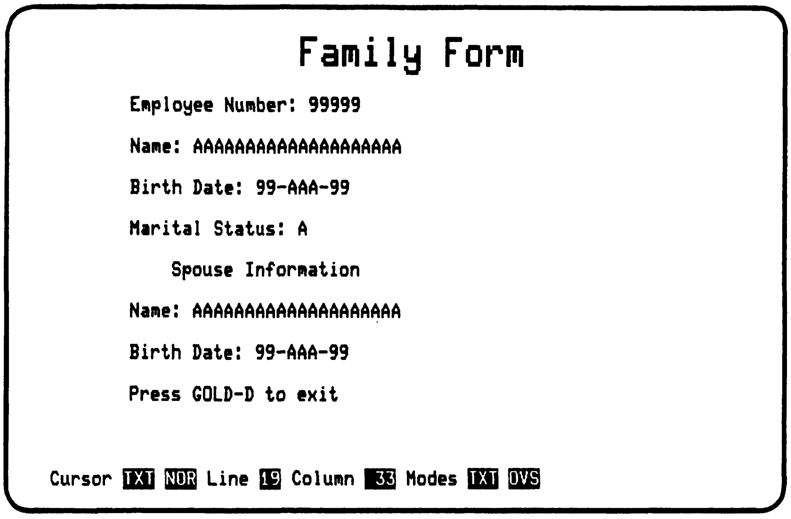


```
                    Family Form

        Employee Number: 99999

        Name: AAAAAAAAAAAAAAAAAAAA

        Birth Date: 99-AAA-99

        Marital Status: A







   Cursor TXT NOR Line 11 Column 30 Modes FLD OVS
```

The remainder of the form is similar to the part you just created. Create the remaining text and fields by entering the following:

```
<KP8>
<RET>
<RET>
<GOLD>-16-<space>
Spouse Information
<RET>
<RET>
<GOLD>-12-<space>
Name:
<GOLD-KP8>
<GOLD>-20-<A>
<KP8>
<RET>
<RET>
<GOLD>-12-<space>
Birth Date:
<GOLD-KP8>
<GOLD-D>
<2>
<RET>
<RET>
<RET>
<KP8>
<GOLD>-12-<space>
Press GOLD-D to exit
```

This is how your completed form looks.



```
                    Family Form

     Employee Number: 99999

     Name: AAAAAAAAAAAAAAAAAAAA

     Birth Date: 99-AAA-99

     Marital Status: A

        Spouse Information

     Name: AAAAAAAAAAAAAAAAAAAA

     Birth Date: 99-AAA-99

     Press GOLD-D to exit


  Cursor TXT NOR Line 19 Column 33 Modes TXT OVS
```

You have finished identifying background text and fields, so you are ready to leave the Layout phase and return to the Phase Selection menu. Use the MENU function key by pressing the sequence:

```
<GOLD-KP7>
```

The form editor displays the Phase Selection menu.

2.2.5 Assigning Field Attributes

In this step, you assign attributes to specific fields that you have created.

Field attributes allow you to:

- Identify the field to TDMS by providing a field name.
- Provide a **default value** for the field. A default value allows you to identify a frequently-used response (for example, NY in a field representing the employee's home state when most of the employees live in New York).
- Provide a help message for the field that the operator can read at run time.
- Affect the appearance of the field and the appearance of data entered into the field. For example, a state abbreviation could appear as uppercase, even if a user entered it in lowercase.

- Place certain conditions on the data entered by the operator. For example, if all employee numbers have five digits, you can set up the field to accept five and only five digits.
- Inform TDMS that there are special validation procedures (field validators) associated with the field. For example, employees listed in your application can only be married or single, so you can set up the marital status field to accept only M or S as valid entries.

You assign field attributes in the Assign phase of the form editor. Chapter 6 provides a detailed discussion about the Assign phase and the use of field attributes. This walkthrough only introduces these topics.

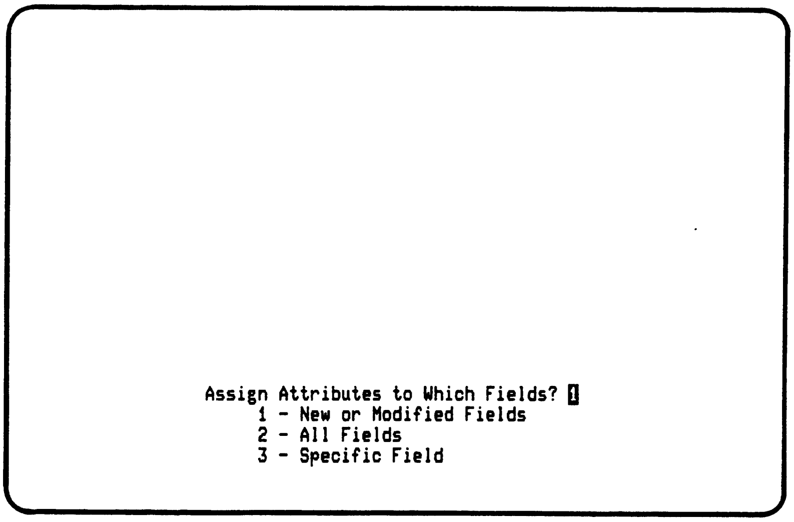
Note

Do not press RETURN while in the Assign phase. If you press RETURN, TDMS brings you to the next field to begin assigning attributes. If you press RETURN while on the last field, TDMS ends the phase and returns you to the Phase Selection menu.

To enter the Assign phase of the form editor from the Phase Selection menu, type:

```
A
```

The form editor displays the Assign Phase menu at the bottom of the screen. See Figure 2-4.



```
Assign Attributes to Which Fields? █
     1 - New or Modified Fields
     2 - All Fields
     3 - Specific Field
```

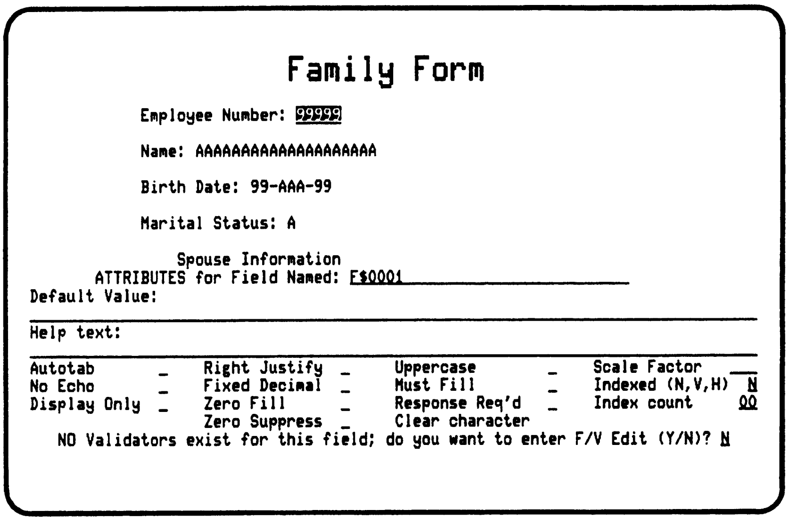
Figure 2-4: Assign Phase Menu

To assign attributes to all fields, type:

```
2
```

The form editor displays the Attribute Assignment form superimposed on the form you are creating.

The field to which you are assigning attributes is bolded, underlined, and blinking. The field next to the background text Employee Number is the first field on the form, so the picture characters (99999) are displayed bolded, underlined, and blinking. The screen looks like the following example.



```
                    Family Form

        Employee Number: 99999

        Name: AAAAAAAAAAAAAAAAAAAA

        Birth Date: 99-AAA-99

        Marital Status: A

             Spouse Information
     ATTRIBUTES for Field Named: F$0001____________________
Default Value:
_____________________________________________________________________
Help text:
_____________________________________________________________________
Autotab      _   Right Justify _   Uppercase       _   Scale Factor  __
No Echo      _   Fixed Decimal _   Must Fill       _   Indexed (N,V,H) N
Display Only _   Zero Fill     _   Response Req'd  _   Index count   00
                 Zero Suppress _   Clear character
   NO Validators exist for this field; do you want to enter F/V Edit (Y/N)? N
```

The attributes that you assign apply only to the current field.

The cursor is now at the first character of the Field Name attribute. Each form field must have a unique field name; the form editor assigns the default field name F$0001 to the first field. You could accept this default field name; however, in order to make the field name more descriptive, change the name to EMPLOYEE_NUMBER. To make this change, press:

```
<LINE FEED>
```

When you press LINE FEED, the name F$0001 is deleted, and the cursor is at the beginning of the line. To assign the name EMPLOYEE_NUMBER to this field, type:

```
EMPLOYEE_NUMBER
```

TDMS field names can contain up to 31 characters, including alphabetic characters (A-Z and a-z), numbers, dollar signs ($), and underscores (_). The first character must be an alphabetic character, and the last character can be neither a dollar sign nor an underscore.

To go to the next attribute, press:

```
<TAB>
```

The cursor is at the first character of the Default Value attribute. When you supply a default value for a field, the value is displayed on the form at run time unless a request explicitly displays a different value in the field. If most employees live in Massachusetts, for example, you could give MA as the default value for a state field. Because each employee number is unique, you do not want a default number here.

To go to the next attribute, press:

```
<TAB>
```

The cursor is now at the first character of the Help Text attribute. You can specify a message (using up to 80 characters) that the operator can read by pressing the HELP (PF2 or F15) key at run time. Type in the message:

```
Type the employee's 5-digit employee number.
```

There cannot be an employee record without an employee number, so you want to make a number entry mandatory. You can do this by using the Response Required attribute. At run time, you must enter a number into this field or TDMS signals an error. To move to the Response Required prompt, press:

`<TAB>` (ten times)

When you do not assign an attribute, you accept a default. In this case, you accept the following defaults:

— Autotab (deassigned, thus preventing the cursor from moving automatically to the next input field when the current one is filled at run time).

— No Echo (deassigned, thus allowing the operator's response to be displayed on the screen at run time).

— Display Only (deassigned, thus allowing the operator to enter data if the request maps to the field for input).

— Right Justify (deassigned, thus allowing run time data to appear at from the left margin on the field.)

— Fixed Decimal (deassigned, providing run-time characteristics for an UNSIGNED NUMERIC field.)

— Zero fill (deassigned, thus assigning a blank fill character).

— Zero suppress (deassigned, thus allowing a zero to be displayed on a form when a numeric record field with a null (zero) value is output to a form field).

- Uppercase (deassigned, thus not converting lowercase input to uppercase).
- Must Fill (deassigned, thus not requiring the operator to fill in the field completely if any data is entered).

Each employee has a unique identification number that must be entered on the form. To do this you assign the Response Required attribute by typing:

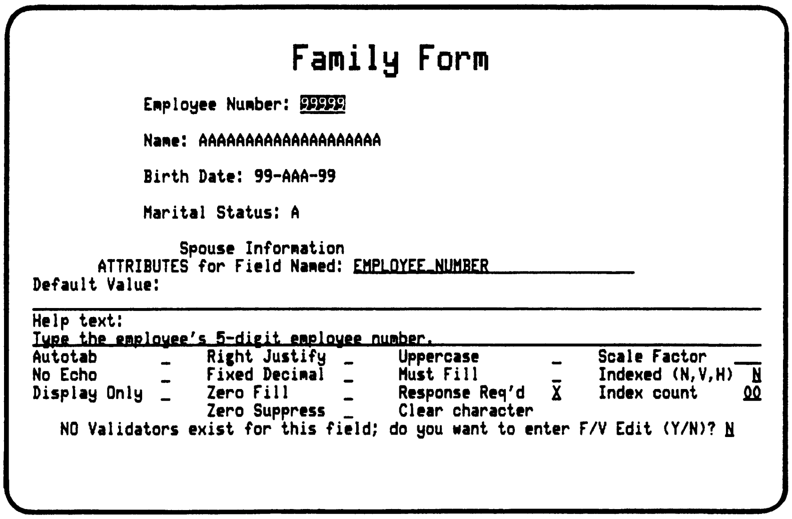
```
X
```

As the result of assigning this attribute, the operator must enter data in this field at run time. To accept the default value of the Clear Character, the Scale Factor, and the Indexed attributes, press:

`<TAB>` (four times)

The Clear Character attribute determines the appearance of the field at run time when the field is blank or partially blank. The Scale Factor is a positive or negative integer that represents the location of a decimal point in a numeric field. The Indexed attribute aligns two or more elements in a single form field either vertically or horizontally.

The Attribute Assignment form looks like the following example.



```
                         Family Form

          Employee Number: 99999

          Name: AAAAAAAAAAAAAAAAAAAA

          Birth Date: 99-AAA-99

          Marital Status: A

              Spouse Information
     ATTRIBUTES for Field Named: EMPLOYEE_NUMBER
Default Value:
________________________________________________________________
Help text:
Type the employee's 5-digit employee number.
Autotab        _   Right Justify  _   Uppercase       _   Scale Factor   ___
No Echo        _   Fixed Decimal  _   Must Fill       _   Indexed (N,V,H)  N
Display Only   _   Zero Fill      _   Response Req'd  X   Index count     00
                   Zero Suppress  _   Clear character
   NO Validators exist for this field; do you want to enter F/V Edit (Y/N)? N
```

The last attribute on the form is the Field Validation attribute. It allows you to select ranges and choices of values that will be accepted by TDMS at run time. Since you do not want to specify any special validators press:

```
<RET>
```

After you press RETURN, TDMS redisplays the Attribute Assignment form and the field representing the employee's name is shown bolded, underlined, and blinking. To override the default field name, press:

```
<LINE FEED>
```

Then type:

```
EMPLOYEE_NAME
```

Now, move to the Help Text attribute by pressing:

`<TAB>` (two times)

Type the message:

```
Type the employee's name.
```

Again, you want to make this field required. Move to the Response Required attribute by pressing:

`<TAB>` (ten times)

Assign the attribute by typing:

```
X
```

The operator cannot move past the Response Required field at run time without making an entry. Accept the current default for the remaining attributes by pressing:

```
<RET>
```

After you press RETURN, TDMS redisplays the Attribute Assignment form and the field representing the employee's birth date is shown bolded, underlined, and blinking. To override the default field name, press:

```
<LINE FEED>
```

Then type:

```
BIRTH_DATE
```

To move to the Help Text attribute, press:

`<TAB>` (two times)

Type the message:

```
Type the employee's birth date.
```

Accept the current defaults for the remaining attributes by pressing:

```
<RET>
```

After you press RETURN, TDMS redisplays the Attribute Assignment form and the field representing the employee's marital status is shown bolded, underlined, and blinking. To override the default field name, press:

```
<LINE FEED>
```

Then type:

```
MARITAL_STATUS
```

Move to the Help Text attribute and type the message:

```
Type the Employee's marital status: M or S
```

Move to the Uppercase attribute and type:

```
X
```

At run time, TDMS converts the entry in this field to an uppercase character. To end the assignment of attributes for this field and move on to the next, press:

```
<RET>
```

After you press RETURN, TDMS redisplays the Attribute Assignment form and the field representing the employee spouse's name is shown bolded, underlined, and blinking. To override the default field name, press:

```
<LINE FEED>
```

Then type:

```
SPOUSE_NAME
```

Move to the Help Text attribute and type the message:

```
Type the name of the employee's spouse.
```

To end the assignment of attributes for this field and move on to the next, press:

```
<RET>
```

After you press RETURN, TDMS redisplays the Attribute Assignment form and the field representing the birth date of the employee's spouse is shown bolded, underlined, and blinking. To override the default field name, press:

```
<LINE FEED>
```

Then type:

```
SPOUSE_BIRTH_DATE
```

Move to the Help Text attribute and type the message:

```
Type the birth date of the employee's spouse.
```

To end the assignment of attributes for this field, press:

```
<RET>
```

Because Birth Date is the last field on the form, the form editor displays the Phase Selection menu.

2.2.6 Saving the Form Definition and Storing It in the CDD

When you are satisfied with the form definition, it is ready to be saved and stored in the CDD. Exit the Phase Selection menu by typing:

```
E
```

Then press:

```
<RET>
```

The cursor moves to the bottom of the form, where you are asked if you want to save the form. Accept the default (Y, for yes) by pressing:

```
<RET>
```

At this point, FDU stores the form definition in the CDD and gives the form definition the form name that you specified when you issued the CREATE FORM command. When the form is stored successfully, you see the FDU> prompt on your terminal.

Now you are done creating the form. To move on to the next phase of building your TDMS application, type:

```
FDU> EXIT
```

TDMS returns you to DCL level.

The next step in creating a TDMS application is to create a record. For this walkthrough, the names you give to the record fields are the same as the names of the form fields.

2.3 Creating Records

A record definition defines the type, structure, and length of data in records. All data called by a request must be defined in a record and stored in the CDD, including data entered by the operator as well as information that is displayed on the terminal screen.

A record is a key part of the TDMS application because it allows you to collect information as part of a database. A record allows you to keep information in a file. Records can also be used to collect temporary information that does not need to be kept as part of the permanent database.

In this part of the walkthrough, you create two records using the VAX CDD Data Definition Language (CDDL).

Creating a record consists of two steps:

1. Using a text editor to create a CDDL source file that defines a record
2. Invoking the CDDL compiler to insert the record definition into the CDD

In the source file, you specify the name and description of the record, the fields that comprise it, and the attributes of those fields.

The source file contains the record definition to be placed in the CDD. The default file type is .DDL.

2.3.1 Source Files

The first record you need to create for your application is the database record. This record collects information that you want to save for later use. For example, you want to keep an employee's name and number as part of a permanent record. Enter the following command at DCL level:

```
$ EDIT/EDT FAMILY.DDL
```

This calls the EDT editor to create the CDDL source file.

The basic unit of a CDDL source file is the DEFINE statement, which names the record you are creating. Following the DEFINE statement is a field description statement defining the record's field attributes. Subordinate field description statements may be embedded within the field description statement. The alignment of text is a visual aid to help you see the organization of the data. It does not affect the compilation of the record.

Enter the following text in the file:

```
DEFINE RECORD FAMILY_RECORD.
   FAMILY_RECORD STRUCTURE.
     EMPLOYEE_NUMBER   DATATYPE SIGNED LONGWORD.
     EMPLOYEE_NAME     DATATYPE TEXT 20.
     BIRTH_DATE        DATATYPE DATE.
     MARITAL_STATUS    DATATYPE TEXT 1.
     SPOUSE_NAME       DATATYPE TEXT 20.
     SPOUSE_BIRTH_DATE DATATYPE DATE.
   END FAMILY_RECORD STRUCTURE.
END FAMILY_RECORD.
```

To exit from the editor and save your edits, enter:

```
<CTRL/Z>
*EXIT
```

2.3.2 Compiling the Record Definition

Once you have created a CDDL source file, you can invoke the CDDL compiler to insert your record definition into the CDD. To invoke the CDDL compiler, type the following command:

```
$ RUN SYS$SYSTEM:CDDL
```

The system responds with:

```
$_File:
```

At this prompt, type in the name of the file:

```
$_File: FAMILY.DDL
```

If the record has no errors and if there is no record in the CDD with the same name, the CDDL compiler stores FAMILY.DDL in the CDD.

If the record you submitted has an error, such as a misspelling, the CDDL compiler returns an error message. To correct the record, edit and then recompile the file.

To make replacing the record easier, define a symbol for CDDL. Enter the following line at DCL level:

```
$ CDDL :== $SYS$SYSTEM:CDDL
```

For example, if FAMILY.DDL could not compile, follow these steps:

1. Edit the file using your text editor.
2. After you have made corrections, replace the old record definition with the new by typing the following text at DCL level:

   ```
   $ CDDL/REPLACE FAMILY.DDL
   ```

You now have a corrected version of the record FAMILY_RECORD in the CDD.

The second record you need to create is a workspace record. A workspace record is different from a database record because it is used to collect information that the application program will use then discard. By using a separate workspace record to contain temporary values, you avoid mapping extraneous information to a database. Enter the following text at DCL level:

```
$ EDIT/EDT FAMILYREC.DDL
```

Then enter the following text in the file:

```
DEFINE RECORD FAMILYREC_RECORD.
   FAMILYREC_RECORD STRUCTURE.
      PROGRAM_REQUEST_KEY DATATYPE TEXT 1.
   END FAMILYREC_RECORD STRUCTURE.
END FAMILYREC_RECORD.
```

Note that PROGRAM_REQUEST_KEY is the only field in FAMILYREC_RECORD. This value is used by the application program to test certain conditions and is not stored permanently as part of the database.

Exit the editor and use CDDL to compile your new record definition by entering:

```
$ RUN SYS$SYSTEM:CDDL
$_File: FAMILYREC.DDL
```

2.4 Creating Requests

Now you are ready to create the requests that control the data interchange for your application. In this section, you create both a simple and a conditional request using the Request Definition Utility (RDU).

Note that when you enter commands in RDU, you are at the RDU> prompt. When you enter request instructions, you are at the RDUDFN> prompt.

If you make errors while you are creating either request, continue with the walkthrough, entering all the remaining request instructions. When you finish entering request text and type the final instruction, END DEFINITION, RDU returns you to the RDU> prompt. You can then go back and correct any typing errors using the EDIT command described in the section entitled Correcting Your Errors.

Note

RDU usually remains at the RDUDFN> prompt and continues to take additional request instructions when you make an error. If RDU encounters an error in your request text from which it cannot recover, it may return you to the RDU> prompt before you enter the END DEFINITION instruction. If this happens, you must reenter the request starting with the CREATE REQUEST command.

2.4.1 Entering the Request Definition Utility (RDU)

To enter RDU, type the following command at DCL level:

```
$ RUN SYS$SYSTEM:RDU.EXE
```

The system responds with:

```
RDU>
```

Once you are in RDU, you can issue commands:

- To create requests and request library definitions
- To manipulate (modify, delete, copy, list, replace, and so on) requests and request library definitions

2.4.2 Creating a Simple Request

You can create a request using one of two methods: interactive or file. In this walkthrough, you use the interactive method. The file method is described in the *VAX TDMS Request and Programming Manual.*

To create the requests interactively, type the CREATE REQUEST command and a request name at the RDU> prompt. To start creating your first request, enter the following:

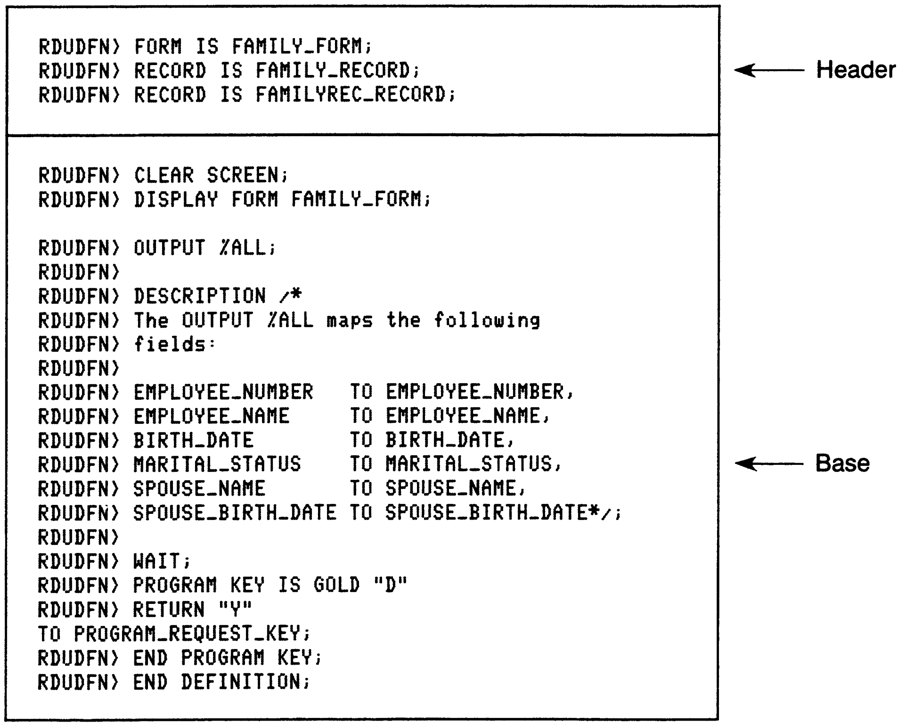
```
RDU> CREATE REQUEST FAMILY_DISPLAY_REQUEST
```

After you enter the CREATE REQUEST command and name the request, RDU displays the RDUDFN> prompt. This prompt indicates that RDU is ready to receive request instructions.

```
RDUDFN>
```

In the next sections, you enter the instructions that make up the FAMILY_DISPLAY_REQUEST. This request clears the terminal screen and displays the form FAMILY_FORM. It also takes data from the database record FAMILY_RECORD and displays it in the form fields. Figure 2-5 shows the two parts of a request:

- The header, which identifies the forms and records used by the request
- The base, which contains instructions that TDMS performs every time an application program calls this request



```
RDUDFN> FORM IS FAMILY_FORM;
RDUDFN> RECORD IS FAMILY_RECORD;
RDUDFN> RECORD IS FAMILYREC_RECORD;

RDUDFN> CLEAR SCREEN;
RDUDFN> DISPLAY FORM FAMILY_FORM;

RDUDFN> OUTPUT %ALL;
RDUDFN>
RDUDFN> DESCRIPTION /*
RDUDFN> The OUTPUT %ALL maps the following
RDUDFN> fields:
RDUDFN>
RDUDFN> EMPLOYEE_NUMBER    TO EMPLOYEE_NUMBER,
RDUDFN> EMPLOYEE_NAME      TO EMPLOYEE_NAME,
RDUDFN> BIRTH_DATE         TO BIRTH_DATE,
RDUDFN> MARITAL_STATUS     TO MARITAL_STATUS,
RDUDFN> SPOUSE_NAME        TO SPOUSE_NAME,
RDUDFN> SPOUSE_BIRTH_DATE TO SPOUSE_BIRTH_DATE*/;
RDUDFN>
RDUDFN> WAIT;
RDUDFN> PROGRAM KEY IS GOLD "D"
RDUDFN> RETURN "Y"
TO PROGRAM_REQUEST_KEY;
RDUDFN> END PROGRAM KEY;
RDUDFN> END DEFINITION;
```

Figure 2-5: Two Parts of FAMILY_DISPLAY_REQUEST

The request header contains the FORM IS and RECORD IS instructions. You must enter these instructions before any mapping instructions.

The request base contains form usage and mapping instructions that TDMS reads and executes each time an application program calls that request.

2.4.2.1 FORM IS Instruction — Usually the first instruction in a request is the FORM IS instruction. This instruction identifies the form or forms you refer to in later instructions.

As you enter instructions, RDU checks that the form you specify exists in the CDD. If the form does not exist, RDU gives you an error message and does not create the request in the CDD. RDU does, however, continue to accept further request instructions and check them for errors.

The form name must be a legal CDD path name. You can select either a *given*, a *full*, or a *relative* path name.

Enter this text:

```
RDUDFN> FORM IS FAMILY_FORM;
```

Be sure to complete each instruction with a semicolon.

A single call to a request can display no more than one form. However, you can identify more than one form definition in a request containing conditional instructions. You will create a conditional request later in this walkthrough.

2.4.2.2 RECORD IS Instruction — You must also name the CDD record definitions you will use later in mapping instructions within the request.

Because you created two records, you need two RECORD IS instructions. Enter this text:

```
RDUDFN> RECORD IS FAMILY_RECORD;
RDUDFN> RECORD IS FAMILYREC_RECORD;
```

2.4.2.3 CLEAR SCREEN Instruction — You want to clear the screen before displaying a form. To do this, use the CLEAR SCREEN instruction. This ensures that there is nothing on the screen before TDMS displays a form. Enter this text:

```
RDUDFN> CLEAR SCREEN;
```

2.4.2.4 DISPLAY FORM Instruction — To view the form on the screen, use the DISPLAY FORM instruction. Enter this text:

```
RDUDFN> DISPLAY FORM FAMILY_FORM;
```

2.4.2.5 OUTPUT TO Instruction — In your application, you want TDMS to move data from the records and display it on the form. To do this, you use the OUTPUT TO instruction with the %ALL parameter. To continue creating your request, enter the following:

```
RDUDFN> OUTPUT %ALL;
```

If you use %ALL, TDMS displays data to all those form fields that have identically named record fields. Note that you do not need to specify any record fields within FAMILY_RECORD.

2.4.2.6 DESCRIPTION Instruction — You can use the DESCRIPTION instruction any place in a request or request library definition where you want to include descriptive text except embedded in a request instruction or a request library definition instruction. The text you enter following the keyword DESCRIPTION and the slash and asterisk symbols (/*) is stored with the request or request library definition in the CDD. You end the descriptive text with the asterisk and slash symbols and a semicolon (*/;).

As in this example, you might want to list which fields are being mapped for output. This is done simply for clarity; the OUTPUT %ALL instruction does the actual mapping. Enter this text:

```
RDUDFN> DESCRIPTION /*
RDUDFN> The OUTPUT %ALL maps the following
RDUDFN> fields:
RDUDFN>
RDUDFN>    EMPLOYEE_NUMBER    TO EMPLOYEE_NUMBER,
RDUDFN>    EMPLOYEE_NAME      TO EMPLOYEE_NAME,
RDUDFN>    BIRTH_DATE         TO BIRTH_DATE,
RDUDFN>    MARITAL_STATUS     TO MARITAL_STATUS,
RDUDFN>    SPOUSE_NAME        TO SPOUSE_NAME,
RDUDFN>    SPOUSE_BIRTH_DATE TO SPOUSE_BIRTH_DATE */;
```

2.4.2.7 WAIT Instruction — You use the WAIT instruction to ensure that the form and the information you mapped to it stays on the screen until you press the RETURN key or PRK at run time. Enter this text:

```
RDUDFN> WAIT;
```

2.4.2.8 PROGRAM KEY IS Instruction — You might want to leave the application while it is running. A convenient way to do this is to use a program request key (PRK).

By returning a message (that you predefine in a request) to the program, the PROGRAM KEY IS instruction permits you to send a run-time message to the application program. The program can then respond to the condition, which in this case lets you to leave the program at any time.

You define a PRK in a request by naming the key and associating request instructions with that key. Enter this text:

```
RDUDFN> PROGRAM KEY IS GOLD "D"
RDUDFN> RETURN "Y" TO PROGRAM_REQUEST_KEY;
RDUDFN> END PROGRAM KEY;
```

When the operator presses the sequence GOLD-D, the value Y is returned to PROGRAM_REQUEST_KEY, a program variable. The program then checks the value of PROGRAM_REQUEST_KEY and exits from the application if it contains Y.

2.4.2.9 END DEFINITION Instruction — When you finish entering request text, type the instruction END DEFINITION. Note that you must put a semicolon (;) after the END DEFINITION instruction. Enter this text:

```
RDUDFN> END DEFINITION;
```

RDU returns you to the RDU> prompt and checks that your mappings are valid according to TDMS mapping rules. It checks that the form and record fields have compatible data types, lengths, and so on.

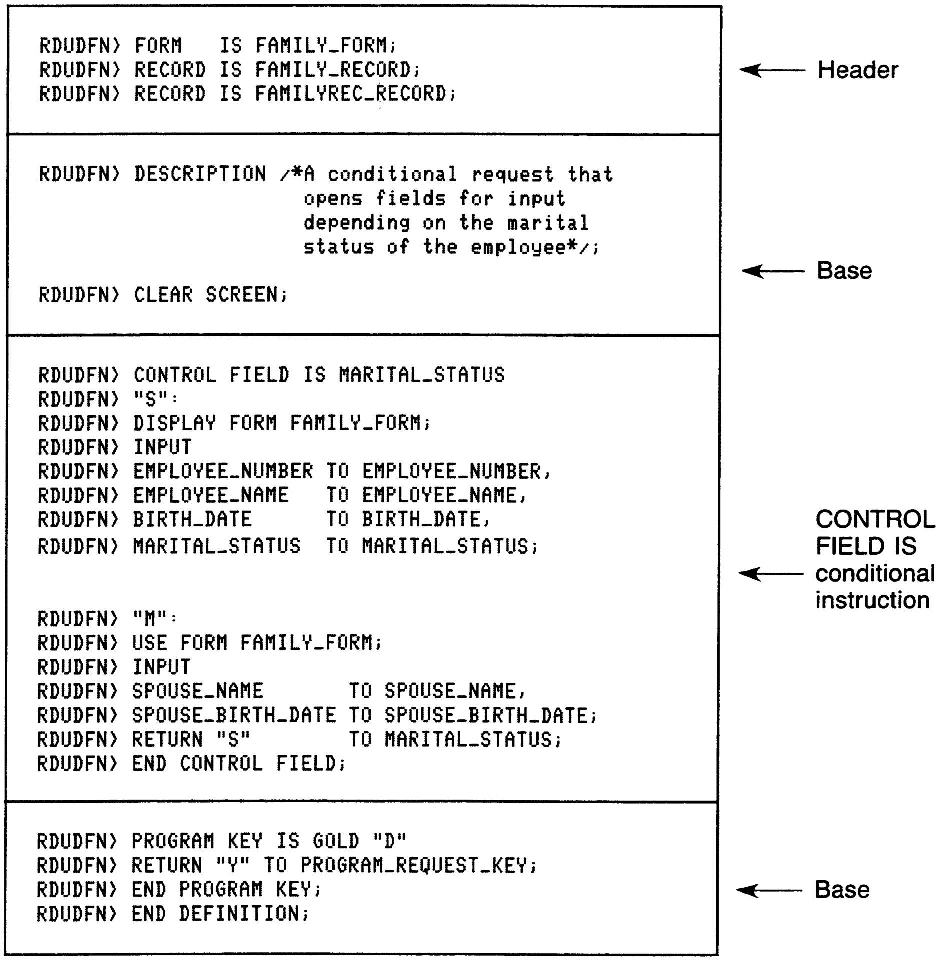
2.4.3 Creating a Conditional Request

A **conditional instruction** is a request instruction that TDMS executes only if certain conditions are true. A request containing one or more of these instructions is called a **conditional request**. Whether or not TDMS executes a conditional instruction depends on a run-time value called a **control value**. A control value is a record field that TDMS evaluates when a program calls a conditional request.

The second request you create allows you to enter certain data into fields on the form depending on the control value (marital status) of the employee.

Figure 2-6 shows the main parts of a conditional request:

- The header, which identifies the forms and records used by the request
- The base, which contains instructions that TDMS performs every time an application program calls this request
- Conditional instructions, which TDMS executes only if certain conditions are true



```
RDUDFN> FORM   IS FAMILY_FORM;
RDUDFN> RECORD IS FAMILY_RECORD;
RDUDFN> RECORD IS FAMILYREC_RECORD;
```

```
RDUDFN> DESCRIPTION /*A conditional request that
                    opens fields for input
                    depending on the marital
                    status of the employee*/;

RDUDFN> CLEAR SCREEN;
```

```
RDUDFN> CONTROL FIELD IS MARITAL_STATUS
RDUDFN> "S":
RDUDFN> DISPLAY FORM FAMILY_FORM;
RDUDFN> INPUT
RDUDFN> EMPLOYEE_NUMBER TO EMPLOYEE_NUMBER,
RDUDFN> EMPLOYEE_NAME   TO EMPLOYEE_NAME,
RDUDFN> BIRTH_DATE      TO BIRTH_DATE,
RDUDFN> MARITAL_STATUS  TO MARITAL_STATUS;


RDUDFN> "M":
RDUDFN> USE FORM FAMILY_FORM;
RDUDFN> INPUT
RDUDFN> SPOUSE_NAME       TO SPOUSE_NAME,
RDUDFN> SPOUSE_BIRTH_DATE TO SPOUSE_BIRTH_DATE;
RDUDFN> RETURN "S"        TO MARITAL_STATUS;
RDUDFN> END CONTROL FIELD;
```

```
RDUDFN> PROGRAM KEY IS GOLD "D"
RDUDFN> RETURN "Y" TO PROGRAM_REQUEST_KEY;
RDUDFN> END PROGRAM KEY;
RDUDFN> END DEFINITION;
```

Figure 2-6: Parts of FAMILY_CONDITIONAL_REQUEST

In the next sections, you create the FAMILY_CONDITIONAL_REQUEST. If RDU notifies you of errors, continue to enter the request. After you finish entering the request text, you can correct errors using the EDIT command described in the section entitled Correcting Your Errors.

To create FAMILY_CONDITIONAL_REQUEST, enter the following text:

```
RDU> CREATE REQUEST FAMILY_CONDITIONAL_REQUEST
```

When RDU displays the RDUDFN> prompt, you can begin to enter the request instructions.

2.4.3.1 Header and Base Information for the Conditional Request

— The header and base information for the conditional request is the same as for the simple request. That is, the form and records you use are the same. Enter this text:

```
RDUDFN> FORM   IS FAMILY_FORM;
RDUDFN> RECORD IS FAMILY_RECORD;
RDUDFN> RECORD IS FAMILYREC_RECORD;

RDUDFN> DESCRIPTION /*A conditional request that
                      opens fields for input
                      depending on the marital
                      status of the employee*/;

RDUDFN> CLEAR SCREEN;
```

2.4.3.2 Beginning Key Phrase

— The CONTROL FIELD IS instruction is the key phrase that begins a conditional instruction. A CONTROL FIELD IS instruction always contains a control value. The application program, or the request, places an S or an M in MARITAL_STATUS. Enter this text:

```
RDUDFN> CONTROL FIELD IS MARITAL_STATUS
```

The beginning key phrase opens the conditional instruction block; therefore, it is not followed by a semicolon (;).

2.4.3.3 Case Values — Each control value must have one or more associated **case values**. In its most basic form, a case value is a quoted string. At run time, TDMS matches the case values that you specify with the value in the control value. You can specify any number of case values in a conditional instruction. In this walkthrough, the case values are S for single and M for married. Enter this text:

```
RDUDFN> "S":
```

Within the conditional instruction block, you include request instructions in the way described earlier for the simple request; that is, at the RDUDFN> prompt. At run time, TDMS executes the following instructions if the control value, MARITAL_STATUS, contains the value S. Enter this text:

```
RDUDFN> DISPLAY FORM FAMILY_FORM;
```

2.4.3.4 INPUT TO Instruction — If you want TDMS to collect data you enter into a form field and return it to a record field, you use the INPUT TO instruction. Enter this instruction:

```
RDUDFN> INPUT
RDUDFN> EMPLOYEE_NUMBER TO EMPLOYEE_NUMBER,
RDUDFN> EMPLOYEE_NAME   TO EMPLOYEE_NAME,
RDUDFN> BIRTH_DATE      TO BIRTH_DATE,
RDUDFN> MARITAL_STATUS  TO MARITAL_STATUS;
```

2.4.3.5 USE FORM Instruction — The second case value in the conditional instruction block is M for married. At run time, TDMS executes this block of instructions only if the control value contains the value M.

Enter this text:

```
RDUDFN> "M":
```

In this block, you use the USE FORM instruction. The USE FORM instruction uses the last form displayed (with the same background text and field contents) when the previous request call ended.

You use the USE FORM instruction in this part of the request because you want the employee information that the operator entered in the previous form displayed as you enter information about the spouse. Enter this text:

```
RDUDFN> USE FORM FAMILY_FORM;
```

You can enter information on an employee's spouse only if the employee is married. Enter this text:

```
RDUDFN> INPUT
RDUDFN> SPOUSE_NAME       TO SPOUSE_NAME,
RDUDFN> SPOUSE_BIRTH_DATE TO SPOUSE_BIRTH_DATE;
```

A conditional instruction block can also return a value to the application program. After you enter information on an employee's spouse, you are finished entering data on that employee. To enter information on a new employee, the control value, MARITAL_STATUS, must be S. In other words, the program assumes the employee is single until M is entered into the MARITAL_STATUS field. Enter the text:

```
RDUDFN> RETURN "S" TO MARITAL_STATUS;
```

2.4.3.6 Ending Key Phrase — You must end each conditional instruction with the ending key phrase that matches the beginning key phrase. This phrase tells TDMS that this conditional instruction block is complete; there are no more case values to compare with the current control value.

Enter this text:

```
RDUDFN> END CONTROL FIELD;
```

2.4.3.7 PROGRAM KEY IS Instruction — When you created the request FAMILY_DISPLAY_REQUEST, you defined a key to be a PRK. This key definition allows you to communicate with the application program. You want the same PRK to appear in this request, so enter the following text:

```
RDUDFN> PROGRAM KEY IS GOLD "D"
RDUDFN> RETURN "Y" TO PROGRAM_REQUEST_KEY;
RDUDFN> END PROGRAM KEY;
```

At run time, the operator can now press GOLD-D to end the terminal session. When the operator presses the sequence GOLD-D, the value Y is returned to PROGRAM_REQUEST_KEY, a program variable. The program then checks the value of PROGRAM_REQUEST_KEY and exits from the application if it contains Y.

2.4.3.8 END DEFINITION Instruction — When you finish entering request text, enter the END DEFINITION instruction:

```
RDUDFN> END DEFINITION;
```

2.4.4 Exiting RDU

You are done creating requests. To move on to the next phase of building your TDMS application, type EXIT at the RDU> prompt.

```
RDU> EXIT
```

2.4.5 Correcting Your Errors

If you have entered all the request text, including the END DEFINITION instruction, RDU reports any problems by issuing an error message. You can correct any typing and spelling errors by using the EDIT command. Type the following command:

```
RDU> EDIT
```

The EDIT command calls the system editor. (In this manual, the EDIT command calls the VMS editor, EDT.) When you issue the EDIT command:

- RDU invokes your editor and displays the last command you entered, CREATE REQUEST, and all the request text you entered after the CREATE REQUEST command
- You can correct your typing errors just as you would in a regular text file, using all of your editor's features
- You can then exit your editor by using the appropriate exit command for that editor.
- RDU then executes the CREATE REQUEST command and checks this corrected request text for further errors

After you correct all the errors and enter the EXIT command, RDU stores this request in the CDD. If you still have errors in your request text, RDU continues to display error messages and does not store the request in the CDD. Use the EDIT command again to correct those errors that RDU identifies.

2.4.6 Creating a Request Library Definition

In this section of the walkthrough, you create a request library definition and a request library file. Both are necessary to make your requests accessible to a TDMS application program. A request library definition allows you to identify, in one place, all the requests that are used in an application or a portion of an application. A request library definition is stored in the CDD. After you create the request library definition, you must build a request library file using RDU. This request library file contains the same information as the request library definition but in a form easier for the application program to access.

To create a request library definition, enter RDU and type this text:

```
RDU> CREATE LIBRARY FAMILY_LIBRARY
```

A request library definition generally contains only two types of instructions:

- The REQUEST IS instruction, which identifies all the requests you want to list in this request library
- The FILE IS instruction, which names the VMS request library file (RLB) that you will subsequently create

In certain cases, such as when the form will be used by VAX DATATRIEVE, it can also contain the FORM IS instruction. See the chapter entitled Using VAX TDMS with VAX DATATRIEVE for more information.

Enter these request library definition instructions:

```
RDUDFN> REQUEST IS FAMILY_DISPLAY_REQUEST;
RDUDFN> REQUEST IS FAMILY_CONDITIONAL_REQUEST;
RDUDFN> FILE IS "FAMILYLIB.RLB";
RDUDFN> END DEFINITION;
```

Note that the RLB file specification, FAMILYLIB.RLB, must be enclosed in quotation marks and must conform to the rules for correct VMS file names.

RDU stores this request library definition in the CDD unless:

- One of the requests you name is not in the CDD
- Syntax errors exist

RDU tells you which request it cannot find. You should check to see that you have entered the correct names for your requests. A spelling or typing error can result in RDU being unable to find a request in the CDD.

If you have made an error, use the EDIT command, as described earlier in this chapter.

Note

You can change the text of any of the requests named in the CDD request library definition without changing the request library definition itself. Because the request library definition contains only the names of your requests, it is not affected by changes you make to text in those individual requests. However, you will have to rebuild the request library file.

2.4.7 Building a Request Library File

The form, two records, and two requests you created are now in your CDD directory. Your CDD directory also contains a request library definition that names both requests.

To build a request library file, enter this text:

```
RDU> BUILD LIBRARY FAMILY_LIBRARY
```

RDU searches the CDD for the request library definition, FAMILY_LIBRARY, as you specified in the BUILD LIBRARY command. It extracts the requests listed in that library from the CDD and places them in the VMS request library file, FAMILYLIB.RLB, that you specified in the request library definition. It also stores the forms used by the request and information about the records used in the request.

2.5 Writing the Application Program

To use the requests that you created, you need to write an application program. Within the program you include calls, which are instructions that TDMS follows to perform tasks, such as displaying a form on the screen. The program you create here uses the following five primary TDMS calls:

- TSS$OPEN_RLB

  To open a request library file

- TSS$OPEN

  To open a channel to the terminal for input and output

- TSS$REQUEST

  To execute a request to display a form and transfer data between a form or a request and a record

- TSS$CLOSE_RLB

  To close the request library file

- TSS$CLOSE

  To close the channel to the terminal

This program also uses three additional calls:

- TSS$SIGNAL

  To receive more information about the previous TDMS call

- TSS$READ_MSG_LINE

  To write a prompt and read a message from the reserved message line (line 24) of the screen

- TSS$WRITE_MSG_LINE

  To write a message on the reserved message line (line 24) of the screen

See the *VAX TDMS Reference Manual* for more information about these calls.

There are two general concepts about TDMS synchronous calls that you should understand:

- Each call will complete before control returns to the program.
- TDMS returns a standard VAX condition code to the program after a call, so you should define a variable to receive the return status code.

This example of the calls is presented in VAX BASIC. The calls are presented in the order that you use them in your program. Following the presentation of the calls is the BASIC program that drives your application.

2.5.1 Declaring Records

This walkthrough uses two record definitions that your program has to know about. Since you are using BASIC, which supports extraction of records from the CDD, you can use simple statements in the source program to bring the record definitions into your program. These statements in BASIC are:

```
%INCLUDE %FROM %CDD 'FAMILY_RECORD'
%INCLUDE %FROM %CDD 'FAMILYREC_RECORD'
```

You can then declare program variables with those record structures using the MAP statement in BASIC. For example:

```
MAP (FAMILY_BUF) FAMILY_RECORD FAM
MAP (FAMILYREC_BUF) FAMILYREC_RECORD FAMREC
```

2.5.2 Opening a Request Library File – TSS$OPEN_RLB

The next step is to open the request library file FAMILYLIB.RLB that you created in the section entitled, Building a Request Library File. At run time, TDMS uses this file to access a request and forms.

TSS$OPEN_RLB should be the first call in a program because if the request library file does not exist or is not accessible, TDMS cannot use requests to transfer data between the form and program. The code to open the request library file in this application is:

```
RET_STATUS = TSS$OPEN_RLB ("FAMILYLIB.RLB", &
             LIBRARY_ID BY REF)
```

2.5.3 Opening a Channel – TSS$OPEN

You must open a TDMS channel to the terminal before you can perform any I/O on that terminal. TDMS assigns a unique number for each channel that you open. The program must then pass that channel number to future TDMS calls to identify which terminal to use. The code to open a channel for input and output is:

```
RET_STATUS = TSS$OPEN (CHANNEL BY REF)
```

Note

The channel number returned is not the VMS channel number returned by the SYS$ASSIGN service. Input/output calls to other systems (such as VAX RMS or $QIO) *should not* be issued to the terminal.

2.5.4 Transferring Data and Displaying the Form - TSS$REQUEST

Now you call the requests you created to transfer data and display the form. When the program issues a request call, TSS$REQUEST reads and executes the instructions in the request. For example, the code to issue a request call for the FAMILY_DISPLAY_REQUEST is:

```
RET_STATUS = TSS$REQUEST (CHANNEL BY REF,         &
                          LIBRARY_ID BY REF,      &
                          "FAMILY_DISPLAY_REQUEST", &
                          FAM BY REF,             &
                          FAMREC BY REF)
```

Note that the order of the records passed on the call is the same as the order of the records declared in the request.

2.5.5 Closing the Request Libary File - TSS$CLOSE_RLB

When you finish using the requests in a request library file, close the request library file with the TSS$CLOSE_RLB call. The code to close a request library file is:

```
RET_STATUS = TSS$CLOSE_RLB (LIBRARY_ID BY REF)
```

2.5.6 Closing a Channel - TSS$CLOSE

When you finish using a channel opened by TDMS, close the channel with the TSS$CLOSE call. TSS$CLOSE releases all TDMS resources associated with that terminal. After you close a channel, you cannot issue any more TDMS calls that may do input or output on that channel unless you issue another TSS$OPEN call on that terminal. The code to close a channel is:

```
RET_STATUS = TSS$CLOSE &
           (CHANNEL BY REF)
```

2.5.7 Using TDMS Calls in a BASIC Program

Figure 2-7 is the TDMS application program that uses all of the primary TDMS calls.

To create the file for the application program, enter the following text at DCL level:

```
$ EDIT/EDT FAMILY.BAS
```

To create the application program, type in the following text:

```
1       OPTION TYPE = EXPLICIT

        ON ERROR GOTO ERROR_HANDLER

        DECLARE INTEGER      &
                RET_STATUS,  &
                CHANNEL,     &
                LIBRARY_ID,  &
                ANSWER_LEN,  &
                WE_ARE_DONE, &
                NUMBER

        DECLARE INTEGER CONSTANT             &
                TRUE = -1%,                  &
                RECORD_NOT_FOUND = 155%,     &
                DUPLICATE_KEY_DETECTED = 134%

        DECLARE STRING ANSWER

        EXTERNAL INTEGER FUNCTION  &
                TSS$OPEN_RLB,      &
                TSS$OPEN,          &
                TSS$REQUEST,       &
                TSS$CLOSE,         &
                TSS$CLOSE_RLB,     &
                TSS$SIGNAL,        &
                TSS$TRACE,         &
                TSS$READ_MSG_LINE, &
                TSS$WRITE_MSG_LINE

        %INCLUDE %FROM %CDD 'FAMILY_RECORD'
        %INCLUDE %FROM %CDD 'FAMILYREC_RECORD'

        MAP (FAMILY_BUF) FAMILY_RECORD FAM
        MAP (FAMILY_BUF) STRING FILL_BUFFER_WITH_BLANKS = 61
        MAP (FAMILYREC_BUF) FAMILYREC_RECORD FAMREC
        FILL_BUFFER_WITH_BLANKS = SPACE$(LEN(FILL_BUFFER_WITH_BLANKS))

        RET_STATUS = TSS$OPEN_RLB &
                ("FAMILYLIB.RLB",LIBRARY_ID BY REF)

        GOSUB CHECK_TDMS_STATUS
```

(continued on next page)

Figure 2-7: BASIC Program Illustrating Primary TDMS Calls

```
100     RET_STATUS = TSS$OPEN(CHANNEL BY REF)

        GOSUB CHECK_TDMS_STATUS

        OPEN "FAMILY.DAT" AS FILE #1        &
                ,ORGANIZATION INDEXED FIXED &
                ,RECORDTYPE LIST            &
                ,ACCESS MODIFY              &
                ,ALLOW MODIFY               &
                ,MAP FAMILY_BUF             &
                ,PRIMARY KEY FAM::EMPLOYEE_NUMBER

200 MAIN_LOOP:

        UNTIL WE_ARE_DONE
                RET_STATUS = TSS$READ_MSG_LINE                  &
                        (CHANNEL BY REF,                        &
                        ANSWER BY DESC,                         &
                        "Do you wish to add or display a form? ", &
                        ANSWER_LEN BY REF)

                ! Change the response to uppercase
                ! and remove blanks or tabs.

                SELECT EDIT$(ANSWER, 32 + 2)
                   CASE "ADD"
                        GOSUB ADD_RECORD

                   CASE "DISPLAY"
                        GOSUB DISPLAY_RECORD

                END SELECT

        NEXT

        GOTO END_OF_PROGRAM

300 ADD_RECORD:

                ! Initialize the buffer.
        FILL_BUFFER_WITH_BLANKS = SPACE$(LEN(FILL_BUFFER_WITH_BLANKS))

                ! First, assume the person is single.
        FAM::MARITAL_STATUS = "S"

310
        RET_STATUS = TSS$REQUEST (CHANNEL BY REF,                    &
                LIBRARY_ID BY REF,                                   &
                "FAMILY_CONDITIONAL_REQUEST",                        &
                FAM BY REF,                                          &
                FAMREC BY REF)

        GOSUB CHECK_TDMS_STATUS
                ! If the person is married, return and gather
                ! their spouse's name.
```

(continued on next page)

Figure 2-7: BASIC Program Illustrating Primary TDMS Calls (Cont.)

```
        GOTO 310 IF FAM::MARITAL_STATUS = "M"

        PUT #1
        RETURN

400 DISPLAY_RECORD:

        RET_STATUS = TSS$READ_MSG_LINE                  &
                (CHANNEL BY REF,                        &
                ANSWER BY DESC,                         &
                "Type in the number of the record ",    &
                ANSWER_LEN BY REF)

        GOSUB CHECK_TDMS_STATUS

        FAM::EMPLOYEE_NUMBER = VAL%(ANSWER)
        GET #1, KEY #0 EQ FAM::EMPLOYEE_NUMBER

        RET_STATUS = TSS$REQUEST (CHANNEL BY REF,  &
                LIBRARY_ID BY REF,                 &
                "FAMILY_DISPLAY_REQUEST",          &
                FAM BY REF,                        &
                FAMREC BY REF)

        GOSUB CHECK_TDMS_STATUS

       RETURN

500 CHECK_TDMS_STATUS:

        IF (RET_STATUS AND 1%) = 0%
        THEN        RET_STATUS = TSS$SIGNAL
                    CALL LIB$STOP (RET_STATUS BY VALUE)
        END IF

                ! Check to see if the user pressed
                ! the program request key.

        IF FAMREC::PROGRAM_REQUEST_KEY = "Y"
        THEN WE_ARE_DONE = TRUE
        END IF

        RETURN

18000   ERROR_HANDLER:

                ! Display the error message, then resume
                ! to the appropriate subroutine.

        RET_STATUS = TSS$WRITE_MSG_LINE  &
                      (CHANNEL BY REF, &
                      ERT$(ERR))
        SLEEP 3
```

(continued on next page)

Figure 2-7: BASIC Program Illustrating Primary TDMS Calls (Cont.)

```
        SELECT ERR
           CASE DUPLICATE_KEY_DETECTED
                RESUME 300

           CASE RECORD_NOT_FOUND
                RESUME 400

           CASE ELSE
                ON ERROR GOTO 0
         END SELECT


32000   END_OF_PROGRAM:

        RET_STATUS = TSS$CLOSE_RLB(LIBRARY_ID BY REF)

        GOSUB CHECK_TDMS_STATUS

        RET_STATUS = TSS$CLOSE(CHANNEL BY REF)

        GOSUB CHECK_TDMS_STATUS

32767   END
```

Figure 2-7: BASIC Program Illustrating Primary TDMS Calls (Cont.)

2.6 Compiling the TDMS Program

You compile a TDMS program just as you do any other program.

Issue the following command at DCL level:

```
$ BASIC FAMILY.BAS
```

When you compile a program and there are no errors, the compiler generates another file in your default directory with the same name and a file type .OBJ.

2.7 Linking the TDMS Program

The TDMS program interface is a shareable image named TSSSHR.EXE. When the TDMS software is installed, TSSSHR.EXE is placed in the VAX/VMS shareable image symbol table library, SYS$LIBRARY:IMAGELIB.OLB.

When you link any program, the VAX Linker searches, by default, SYS$LIBRARY:IMAGELIB.OLB.

You can link a TDMS program simply by issuing the following command at DCL level:

```
$ LINK FAMILY
```

2.8 Running the TDMS Program

You can run this program simply by issuing the following command at DCL level:

```
$ RUN FAMILY
```

Using the Form Definition Utility (FDU) and the Form Editor 3

This chapter describes some important FDU commands, including those that let you to use the form editor, and provides an overview of the form editor. The rest of this manual provides more detailed explanation. Complete information about the syntax of all FDU commands can be found in the *VAX TDMS Reference Manual.*

FDU allows you to create, modify, and store customized form definitions. The product of your work with the form editor is a form definition, which FDU stores in the Common Data Dictionary (CDD). The form definition contains the information that identifies:

- Screen image of the form. The screen image includes the location of background text and fields as well as video highlighting. (**Background text** is text that is always displayed when the form is displayed; **fields** are locations on the form where data can be collected or displayed.)
- Length and data type of each field.
- Set of attributes for each field on the form (including means for validating data).
- Location of scrolled regions on the form.
- Name of a Help form, which the operator can display at run time.

3.1 Entering FDU

To enter FDU from DCL level, type the command:

```
$ RUN SYS$SYSTEM:FDU.EXE
```

You can set up a global symbol in your login command file to allow you to enter FDU commands at DCL level. For example:

```
$ FDU :== $FDU
```

When you enter FDU, the FDU> prompt is displayed on your terminal. When the FDU> prompt is displayed, you can give only valid FDU commands.

This chapter discusses the use of several FDU commands that you will use most frequently, including those that allow you to use the form editor and create a listing that provides information about the form definition and objects in the CDD.

3.2 Leaving FDU

To leave FDU, type EXIT or press CTRL/Z. Both of these commands will return you to DCL level. For example:

```
FDU> EXIT
$
```

or

```
FDU> <CTRL/Z>
$
```

3.3 Using the Form Editor

You can use the form editor only when you issue one of the following commands at FDU level:

- CREATE FORM
- MODIFY FORM
- REPLACE FORM

All FDU commands can be abbreviated to their shortest unambiguous form.

3.3.1 Creating a New Form Definition

To create a new form definition, enter FDU. At the FDU> prompt, issue the command CREATE FORM followed by the form name that you want to assign to the form definition. The form name is generally the CDD path name of the form definition; it is stored in your default CDD directory unless you explicitly state a different location.

The given name is the last name you type when referring to the form. It is usually the same as the final name of the CDD path name. However, from DCL level, you can assign a logical name to the given name of the form. If you do, the final name of the object in the CDD can be different from the given name of the form. For instance, if you use the name EMPLOYEE_FORM for your form, and you define, at DCL level, EMPLOYEE_FORM to be EMPLOYEE_FORMS.EMPLOYEE_ONE, the final name of the form in the CDD is EMPLOYEE_ONE.

For example, to create a form named CDD$TOP.ACCOUNTING.PAYROLL_FORM when your CDD$DEFAULT is defined as CDD$TOP.ACCOUNTING, type:

```
FDU> CREATE FORM PAYROLL_FORM
```

(You can abbreviate this command to a few characters, such as CR FO PAYROLL_FORM.)

The form definition that you create is stored in the CDD location CDD$TOP.ACCOUNTING.PAYROLL_FORM.

After you issue the CREATE FORM command, FDU calls the form editor and displays the form editor Phase Selection menu.

Note

If you do not want to use the form editor with the CREATE FORM and REPLACE FORM commands, then be sure to use the /FORM_FILE qualifier. This qualifier allows you to use a file, generated by the VAX FMS form editor, that contains a form definition.

If you do not use the /FORM_FILE qualifier on a CREATE FORM or REPLACE FORM command, the form editor will be invoked automatically.

You can then create the form definitions as described later in this manual. When you issue the form editor command that saves the finished form definition, FDU stores it in the CDD location specified by the form name.

3.3.2 Modifying an Existing Form Definition

You may wish to make changes or additions to a form definition already stored in the CDD. The MODIFY FORM command allows you to enter the form editor and make any desired changes or additions without having to recreate the form definition.

For example, to modify the existing form definition PAYROLL_FORM, issue the command:

```
FDU> MODIFY FORM PAYROLL_FORM
```

(You can abbreviate this command to MO FO PAYROLL_FORM.)

FDU places the form definition PAYROLL_FORM in the form editor. You can then use the form editor to make any desired changes.

When you have made the desired changes and additions to the form definition, FDU stores the modified form with the same name and in the same CDD location as the original form; the original form definition is deleted.

The form definition that you identify in the MODIFY FORM command must exist in the CDD. If you provide a CDD path name that does not exist or does not contain a form, FDU gives an error message and returns you to FDU level.

3.3.3 Making Changes After Modifying a Form Definition

If you modify (or replace) a form definition after a request that uses the form has been created, it may be necessary to revise the request. If you modify a form definition, keep in mind:

- Changes that were made to the form definition
- Time that the changes were made

If you modify or replace a form definition after a request library file (RLB) that uses the form has been built, it will always be necessary to rebuild the RLB file.

3.3.4 Replacing an Existing Form Definition

If you wish to replace a form definition that already exists in the CDD with a new form, use the REPLACE FORM command. For example, to create a new form to replace the form definition PAYROLL.INFORMATION_FORM, issue the command:

```
FDU> REPLACE FORM PAYROLL.INFORMATION_FORM
```

(You can abbreviate this to REP FO PAYROLL.INFORMATION_FORM.)

When you issue the REPLACE FORM command, FDU calls the form editor with no form defined. After you finish creating the new form definition, the newly created form definition is stored in PAYROLL.INFORMATION_FORM, and the old form is deleted. The MODIFY FORM and REPLACE FORM commands are similar, since both commands call the form editor and replace an existing form definition.

When you issue the MODIFY FORM command, the form editor begins by placing the existing form that you name in the form editor buffer. When you issue the REPLACE FORM command, the form editor begins with no existing information in the form editor buffer.

3.3.5 Five Phases of the Form Editor

When you issue FDU commands to create, modify, or replace a form definition, you ordinarily enter the form editor. The form editor has five phases, each of which has a distinct function:

1. Form phase

   In the Form phase, you define characteristics that apply to the entire form definition or to all fields that you create on the form. In this phase, you can specify:

   - Screen background for the form (dark, light, or current screen)
   - Screen width (allowing 80-column or 132-column forms)
   - Video highlighting for input fields (that determine the run-time video attributes of a field available for input)
   - Name of a Help form to be associated with the form
   - Default field attribute assignments for the form

2. Layout phase

   In the Layout phase of the form editor, you create the overall appearance of the form and identify the fields on the form. You use the Layout phase to specify:

   - Location and text of all background text
   - Location, length, and picture type of fields
   - Location and number of lines of scrolled regions
   - Video highlighting for any areas of the form

The Layout phase is one of two required phases of the form editor. (The other is the Exit phase.) You must explicitly include background text and/or fields during the Layout phase to have a meaningful form definition.

3. Assign phase

   In the Assign phase, you can assign particular attributes to individual fields that you identified in the Layout phase. The most important attribute assigned in this phase is the Field Name, since form field names must be associated, or mapped, to record field names in order to transfer data between a form and a record. Other attributes that can be assigned to each field:

   - Provide a default value for the field, which is displayed in the field at run time (only if a request does not map data into the field)
   - Provide a help message, which the operator can read at run time
   - Affect the appearance of the field and of data entered into the field
   - Provide special validation procedures, called field validators, which can specify a list or range of valid entries or apply a predefined check digit algorithm to data entered by the operator at run time

4. Order phase

   In the Order phase, you specify the order in which the operator accesses the fields for input at run time.

   Use the Order phase to change the access order or to determine the current field access order.

5. Exit phase

   In the Exit phase, you either save the form that you have just defined (storing it in the CDD) or you choose not to save it. The Exit phase of the form editor is always the final phase.

Note

If you entered the form editor using either the REPLACE FORM or MODIFY FORM command and choose not to save the newly-defined form, the original form definition (that would have been replaced or modified) is *not* deleted from the CDD.

After completing the Exit phase, TDMS returns you to FDU level and displays the FDU> prompt.

3.4 Using Other FDU Commands

In addition to the three FDU commands that enable you to use the form editor, other especially useful FDU commands are:

- COPY FORM

  Makes a copy of a form definition within the CDD.

- LIST FORM

  Produces a listing of important information about the form definition.

- DELETE FORM

  Deletes a form definition from the CDD.

These commands are described in the following sections.

Other commands that you can issue at the FDU> prompt, as well as complete information concerning syntax, qualifiers, defaults, and usage for all FDU commands are found in the *VAX TDMS Reference Manual.*

3.4.1 Copying a Form Definition in the CDD

Use the COPY FORM command in FDU to copy an existing form definition in the CDD and store it in a new CDD location.

For example, to make a copy of the form definition named PAYROLL_FORM, issue the command:

```
FDU> COPY FORM PAYROLL_FORM FINANCE_FORM
```

This creates a new form definition named FINANCE_FORM, stored in your current default CDD directory with the name FINANCE_FORM. This form definition is identical to PAYROLL_FORM.

Use the COPY FORM command to save time when you are creating similar forms. You can create a single form, use the COPY FORM command to generate copies, and then make any modifications needed using the MODIFY FORM command.

3.4.2 Listing Information About a Form Definition

You need to know a great deal of information about the form definition when creating requests or if you are creating record definitions after you create form definitions. Use the LIST FORM command in FDU to generate this information.

The LIST FORM command provides the following:

- Form name
- Formwide attributes, including:
  - Screen width
  - Screen background
  - Video characteristics for input fields
  - CDD path name of the Help form
  - Date and time that the form was stored in the CDD
  - Field access order list
- Information for each field on the form, including:
  - Field name
  - Field picture characters
  - Length
  - Default value
  - Help text
  - Attributes
  - Field validators
  - Video highlighting
  - Scrolling
  - Indexing

You usually need all of this information *before* creating the requests that use the form. If records are defined after the form, you need to know the names, picture types, and lengths of the form fields.

To create a file that lists this information about the form definition, use the LIST FORM command, giving the form name and the /OUTPUT qualifier. For example, this command directs the form definition information to a file named PAYROLL.TXT in your default VMS directory:

```
FDU> LIST FORM PAYROLL_FORM/OUTPUT=PAYROLL.TXT
```

If you do not use the /OUTPUT qualifier, the form definition information in PAYROLL_FORM is displayed on your terminal (or to the device designated as SYS$OUTPUT).

You can send the output of the LIST FORM command directly to your default line printer by using the /PRINT qualifier with the LIST FORM command. For example:

```
FDU> LIST FORM PAYROLL_FORM/PRINT
```

This command creates a listing of information about the form definition PAYROLL_FORM that is sent to the default line printer.

You may use both the /OUTPUT and /PRINT qualifiers on the LIST FORM command to create a file and a line printer listing. For example:

```
FDU> LIST FORM PAYROLL_FORM/OUTPUT=PAYROLL.TXT/PRINT
```

Issuing this command does the following:

1. Writes the information about PAYROLL_FORM to a file named PAYROLL.TXT
2. Sends the file PAYROLL.TXT to the line printer

3.4.3 Deleting a Form Definition

You can delete a form definition from the CDD by using the DELETE FORM command. For example, to delete the form PAYROLL_FORM from the CDD, issue the command:

```
FDU> DELETE FORM PAYROLL_FORM
```

Assigning Formwide Attributes 4

In the Form phase, you assign attributes that apply to the entire form. This chapter discusses the use of features that are available in this phase.

4.1 Introduction to the Form Phase

Use the Form phase to:

- Set the screen background of the form
- Set the width of the form
- Assign video attributes for fields when they are available for input
- Identify a Help form
- Assign default field attributes

To assign attributes in the Form phase, move the cursor and denote your selections with a single character or a string. When in the Form phase, you can use the function keys listed in Table 4-1.

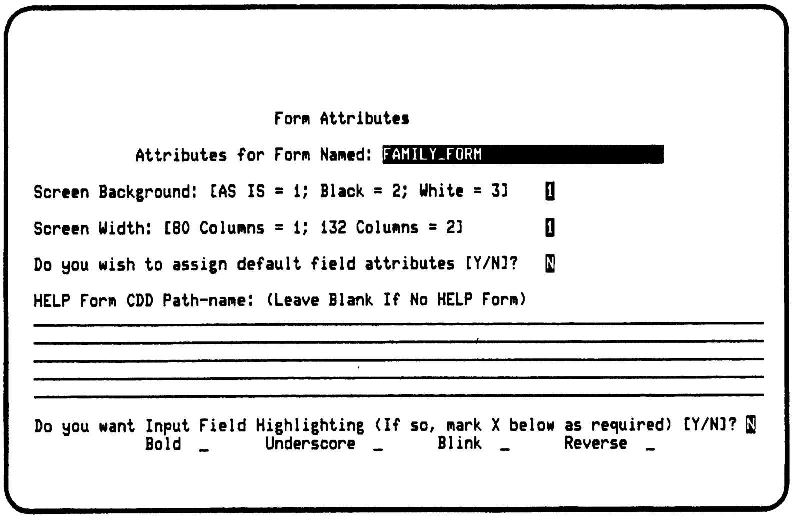
Table 4-1: Form Phase Function Keys

Key	Function
TAB	Moves the cursor to the next field.
BACK SPACE (or F12)	Moves the cursor to the previous field.
LINE FEED (or F13)	Deletes contents of the current field.
RETURN	Saves all current field values and returns you to the Phase Selection menu or saves all current values and displays the Default Field Attribute form.
MENU (GOLD-KP7)	Saves all current field values and returns you to the Phase Selection menu.
HELP (PF2 or F15)	Displays a line of help information at the bottom of the screen.

The following section describes the use and meaning of the parts of the Form phase.

4.2 Entering the Form Phase

You can enter the Form phase only from the Phase Selection menu. (The Phase Selection menu is displayed when you enter the form editor after issuing the CREATE FORM, MODIFY FORM, or REPLACE FORM command at FDU level.) At the Phase Selection menu, type FORM (or just the abbreviation F) and press RETURN. The Form phase screen is then displayed, as shown in Figure 4-1.



```
                         Form Attributes

              Attributes for Form Named: FAMILY_FORM

Screen Background: [AS IS = 1; Black = 2; White = 3]    1

Screen Width: [80 Columns = 1; 132 Columns = 2]         1

Do you wish to assign default field attributes [Y/N]?   N

HELP Form CDD Path-name: (Leave Blank If No HELP Form)

Do you want Input Field Highlighting (If so, mark X below as required) [Y/N]? N
          Bold _       Underscore _       Blink _       Reverse _
```

Figure 4-1: Form Phase Screen

To leave the Form phase of the form editor, press either RETURN or GOLD-KP7. If you press RETURN and have chosen to assign default field attributes, the Default Field Attribute form is displayed; otherwise, the Phase Selection menu is displayed. (Press RETURN to reach the Phase Selection menu from the Default Field Attribute form.)

4.2.1 Setting the Screen Background

During the Form phase, you determine the screen background that the operator sees at run time. You can specify one of the following:

1. As Is (Default)

   TDMS does not change the screen background on the terminal. If the operator's screen is set to dark background, the run-time form has dark background. If the operator's screen is set to light background, the run-time form has light background.

2. Black

   The run-time form is displayed with black (dark) background.

3. White

   The run-time form is displayed with white (light) background.

Choose the screen background for the form by entering the corresponding number in the Screen Background field or accept the current value (as displayed) and move to the next field by pressing the TAB key.

4.2.2 Setting the Screen Width

The screen width setting allows you to specify whether the form will be 80 columns wide or 132 columns wide form. The 132-column form allows you to include more fields on the form than an 80-column form; however, the larger characters on the 80-column form are easier to read. You can achieve an especially readable form by using a 132-column form (assigned in the Form phase) and double-wide lines (assigned in the Layout phase). Characters displayed with this configuration are about 20% larger than characters displayed in 80-column mode with normal-width lines.

To set the screen width, enter either 1 or 2 at the Screen Width field depending on the size you need:

1. 80 Columns

   The form editor is set to 80-column mode and the run-time form has 80 columns.

2. 132 Columns

   The form editor is set to 132-column mode and the run-time form has 132 columns.

4.2.3 Assigning Default Field Attributes

Field attributes are special characteristics that you can apply to form fields during the Assign phase. Chapter 6 of this manual discusses the use and meaning of field attributes.

Note

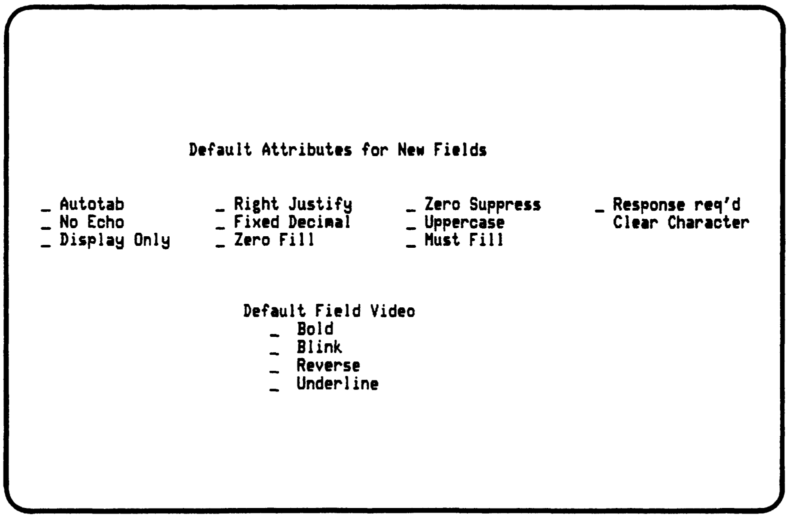
Avoid assigning default field attributes if you do not understand their use and meaning. You can leave the Form phase and not assign default field attributes by pressing RETURN.

In the Form phase, you can specify field attributes that are assigned by default to any field you subsequently create in the form while in the current FDU session. Any field attribute that has been assigned by default in the Form phase can be deassigned in the Assign phase. Similarly, you can assign any attribute in the Assign phase, regardless of the default specified in the Form phase.

To assign default field attributes, type Y at the question on the Form Attributes form:

```
"Do you wish to assign default field attributes?"
```

When you complete the Form phase by pressing RETURN, the form editor displays a list of field attributes. To return to the Phase Selection menu, press GOLD-KP7. Figure 4-2 shows the Default Attributes for New Fields form.



```
                 Default Attributes for New Fields

_ Autotab          _ Right Justify     _ Zero Suppress     _ Response req'd
_ No Echo          _ Fixed Decimal     _ Uppercase           Clear Character
_ Display Only     _ Zero Fill         _ Must Fill

                       Default Field Video
                         _  Bold
                         _  Blink
                         _  Reverse
                         _  Underline
```

Figure 4-2: Default Attributes for New Fields Form

Enter X next to those attributes that you want to assign by default; press the space bar when the cursor is next to those attributes that you want to deassign. For Default Value and Help Text selections, type in the text you wish to be displayed by default for each field.

Remember that the default field attributes that you specify apply *only* to fields that you create later in the Layout phase. If you leave FDU, save the form, and then later modify it, any new fields will not have the defaults. You do not affect the attributes of any fields that have already been created when you specify default field attributes.

4.3 Help Forms

Help forms are forms that provide information to the operator at run time. They are created in FDU and stored in the CDD. Help forms should include only background text since you cannot map any fields on a Help form for input or output.

At run time, if the cursor is located at a field that has a help message, the help message is displayed the first time that the operator presses the HELP key (PF2 or F15). This message is displayed on the last line of the screen and is created in the Assign phase. See chapter 6, Assigning Field Attributes and Validators, for more information. When the operator presses the HELP key a second time, the entire screen is cleared (regardless of the screen area used by the Help form) and a Help form is displayed.

Each Help form can have one Help form assigned to it in the Form phase. These additional forms are displayed each time you press the HELP key.

You can have a single Help form for any form and a single help message for each field on a form.

If your form has an associated Help form, enter the name of the Help form. It is good practice to use the full CDD path name (CDD$TOP.PAYROLL.HELPFORM, for example) when identifying a Help form.

If the name of your Help form continues beyond one line, you can continue typing and the text automatically wraps to the next line. Whenever the name of a Help form is on more than one line, FDU concatenates all of the text.

4.3.1 Assigning Input Field Highlighting

VAX TDMS allows you to specify video highlighting for fields when they are available (open) for input at run time. The video attributes that you assign for input fields during the Form phase *replace* other video attributes that you may assign to the field during the Layout phase but *only* when the field is the current input field. If the cursor is not in the current input field, the video attributes assigned in the Layout phase override those set in the Form phase.

Video characteristics set by a request override those set during the Form phase.

The video highlights available include:

- Blink
- Bold
- Reverse
- Underline

To assign video highlights, type Y to the question on the Form Attributes form:

```
"Do you want Input Field Highlighting"
```

Mark the appropriate video highlights with an X. The cursor moves to the next video highlight when you press:

- X to assign the video highlights
- Space bar to deassign the video highlights
- TAB key to keep the current value

You can choose more than one video highlight attribute, or you can choose none. Table 4-2 provides several examples that demonstrate the run-time effect of input field video highlighting.

Table 4-2: Examples of Input Field Highlighting

Input Highlighting Activated (Form Phase)	Input Field Highlights (Form Phase)	Other Video Highlights (Layout Phase)	Run-time result when the cursor is in the field
Yes	Bold	Reverse	Bold
Yes	Blink	Blink, Bold, Reverse	Blink
Yes	None	Blink, Underline	None
No	None	Underline	Underline
No	Blink, Bold	Underline	Underline

To remove all video attributes from a field when it is open for input, leave each of the video characteristics blank after typing Y to the question:

```
"Do you want Input Field Highlighting?"
```

Determine the screen background, screen width, and video highlights that will render the most legible run-time form. The readability of the form that you define can have a significant impact on the accuracy and productivity of operators.

Laying Out the Form 5

In the Layout phase of the form editor, you create the screen image of the form and the fields that are included on the form. Specifically, you identify:

- Location and text of all background text
- Location, length, and picture type of all fields
- Scrolled regions
- Video attributes that are displayed whenever the form is displayed, including:
  - Video highlights (bold, blink, underline, and reverse)
  - Lines that are double-wide or double-size
  - Rectangles or lines drawn by the form editor

5.1 Entering the Layout Phase

To enter the Layout phase, type the word LAYOUT, or use the abbreviation L, when the Phase Selection menu is displayed. When you press RETURN, the form editor displays the Layout phase screen.

You can enter the Layout phase *only* from the Phase Selection menu.

5.2 Leaving the Layout Phase

MENU key (GOLD-KP7)	Press the MENU key to exit from the Layout phase and return to the Phase Selection menu. To use the MENU key, press the sequence GOLD-KP7. You can use the MENU key at any time except when there is an active select range. To leave the Layout phase when a select range is active, first press RESET (GOLD-KP period) and then press MENU.

5.3 Layout Phase Screen

The Layout phase screen has 24 lines:

- Lines 1 to 23 show the current background text, picture characters, field constants, and video highlights.
- Line 24 is a message line that displays the cursor status line, help messages, date or time formats, and other information.

When you attempt a Layout phase operation that is not valid, the form editor signals an error by ringing the terminal bell. You can get specific information about the error that you have just made by pressing the HELP key on your terminal. If you defined the screen background to be black or white in the Form phase, you can press the sequence GOLD-Q to cause the form editor to signal errors by reversing the screen background (rather than ringing the terminal bell). If you defined the screen to be "As is", you cannot use the sequence GOLD-Q.

The **cursor status line** is always displayed on the 24th line of the screen during the Layout phase. When you create a new form and enter the Layout phase, the first 23 lines are blank and the cursor status line looks like this:

```
Cursor TXT NOR Line 1 Column  1 Modes TXT OVS
```

Each of the blocks of text on the cursor status line provides information that is useful during the creation of the form definition. The following section explains the meaning of each block.

Cursor

The two blocks of text to the right of Cursor provide information about what already exists (in terms of background text or fields and scrolled or nonscrolled lines) at the location where the cursor is positioned.

TXT	When the first Cursor block reads TXT, the cursor is on a character (or blank space) that is currently defined as background text.
FLD	When the first Cursor block reads FLD, the cursor is on a character that is currently defined as belonging to a field.

Every space on a form is either background text or part of a field.

NOR	When the second Cursor block reads NOR, the cursor is on a character that is on a nonscrolled (normal) line.
SCR	When the second Cursor block reads SCR, the cursor is on a character that is on a scrolled line.

Every line on a form is defined as either normal or scrolled; the default is normal.

Line

The number in the Line block indicates the line (row) on which the cursor is currently positioned. If you press down arrow, the cursor moves down one line and the line number increases by one. You cannot move the cursor above Line 1 or below Line 23.

Column

The number in the Column block indicates the column on which the cursor is currently positioned. If you press right arrow, the cursor moves one column to the right and the column number increases by one. The column range is from 1 to 80 (on an 80-column form) or from 1 to 132 (on a 132-column form).

Modes

The two blocks to the right of Modes provide information about whether you are typing a field or background text, and how characters will appear as they are typed on the screen.

TXT	When the first Modes block reads TXT, anything that you type is background text. You can type any character when you are in Text mode (TXT).
FLD	When the first Modes block reads FLD, anything that you type is part of a field. You can type only valid picture characters, field constants, or spaces when you are in Field mode (FLD).

The form editor is always in either Text mode (TXT) or Field mode (FLD). The default is Text mode.

OVS When the second Modes block reads OVS, the form editor is in Overstrike mode. In Overstrike mode, if you type a character with the cursor at a position where another character already exists, you replace that character. For example, say you are in Overstrike mode with the cursor at the letter "s" in the word "seven." If you type "eight," the word "seven" is erased from the form, and the word "eight" replaces it.

If you are in Overstrike mode and type background text over picture characters, the field or portion of a field represented by the characters is deleted. Similarly, typing picture characters over background text replaces the background text with a field, if you are in FLD mode.

INS When the second Modes block reads INS, the form editor is in Insert mode. In Insert mode, typing a character with the cursor at a position where another character already exists moves the existing character to the right. For example, say you are in Insert mode with the cursor at the letter "s" in the word "seven". If you type "eight," the word "seven" is moved five spaces to the right. The result is the word "eightseven".

The form editor is always in either Overstrike mode or Insert mode. The default is Overstrike mode.

When the cursor is positioned on an existing field, the cursor status line also includes a **Field-name block**, to the right of the second Modes (INS/OVS) block. The Field-name block displays the current name of the field on which the cursor is positioned. If no field name has been assigned, the default field name assigned by the form editor (F$NNNN) is shown. You learn about field names and how to assign them in Chapter 6 of this manual. When the cursor is positioned on any location that is not an existing field, the Field-name block is not displayed.

The Field-name block is especially useful when reviewing a form definition in the Layout phase because it provides you with the current field name. Additionally, if you define adjacent fields, the Field-name block shows you where one field ends and the second begins. (You learn how to define adjacent fields later in this chapter.)

5.4 Layout Phase Keypad and Function Keys

In the Layout phase, you use the main keyboard to type background text and picture characters, and you use the keypad (to the right of the main keyboard) to activate various features of the form editor. Figure 5-1 shows the keypad for the Layout phase.

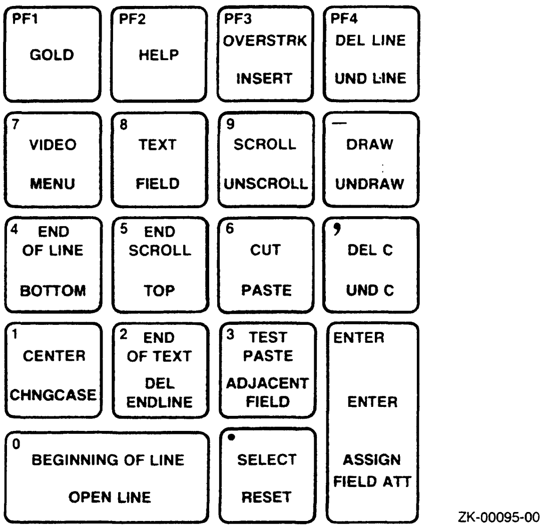


Figure 5-1: Layout Phase Keypad

To use function keys shown in the lower half of the key, first press the GOLD key and then press the keypad key.

In addition to the keypad function keys, there are several other function keys available in the Layout phase as shown in Table 5-1.

Table 5-1: Layout Phase Function Keys

Key	Function
LINE FEED (F13)	Moves down one line.
BACK SPACE (F12)	Moves backward one character.
HELP (PF2 or F15)	Displays the Help form for the Layout phase.
Right arrow	Moves forward one character.

(continued on next page)

Table 5-1: Layout Phase Function Keys (Cont.)

Key	Function
Left arrow	Moves backward one character.
Up arrow	Moves up one line.
Down arrow	Moves down one line.
CTRL/U	Deletes to beginning of line.
CTRL/R	Refreshes screen.
CTRL/W	Refreshes screen.
GOLD-D	Inserts Date field.
GOLD-T	Inserts Time field.
GOLD-Q	Reverses error mode.
GOLD-S	Specifies a double-size line.
GOLD-W	Specifies a double-wide line.
GOLD-n	Repeats n times.

The remainder of this chapter describes how you use these keys to take advantage of the features available in the Layout phase.

5.5 Using Function Keys to Move the Cursor

Use the keypad and other function keys to move the cursor on the form. You can always determine the exact cursor location by checking the Line and Column blocks on the cursor status line.

Right arrow key	Pressing the right arrow key once moves the cursor one character to the right. If you are at the end of a line, pressing the right arrow key moves the cursor to the first position of the next line.
Left arrow key	Pressing the left arrow key once moves the cursor one character to the left. If you are at the beginning of a line, pressing the left arrow key moves the cursor to the end of the previous line.

Up arrow key	Pressing the up arrow key once moves the cursor up one line, in the same column position. You cannot move the cursor above Line 1.
Down arrow key	Pressing the down arrow key once moves the cursor down one line, in the same column position. You cannot move the cursor below Line 23.
LINE FEED key (F13)	Pressing the LINE FEED key moves the cursor down one line, in the same column position. The LINE FEED key is identical to the down arrow key.
BACK SPACE key (F12)	Pressing the BACK SPACE key moves the cursor one character to the left.
END-OF-LINE key (KP4)	Pressing the END-OF-LINE key moves the cursor to the rightmost column of the current line. To use the END-OF-LINE key, press KP4. If the cursor is already at the rightmost position of the current line, pressing END-OF-LINE moves the cursor to the rightmost column of the next line.
END-OF-TEXT key (KP2)	Pressing the END-OF-TEXT key moves the cursor to the last nonblank character of the current line. (A nonblank character is any picture character, any field constant or any background text other than a space.) To use the END-OF-TEXT key, press KP2. If the cursor is already positioned at (or to the right of) the last nonblank character, the cursor moves to the last nonblank character of the next line.
BEGINNING-OF-LINE key (KP0)	Pressing the BEGINNING-OF-LINE key moves the cursor to the first column of the current line. To use the BEGINNING-OF-LINE key, press KP0. If the cursor is already in Column 1, pressing BEGINNING-OF-LINE moves the cursor to the first column of the *previous* line.

BOTTOM key (GOLD-KP4)	Pressing the BOTTOM key moves the cursor to Column 1 on the bottom line of the form. To use the BOTTOM key, press the sequence GOLD-KP4.
TOP key (GOLD-KP5)	Pressing the TOP key moves the cursor to the first column of the top line of the form (Line 1, Column 1). To use the TOP key, press GOLD-KP5.

5.6 Creating Background Text and Fields

The most important parts of a form definition are fields and background text. Although not every form definition has both fields and background text (Help forms, for example, are usually entirely background text), most forms used to enter or display data include both fields and background text. When defining a form with the form editor, use two keypad function keys to distinguish between background text and fields: the TEXT key (KP8) and the FIELD key (GOLD-KP8).

5.6.1 Creating Background Text: The TEXT Key

TEXT key (KP8)	Pressing the TEXT key puts the form editor in Text mode, where any character that you type is background text. To use the TEXT key, press KP8. Text mode is the default when you enter the Layout phase to create a new form. When you press the TEXT key or when you enter the Layout phase, the first Modes block on the cursor status line reads TXT. When the form editor is in Text mode (TXT), anything that you type becomes background text and is shown whenever the form is displayed at run time.

Background text is often used to:

- Give a title or header to a form (for example, Basic Employee Information)
- Provide prompts for a field (for example, a field that collects or displays an employee's name is likely to have the background text Name: next to the field)

- Provide messages that are always displayed to the operator (For example, a message that says: Be sure to check your work)

You can type any character from the main keyboard when the form editor is in Text mode.

5.6.2 Creating Fields: The FIELD Key

FIELD key (GOLD-KP8)	Pressing the FIELD key puts the form editor in Field mode, in which any characters that you type are picture characters, blank spaces, or field constants.To use the FIELD key, press the sequence GOLD-KP8. When you use the FIELD key, the first Modes block on the cursor status line reads FLD indicating that you are in Field mode.

5.6.3 Identifying Field Picture Type and Length

When the editor is in Field mode (FLD), you can type only picture characters, field constants, or spaces. A picture character indicates the picture type of the data that is allowed for input at run time. A field constant is a character or embedded space in a field that is displayed at run time. For example, to include a hyphen in a field representing a telephone number, you would use the picture character 9 and the hyphen field constant (-) to create the seven-character numeric field 999-9999.

Table 5-2 lists the valid picture characters and the picture type that each represents, and Table 5-3 lists the valid field constants.

Table 5-2: Picture Characters

Picture Character	Picture Type
9	Numeric (0-9)
A	Alphabetic (A-Z, a-z, space, and any alphabetic character in the DEC Multinational Character Set)
C	Alphanumeric (A-Z, a-z, 0-9, space, and any alphanumeric character in the DEC Multinational Character Set)
N	Signed numeric (0-9, +, -, period)
X	Any displayable character in the DEC Multinational Character Set

Characters from the DEC Multinational Character Set are valid for background text.

Table 5-3: Field Constants

!	"	#	$	&	'	(	)
*	+	,	.	-	/	:	%
;	<	>	=	?	@	[	]
\	^	_	`	{	}	|	~

B (marks an embedded space)

If you want to have a space embedded within a single field, use the field constant B. At run time, one space appears in the field for each B used in the Layout phase field definition. For example, a field indicated by the following picture characters and field constant results in a single, six-character field:

```
XXXBXXX
```

At run time, TDMS automatically inserts a space in the fourth position of the form field when the operator enters data or when data is output to the field by a request.

A field can be no longer than one screen line (80 or 132 characters depending on the current form width).

5.6.4 Determining Form Field Data Types

The form field data type is determined by the picture string, which is the group of picture characters that make up the field. Table 5-4 shows the form field data types and the corresponding picture characters for each data type.

Table 5-4: Form Field Data Types

Picture Characters	Field Data Type
All 9s	UNSIGNED NUMERIC
All A's	TEXT
All Cs	TEXT
All Ns	SIGNED NUMERIC
All Xs	TEXT

(continued on next page)

Table 5-4: Form Field Data Types (Cont.)

Picture Characters	Field Data Type
Any combination of identifiers	TEXT
All 9s or all Ns with size field validator	Named size field validator (UNSIGNED BYTE, SIGNED WORD, and so on)
Date (inserted using DATE key)	DATE
Time (inserted using TIME key)	TIME

For example, a form field defined as AAAAA is a five-character TEXT field, and a field defined as NNN is a three-character SIGNED NUMERIC field. You can also use field validators to define numeric fields to have data types of SIGNED or UNSIGNED BYTE, WORD, or LONGWORD or SIGNED QUADWORD. You learn how to use and assign field validators in Chapter 6.

You can use combinations of picture characters to identify a field. For example, if your invoice numbers are always two letters followed by five numbers, you would define a field as:

```
AA99999
```

At run time, if that field is open for operator input, the operator can enter only alphabetic characters in the first two positions and only numeric characters in the last five positions. (You can impose additional validation of operator input for each field using field validators, as described in Chapter 6.)

Any field that includes at least one Text picture character has a TEXT data type. The following field pictures show fields with TEXT data types:

```
AAAAAA
(999)CC9-9999
XX-9999
A.X.B.9
```

In a TDMS application, the data type of a form field must be compatible with the data type of any record field to which the form field is mapped in a request. If the form and record fields are incompatible, TDMS request library files cannot be built.

5.6.5 Creating Special Fields: DATE and TIME Keys

DATE-FIELD key (GOLD-D)

The form editor provides a special function key that automatically inserts a Date field in the form definition. To use the DATE-FIELD key, press the sequence GOLD-D. When you press DATE-FIELD, the cursor status line clears and five date formats are displayed on the bottom line of the screen:

```
1 Month Day, Year (AAAAAAAAAB99,B9999)
2 Day-Month-Year (99-AAA-99)
3 Month/Day/Year (99/99/99)
4 Day-Month-Year (99-99-99)
5 Day-Month-Year (99-AAA-9999)
```

To select a date format, type the appropriate number for the format and press RETURN. The form editor inserts a Date field in the format that you chose on the form. If there is insufficient space on the line for the Date field, the form editor signals an error. Once the Date field is inserted, you can delete only the entire field and not a portion of it.

When you create a field using the DATE-FIELD key, the field has a DATE data type. When a Date field is mapped for input, the current date is displayed in the Date field unless specifically overridden by a default value on the form or an output mapping in a request.

TIME-FIELD key (GOLD-T)

The form editor provides a special function key that automatically inserts a Time field on the form definition. To use the TIME-FIELD key, press the sequence GOLD-T.

When you press TIME-FIELD, the cursor status line clears and two time formats are displayed on the bottom line of the screen. Choose one of the two formats by typing the appropriate number. The two time field formats are:

```
1 hour:minute:second (99:99:99)
2 hour:minute:AM/PM (99:99BAA)
```

After you select a format, a Time field is inserted on the form. If there is insufficient space on the line for the Time field, the form editor signals an error. Once the Time field is inserted, you can delete only the entire field and not a portion of it.

When you create a field using the TIME-FIELD key, the field has a TIME data type.

Mapping combined DATE/TIME FIELDS

In a request, you can map between two FIELDS form fields with data types of TIME and DATE and a single record field with data type DATE (ADT). For example:

```
INPUT FORM-TIME-FLD TO RECORD-DATE-FLD;
INPUT FORM-DATE-FLD TO RECORD-DATE-FLD;
```

TDMS moves the data from two separate fields and merges it into the single record field with data type DATE.

You can also map the single record field with DATE data type to two separate form fields TIME and DATE. For example:

```
OUTPUT RECORD-DATE-FLD TO
(FORM-DATE-FLD, FORM-TIME-FLD);
```

At run time TDMS moves the data from a single record field and displays it in the appropriate Date and Time form fields.

5.6.6 Creating Special Fields: The ADJACENT FIELD Key

ADJACENT FIELD key (GOLD-KP3)

The ADJACENT FIELD key lets you define separate fields when you have contiguous picture characters. (By default, a string of contiguous picture characters represents a single field.) To use the ADJACENT FIELD key, press the sequence GOLD-KP3.

For example, say that you want a ten-digit telephone number to be represented on the form as 13 consecutive characters (ten digits, a set of parentheses, and a hyphen), but you want two separate fields. The first field represents the first three digits of the phone number (the area code) and the second field represents the final seven digits. Type the background text, picture characters, and field constants as follows:

```
Telephone: (999)999-9999
```

Telephone: is background text, the 9s are picture characters representing a ten-digit telephone number, and the parentheses and hyphen are field constants that are displayed on the form without interrupting the continuity of the field. So far, this creates a single, ten-character field, with a numeric picture type. (The field constants are not counted when determining the length of the field.)

To divide the one field into two, move the cursor (with the right and left arrows) to the first character of the desired *second* field. In this case, you would move the cursor to the first 9 after the right parenthesis. Press the ADJACENT FIELD key and you now have two fields, one represented by the first three picture characters (999) and the second represented by the last seven picture characters (999-9999).

When the new field is created, the Field-name block on the cursor status line displays the default field name of the newly created field. When you have adjacent fields, you can see where one field ends and a second begins by moving the cursor (with the left arrow and right arrow keys) and checking the Field-name block. The second field begins at the position where the Field-name block changes.

5.7 Editing Text

Form editor function keys let you edit background text and fields directly in your form definition. The following sections describe these keys.

Note that when you delete a field and then undelete it, field attributes assigned to the field, except field validators, are carried over.

5.7.1 Deleting and Undeleting Text

DELETE-LINE key (PF4)	Pressing the DELETE-LINE key deletes the entire line on which the cursor is positioned. To use the DELETE-LINE key, press PF4 (on the keypad). If you press DELETE-LINE while the form editor is in Overstrike mode, no other lines on the form are affected. If you press DELETE-LINE while the form editor is in Insert mode, each line below the deleted line moves up one line. For example, if the form editor is in Insert mode and the cursor is on line 8. If you press DELETE-LINE, all of line 8 is deleted and stored in the line buffer, and any text or fields on lines 9 to 23 move to lines 8 to 22, respectively. You cannot use the DELETE-LINE key to delete a line in a scrolled region or a vertically-indexed field. To delete a line in a scrolled region, use the UNSCROLL key. To delete a line on which there is a vertically-indexed field, you must first deassign the vertical indexing, and then you can delete the line.
DELETE-TO-BEGINNING-OF-LINE (DEL/BOL) Key (CTRL/U)	Pressing the DELETE-TO-BEGINNING-OF-LINE (DEL/BOL) key deletes all characters from the beginning of the current line to the character to the immediate left of the cursor. To use the DEL/BOL key, press CTRL/U. For example, if the cursor is on line 9, column 41 and you press DEL/BOL, the characters in columns 1 to 41 are deleted; any characters between column 42 and the end of the line are unaffected.

DELETE-TO-END-OF-LINE (GOLD-KP2)	Pressing the DELETE-TO-END-OF-LINE (DEL/EOL) key deletes all characters from the cursor position to the end of the current line. To use the DEL/EOL key, press the sequence GOLD-KP2. For example, if the cursor is on line 9, column 41 and you press DEL/EOL, the characters in columns 41 to 80 (or, on a 132-column form, the characters in columns 41 to 132) are deleted.
UNDELETE-LINE key (GOLD-PF4)	The UNDELETE-LINE key places the contents of the last line that you deleted on the screen. If the form editor is in Overstrike mode, the cursor must be on a blank line for the line to appear. If the form editor is in Insert mode, the cursor can be on any line, but existing lines are moved down one line. To use the UNDELETE-LINE key, press the sequence GOLD-PF4.
DELETE-CHARACTER key (KP comma)	Pressing the DELETE-CHARACTER key comma) deletes the character on which the cursor is positioned. To use the DELETE-CHARACTER key, press the comma key on the keypad (KP comma). If the form editor is in Overstrike mode, pressing DELETE-CHARACTER deletes one character but does not move the cursor. If the form editor is in Insert mode, pressing DELETE-CHARACTER deletes one character and moves the remaining characters on the line one position to the left. The cursor remains in its current position.

DELETE key	The DELETE key on the main keyboard deletes the character to the left of the cursor. If the form editor is in Overstrike mode, pressing the DELETE key moves the cursor one character to the left, replacing that character with a space. All other characters on the line are unchanged. If the form editor is in Insert mode, the DELETE key moves the cursor and all characters to the right of the cursor one character to the left, and inserts a blank space at the end of the line.
UNDELETE-CHARACTER key (GOLD-KP comma)	The UNDELETE-CHARACTER key places the last single character that you deleted (with the DELETE-CHARACTER or DELETE key) in the current cursor position. To use the UNDELETE-CHARACTER key, press the sequence (GOLD-KP comma).

5.7.2 Moving Text: The Cut and Paste Feature

The cut and paste function allows you to move an area of the form to one or more other areas of the form. The following rules apply to cutting and pasting:

- You can cut and paste a portion of a line (except in a scrolled region), an entire line, or a rectangular area with the following exceptions:
  - A portion of a field
  - A portion of a vertically indexed field
  - A complete horizontally indexed field
- The area onto which you paste can contain neither text nor picture characters.
- The pasted area cannot extend beyond any boundary of the form.

To use the cut and paste feature of the form editor, follow this procedure:

1. Identify the area that you want to move by creating a select range using the SELECT key (KP period) and cursor movement keys.
2. Remove the area with the CUT (KP6) key.
3. Replace the section that you removed in one or more locations with the PASTE (GOLD-KP6) key.

SELECT key (KP period)	The SELECT key identifies the area of the form that you wish to move. (After you select and cut the area, you may replace it in its original position if you wish.) To use the SELECT key, press the period (.) key on the keypad (KP period). (On the VT200 keyboard, press the SELECT key or KP period.) Position the cursor at the beginning of the line, line segment, or rectangular area that you want to move. When you press the SELECT key, the character where the cursor is positioned is shown in reverse video. When you use the SELECT key, you will find it very helpful to use an underscore cursor rather than a block cursor. The select function shows the currently selected area in reverse video, and it is often difficult to see exactly where the select area ends when you use a block, rather than underline cursor. Now, move the cursor to the end of the line or line segment, or the opposite corner of the rectangle, depending on the type of area that you intend to be moved. While in Select mode, you can use any function key that moves the cursor. As the cursor moves, the current selected range is shown in reverse video, and all other video characteristics on the form are temporarily turned off.
CUT key (KP6 or REMOVE)	Use the CUT key to move the contents of the selected range to the paste buffer. To use the CUT key, press KP6. (On the VT200 keyboard, press the REMOVE key or KP6.) With the select range shown in reverse video, press CUT. The area that you want to move is deleted from the form and stored in the paste buffer. When you press CUT, the select function stops, and TDMS restores the video screen attributes.
TEST-PASTE key (KP3)	Press the TEST-PASTE key to determine where you can paste an area on a form. To use the TEST-PASTE key, press KP3.

After you have selected and cut the area that you want to duplicate elsewhere on the form, move the cursor to the upper-left corner of an area on the form. Press TEST-PASTE and you see an outline of the paste buffer in reverse video. TDMS removes all other video attributes. If the paste buffer can validly be placed in the space that you test, you see an entire outline of the paste buffer and receive no error. If the contents of the paste buffer do not meet the form editor paste requirements, you get an error message and see only a portion of the outline. As soon as you type any character, the form editor removes the outline of the paste buffer and restores all other video attributes. Using the TEST-PASTE key does not affect the contents of the paste buffer.

PASTE key (GOLD-KP6 or INSERT)

The PASTE key duplicates the contents of the paste buffer on the form. To use the PASTE key, press the sequence GOLD-KP6. (On the VT200 keyboard, press the INSERT key or GOLD-KP6.)

When you use the PASTE key, all of the conditions for pasting in the form editor must be met. (The area that is used for pasting can contain neither text nor picture characters. The pasted area can extend neither beyond the bottom of the form nor beyond the right boundary of the form.) If you press PASTE and the operation is not valid, the form editor signals an error and does not paste anything on the form. The contents of the paste buffer are unchanged.

You can paste onto an area that includes a double-wide or double-size line as long as the resulting line does not extend beyond the right margin of the form. However, it is generally better practice to create any double-wide or double-size lines after you complete the cutting and pasting.

When you cut and paste fields, the form editor continues to recognize the areas as fields. Any attributes assigned to the original fields are carried over to the fields created by the cut/paste feature. Field validators assigned to the form fields are not carried over.

Cutting and pasting a field places the field at the end of the access order list. The access order list, used to determine the run-time access order of input fields, is discussed in the chapter entitled Assigning Field Order.

5.7.3 Centering Text: The CENTER Key

CENTER key (KP1)

The CENTER key lets you center text on a line or in a rectangular area. To use the CENTER key, press KP1.

To center text on a line, move the cursor to any point on the line and press CENTER. The form editor centers the existing background text and fields between the left and right boundaries of the form.

To center text within a rectangular area, use the cursor-movement keys and the SELECT key to put the area in a select range. With the rectangular area in the select range (shown in reverse video), press CENTER. All of the text in the area is centered on each line between the left and right boundaries of the select range. When you press CENTER, the SELECT function stops and TDMS restores video attributes.

5.7.4 Inserting a Blank Line: The OPEN LINE Key

OPEN-LINE key (GOLD-KP0)

Pressing the OPEN-LINE key inserts a blank line on the form. To use the OPEN-LINE key, press the sequence GOLD-KP0. You can use the OPEN-LINE key only if line 23 of the form is blank.

You cannot use the OPEN-LINE key if the cursor is on the bottom line of the form (line 23) or if there are any characters on line 23.

Pressing OPEN-LINE causes the current line to be blank and moves each line below the current line down one line. For example, if the cursor is anywhere on line 7 and you press OPEN-LINE, line 7 becomes blank and all text and fields that were on lines 7 to 22 are moved to lines 8 to 23, respectively.

5.7.5 Changing the Case of Existing Text

CHANGE-CASE key (GOLD-KP1)

Press the CHANGE-CASE key to change uppercase background text to lowercase and to change lowercase background text to uppercase. To use the CHANGE-CASE key, press the sequence GOLD-KP1.

You can use the CHANGE-CASE key with the REPEAT key (GOLD-n) to change the case of a series of letters. For example, to change the case of the background text "address," move the cursor to the first letter, press the sequence GOLD-7 (use the 7 on the main keyboard), then press the sequence GOLD-KP1. The form editor changes the text to "ADDRESS."

You can also select text with the SELECT key and change the case of the text within the selected area.

The CHANGE-CASE key does not affect any characters other than alphabetic background text.

5.8 Creating Video Features

The form editor provides a variety of video attributes that you can include on a form. Using video attributes can enhance form design and improve operator productivity.

The video features that the form editor provides in the Layout phase include the ability to:

- Emphasize lines with double-wide or double-size characters
- Highlight any parts of the form with any combination of bolding, blinking, underlining, and reverse video
- Draw solid boxes or lines on the form

In addition, a request can specify that a field is displayed with a particular video attribute. See the *VAX TDMS Request and Programming Manual* to learn about this feature.

This section describes how to use the video features that the Layout phase of the form editor provides.

5.8.1 Double-Wide and Double-Size Lines

DOUBLE-WIDE key (GOLD-W)	Pressing the GOLD-W key sequence either causes a normal-width line to change to a double-wide line or causes a double-wide line to change to normal width line. When you press the GOLD-W sequence, any characters on the current line double in width. All characters that you type on the line after pressing GOLD-W also become double-wide. At run time, all background text and fields are shown double-wide. You can assign video attributes to double-wide lines. You can also include double-wide lines in scrolled regions. To do this, first define the lines of the scrolled region as double-wide. Then create the fields and define the region as scrolled.

A double-wide line displays characters twice as wide as normal VT100 or VT200 characters. There are 40 columns in a double-wide line for an 80-column screen and 66 columns in a double-wide line for a 132-column screen. In a double-wide line, all characters and spaces are enlarged. You cannot create a double-wide line when any characters are to the right of column 40 (for an 80-column form) or column 66 (for a 132-column form).

To change a double-wide line to normal width, press GOLD-W.

Double-wide lines on a 132-column form provide excellent readability. Characters displayed in this format are larger than characters displayed in 80-column mode.

DOUBLE-SIZE key (GOLD-S)

Pressing GOLD-S either causes a normal line to change to a double-size line or causes a double-size line to change to normal size.

When you press GOLD-S, the characters on the current line are redisplayed in double size, and each line below the current line moves down one line. (For example, if the cursor is at line 8 and you press GOLD-S, text and fields on lines 9 to 22 move to lines 10 to 23.) You cannot use GOLD-S if there are any characters on line 23 of the form.

A double-size line displays characters that are twice as wide and twice as high as normal VT100 or VT200 characters. All double-size lines use two lines of a form; they use 40 columns on an 80-column screen or 66 columns on a 132-column screen.

Any field on a double-size line must be a Display Only field and therefore it cannot be used as an input field. If you create a field on a double-size line (in the Layout phase), be sure to assign the Display Only attribute in the Assign phase. FDU defaults the Display Only attribute to fields in double-size lines.

To change the current line from a double-size line to a normal line, press GOLD-S. The double-size line returns to normal size, and each line below the current line moves up one line. (For example, if the cursor is on the double-size line 9 and you press GOLD-S, the current line becomes normal-size line 8, and any text and fields on lines 10 to 23 move to lines 9 to 22.)

5.8.2 Video Highlighting

The form editor allows you to display any portion of the form with bolding, blinking, underlining, reverse video, or any combination of these features. In the Layout phase, you assign video highlights that display whenever the run time form displays. In the Form phase, you can also assign video highlights that display only when a field is open for input (see Chapter 4).

If you assign video attributes to a field in the Layout phase, these video attributes are added to the default video attributes assigned in the Form phase.

If you assign video attributes to an area (rather than a field) in the Layout phase and then create a field in that area, the video attributes are not added to the default video attributes assigned in the Form phase. Rather, the default attributes assigned in the Form phase apply to the field.

The following restrictions govern the use of video highlighting:

- The same video attributes must apply to the entire field. (That is, you cannot have a field that is part blinking and part reverse video.)
- In a scrolled region, all columns in the scrolled region must have the same video attributes.

You may apply video attributes to all or part of any double-wide or double-size line except for the conditions previously listed.

To apply video attributes to a character, line segment, or rectangular area, follow this procedure:

1. (Optional). Make sure that you are using an underline cursor rather than a block cursor. Using the block cursor can make it very difficult to determine exactly where a video attribute begins or ends.

2. Move the cursor to the beginning or end of the area to which you want to assign or change video highlights. To assign highlights:

   - To a single space, move the cursor to that space
   - To a line segment, move the cursor to the beginning of that line segment
   - To a rectangular area, move the cursor to one corner of the rectangle

3. Press the SELECT key (KP period or SELECT). The cursor position is shown in reverse video, and all other video attributes on the form are temporarily removed.

4. Move the cursor to the other end of the area of video highlighting with the arrow keys and other form editor function keys (END-OF-LINE, LINE FEED, and so on). For a line, move to the other end of the line or line segment; for a rectangle, move to the opposite corner of the rectangle. As you move the cursor, the form editor shows the select range in reverse video. (Skip this step if you want to assign video attributes to only one character.)

   Note

   At this point, the select range is shown in reverse video. Any video attribute you assign applies to the entire area that is shown in reverse video. If you are satisfied that the area shown is the area to which you want to assign video attributes, go to the next step.

   To expand or reduce the area, move the cursor. To cancel the select range and start over, use the RESET key (GOLD-KP period). When you press RESET, the select range is canceled and the previous video attributes are restored.

5. Press the VIDEO key (KP7) when the correct area is in the select range. The cursor status line clears and the VIDEO prompt appears in the lower left corner of the screen. At the VIDEO prompt, you can type one or more attributes, as in the following list. You can type either the full name of the attribute or the abbreviation as shown with underlined letters:

   BOLD – Activates the Bold attribute

   BLINK – Activates the Blinking attribute

   REVERSE – Activates the Reverse Video attribute

UNDERLINE – Activates the Underline attribute

CLEAR – Clears all video attributes

SAVE – Saves current (or named) attributes

RESTORE – Ends the video attribute assignment and restores any video attributes that had previously been assigned to the select range

To assign more than one video attribute, separate the attribute names with a space, a comma, or a RETURN.

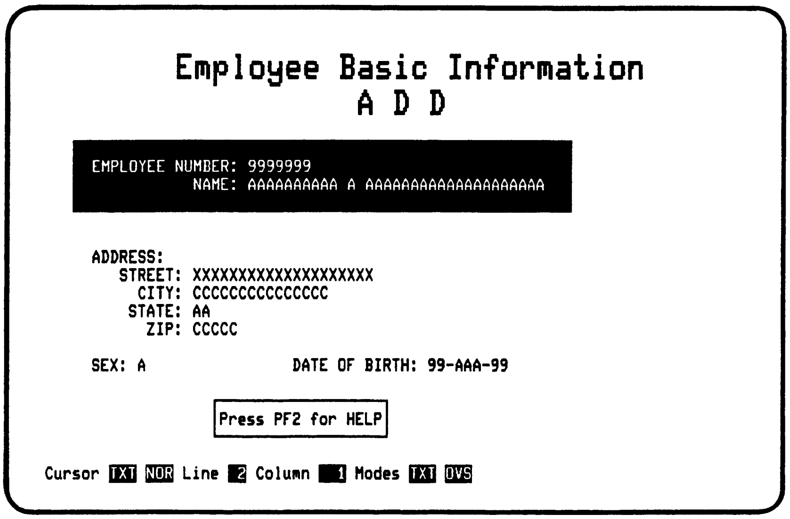
When you are finished assigning video attributes, type SAVE or S or press RETURN at the VIDEO prompt. The form with its new video attributes displays, and all video attributes that had previously been assigned outside the select range are restored.

5.8.3 Drawing Solid Lines and Rectangles

DRAW key (KP hyphen)	Use the DRAW key to create solid boxes or lines on the form. To use the DRAW key, press KP hyphen. The DRAW key creates a solid rectangle or solid line around the the current select range. To draw a box, use the SELECT key and cursor-movement keys to create a select range where you want the box drawn. Press DRAW and the form editor draws the box.

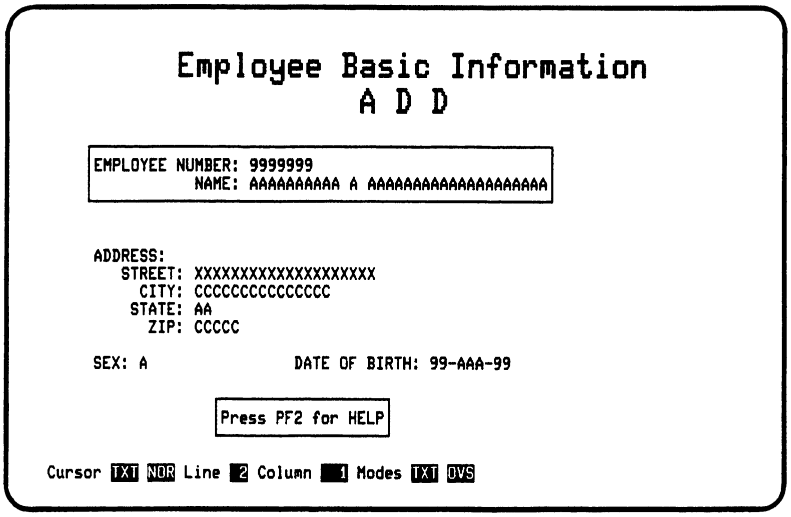
The following steps demonstrate how to use the DRAW key:

1. Create a select range, including blank rows and columns on either side.



```
              Employee Basic Information
                        A D D

   EMPLOYEE NUMBER: 9999999
              NAME: AAAAAAAAAA A AAAAAAAAAAAAAAAAAAAA

   ADDRESS:
      STREET: XXXXXXXXXXXXXXXXXXXX
        CITY: CCCCCCCCCCCCCCC
       STATE: AA
         ZIP: CCCCC

   SEX: A                DATE OF BIRTH: 99-AAA-99

                 Press PF2 for HELP

Cursor TXT NOR Line 2 Column 1 Modes TXT OVS
```

2. Press the DRAW key, and the form editor draws a box around the area.



```
              Employee Basic Information
                        A D D

   EMPLOYEE NUMBER: 9999999
              NAME: AAAAAAAAAA A AAAAAAAAAAAAAAAAAAAA

   ADDRESS:
      STREET: XXXXXXXXXXXXXXXXXXXX
        CITY: CCCCCCCCCCCCCCC
       STATE: AA
         ZIP: CCCCC

   SEX: A                DATE OF BIRTH: 99-AAA-99

                 Press PF2 for HELP

Cursor TXT NOR Line 2 Column 1 Modes TXT OVS
```

To draw a solid line, use the SELECT key to create a linear select range (either vertical or horizontal), and then press DRAW. All lines in the select range must have the same width and height characteristics.

You cannot use the DRAW key if there are any characters (background text, picture characters, or field constants) in the top or bottom lines of the rightmost or leftmost column of the select range. If there are any characters in these locations, the form editor signals an error. However, you can use the DRAW key to create intersecting lines and boxes.

UNDRAW key (GOLD-KP hyphen)	Use the UNDRAW key to remove boxes and lines (or portions of boxes and lines) that were created with the DRAW key. To use the UNDRAW key, press the sequence GOLD-KP hyphen.
	To remove a box or a line, use the SELECT key to create a select range that includes the box or line that you want to remove. Press UNDRAW, and all boxes and lines, or portions of boxes and lines, on the perimeter of the select range are deleted.
	You can also remove lines and boxes by using any of the delete function keys provided with the form editor.

5.9 Creating Scrolled Regions

A scrolled region is a section with one or more identical lines on a form that enables the operator to enter and/or read many lines of data on the form.

In TDMS applications, scrolled regions are defined on the form. The request identifies the form (for display) and its fields (for input/output) and maps the scrolled region on the form to one or more record arrays. The application program opens data files and executes the request.

5.9.1 Rules for Scrolled Regions

The following rules govern the use of scrolled regions on a form:

1. Only entire lines can be scrolled. All fields and background text on a line become part of the scrolled line.
2. All lines in a scrolled region must be identical. That is, each line must have the same fields (with the same picture type), the same background text, and the same video attributes.
3. Scrolled regions can be from 1 to 23 lines long. The last line on the form is reserved by TDMS.

4. Scrolled regions can include double-wide lines but they cannot include double-size lines. If one line in a scrolled region is a double-wide line, all lines in the scrolled region must be double-wide. To create a double-wide line scrolled region, first create the double-wide lines, and then create the scrolled region.

5. Scrolled regions can be contiguous on the form. For example, you can define one scrolled region on lines 2 to 7 and a second scrolled region on lines 8 to 10.

6. If you create a scrolled region in which there will be no operator input, you must follow certain procedures in the form definition. If your scrolled region does not include any fields from which operator information will be collected, be sure to read the section Special Provisions for a Display-Only Scrolled Region.

5.9.2 Defining a Scrolled Region on the Form

You create scrolled regions in the form definition during the Layout phase, using the form editor keypad function keys. This section describes the function keys that you use; the following section is a brief example of a scrolled region in a form definition.

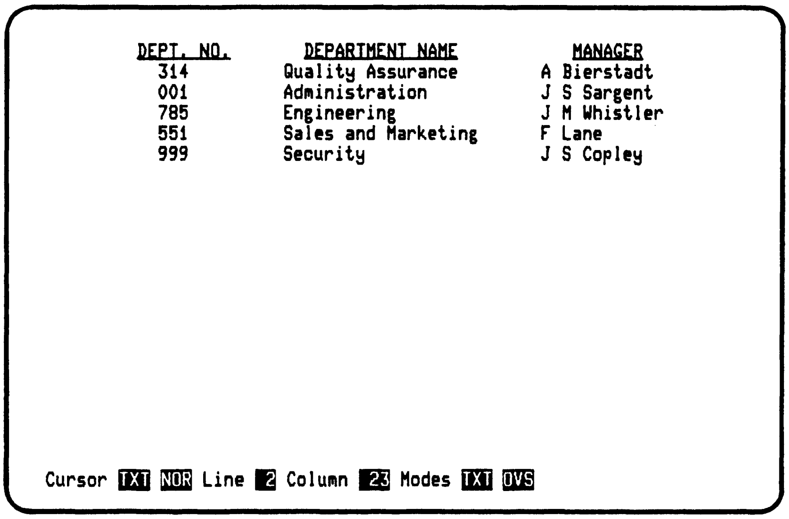
SCROLL key (KP9)	The SCROLL key identifies a line that is to be part of a scrolled region. To use the SCROLL key, press KP9. When you press SCROLL, the current line is shown in reverse video. To add or create lines in the scrolled region *below* the current line, use the down arrow or the LINE FEED key. Each time you press either of these keys, the scrolled line is duplicated below the current line and shown in reverse video. To add or create lines in the scrolled region *above* the current line, use the up arrow or the BEGINNING-OF-LINE key. Each time you press either of these keys, the scrolled line is duplicated above the current line and shown in reverse video.

	Adding or creating lines in this way occurs only when you are first defining the scrolled region. To add or create lines in an existing scrolled region, see the section entitled Adding Lines to a Scrolled Region.
ENDSCROLL key (KP5)	The ENDSCROLL key ends the definition of a scrolled region. To use the ENDSCROLL key, press KP5. After defining the scrolled region using the SCROLL key and the up or down arrows, press the ENDSCROLL key to end the scrolled region. When you press ENDSCROLL, the scrolled region is no longer shown in reverse video. After you press ENDSCROLL, you can determine the location and length of any scrolled region by looking at the second cursor block of the cursor status line. When this block reads SCR, the line is scrolled; when the block reads NOR, the line is not scrolled. No line can be only partially scrolled.
UNSCROLL key (GOLD-KP9)	The UNSCROLL key deletes a scrolled line. To use the UNSCROLL key, press the sequence GOLD-KP9. When you press UNSCROLL, the form editor deletes the *entire* line on which the cursor is positioned as long as that line is part of a scrolled region. If you press UNSCROLL with the cursor on a one-line scrolled region, that line is changed from a scrolled line (SCR on the cursor status line) to a normal line (NOR).

5.9.3 Example of a Scrolled Region

This section gives a brief example of how to include a scrolled region in a form definition. In a Corporate Personnel application, one of the forms collects and displays information about specific departments in the company. Because there are more than 50 departments, it is useful to collect and display this information in a scrolled region.

After an operator enters data onto the form, it looks like this:



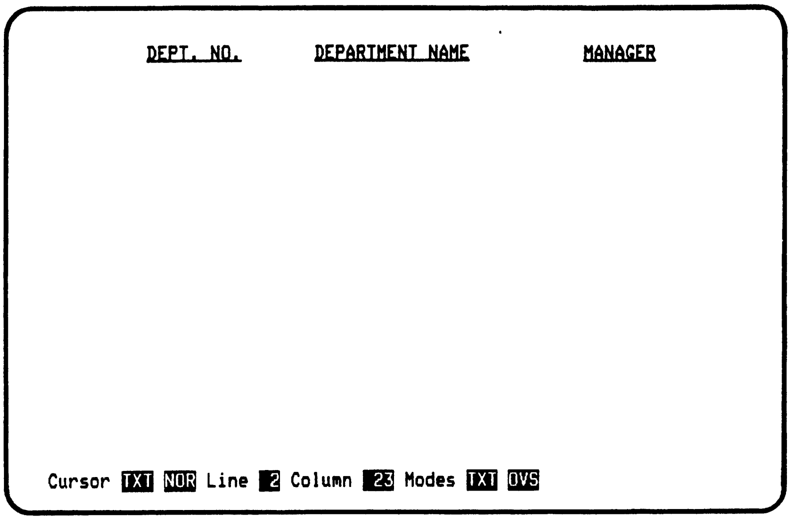
```
            DEPT. NO.       DEPARTMENT NAME          MANAGER
               314          Quality Assurance        A Bierstadt
               001          Administration           J S Sargent
               785          Engineering              J M Whistler
               551          Sales and Marketing      F Lane
               999          Security                 J S Copley
```

```
  Cursor TXT NOR Line 2 Column 23 Modes TXT OVS
```

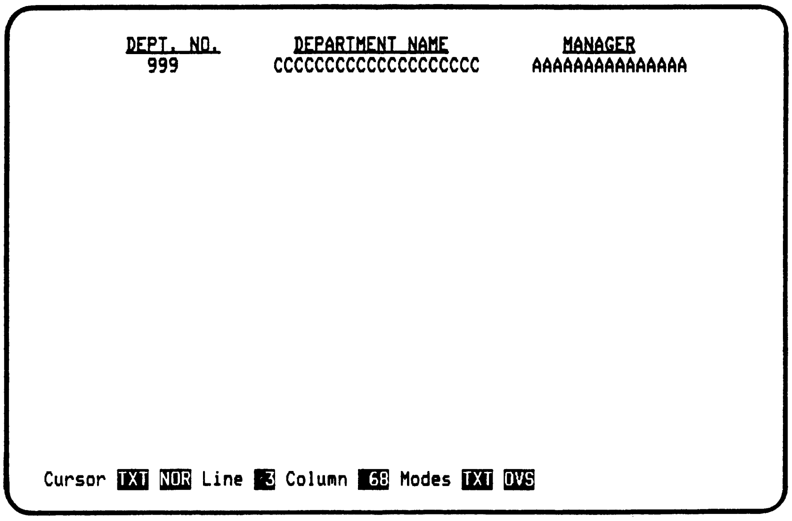
The first line (DEPT. NO., DEPARTMENT NAME, MANAGER) is background text that is not part of the scrolled region; the other five lines are the scrolled region and contain data. In addition to the five lines of data shown on the form, the operator can enter or read as many lines of information as the TDMS request allows using TAB, BACK SPACE, up arrow, and down arrow keys.

To create a scrolled region such as this on the form, make sure that you are in the Layout phase and follow this procedure:

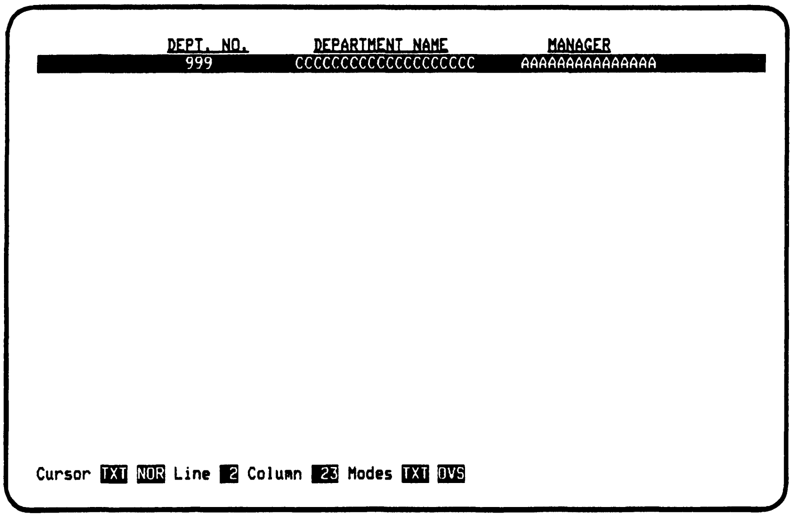
1. Type the background text DEPT. NO., DEPARTMENT NAME, and MANAGER on a single line of the form.


```
           DEPT. NO.      DEPARTMENT NAME          MANAGER











  Cursor TXT NOR Line 2 Column 23 Modes TXT OVS
```

2. Move to the next line on the form and change from Text mode (TXT) to Field mode (FLD). Type the picture characters representing the department number (three 9s, representing a 3-character numeric field), department name (20 Cs, representing a 20-character alphanumeric field), and manager (15 A's, representing a 15-character alphabetic field) under the corresponding background text.

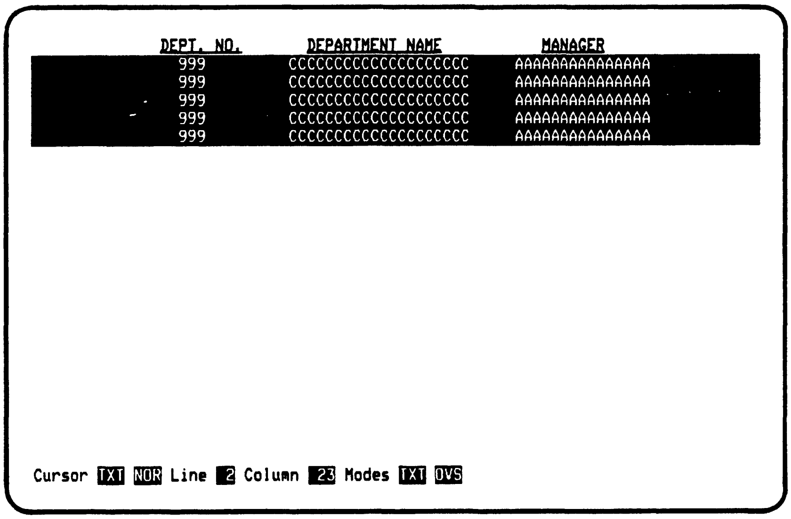

```
         DEPT. NO.      DEPARTMENT NAME          MANAGER
           999       CCCCCCCCCCCCCCCCCCCC    AAAAAAAAAAAAAAA










  Cursor TXT NOR Line 3 Column 68 Modes TXT OVS
```

3. Press SCROLL (KP9) when you have finished typing the three fields on the line and with the cursor still on the line with the three fields. The line is shown in reverse video.



```
                DEPT. NO.        DEPARTMENT NAME          MANAGER
                   999          CCCCCCCCCCCCCCCCCCCC     AAAAAAAAAAAAAAA

Cursor TXT NOR Line 2 Column 23 Modes TXT OVS
```

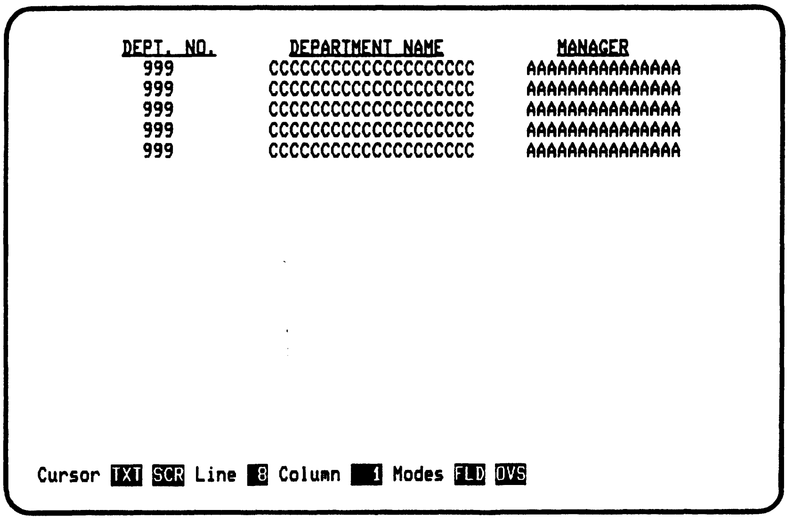
4. Press the down arrow (or LINE FEED) four times to create a five-line scrolled region. You will see five identical lines in reverse video.



```
                DEPT. NO.        DEPARTMENT NAME          MANAGER
                   999          CCCCCCCCCCCCCCCCCCCC     AAAAAAAAAAAAAAA
                   999          CCCCCCCCCCCCCCCCCCCC     AAAAAAAAAAAAAAA
                   999          CCCCCCCCCCCCCCCCCCCC     AAAAAAAAAAAAAAA
                   999          CCCCCCCCCCCCCCCCCCCC     AAAAAAAAAAAAAAA
                   999          CCCCCCCCCCCCCCCCCCCC     AAAAAAAAAAAAAAA

Cursor TXT NOR Line 2 Column 23 Modes TXT OVS
```

5. Press the ENDSCROLL key (KP5) to end the definition of the scrolled region. The lines that had been shown in reverse video return to normal video and the definition of the scrolled region is complete.

The form definition, including the cursor status line, looks like this:



```
          DEPT. NO.       DEPARTMENT NAME         MANAGER
            999        CCCCCCCCCCCCCCCCCCCC   AAAAAAAAAAAAAAA
            999        CCCCCCCCCCCCCCCCCCCC   AAAAAAAAAAAAAAA
            999        CCCCCCCCCCCCCCCCCCCC   AAAAAAAAAAAAAAA
            999        CCCCCCCCCCCCCCCCCCCC   AAAAAAAAAAAAAAA
            999        CCCCCCCCCCCCCCCCCCCC   AAAAAAAAAAAAAAA









Cursor TXT SCR Line 8 Column 1 Modes FLD OVS
```

The cursor status line indicates that the cursor is:

- At background text (TXT)
- In a scrolled region
- At line 8, column 1
- In Field and Overstrike mode

5.9.4 Adding Lines to a Scrolled Region

To increase the number of lines in an existing scrolled region on a form, you must follow this procedure:

1. Use the UNSCROLL key (GOLD-KP9) to delete *all but one* of the lines in the scrolled region. For example, to change an existing four-line scrolled region to a six-line scrolled region, first unscroll the fourth, third, and second lines of the scrolled region.
2. Use the UNSCROLL key to change the remaining *first* line of the scrolled region from a scrolled line (SCR on the cursor status line) to a normal line (NOR).

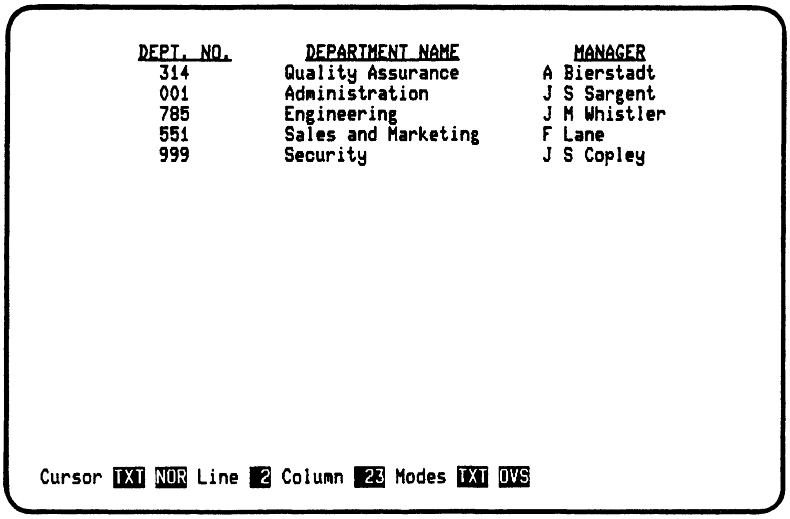
3. Press the SCROLL key (KP9) with the cursor on the remaining line to move the cursor down five lines (using either down arrow or LINE FEED), then press ENDSCROLL (KP5). You now have a six-line scrolled region, and each field in the new scrolled region has the same name, attributes, and field validators as the previous scrolled region.

5.9.5 Special Provisions for a Display-Only Scrolled Region

For any field, the cursor moves at run time to that field *only* when the field is mapped for input (and the field is not defined in the form definition as Display Only). For nonscrolled fields, this is not a problem, because the operator can read any information that is displayed in (that is, output to) the fields regardless of the cursor position. However, in a scrolled region where none of the fields are mapped for input, the operator would be able to read only those lines that appear on the screen when the form is displayed. The cursor, which is used to scroll the information up or down, cannot get to the scrolled region, and the operator cannot use the scrolling mechanism.

The solution to the problem is to include a one-character, "dummy" field in the scrolled region, which will be mapped for input by the request. The following example demonstrates the use of a dummy input field for a Display-Only scrolled region.

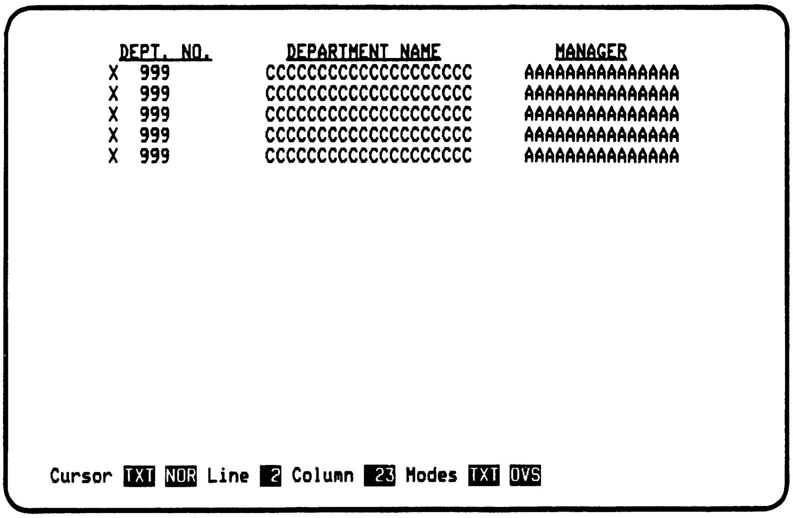
The information that you generated in the previous example of a scrolled region could be part of another form, which is used to display information about your company:



```
            DEPT. NO.     DEPARTMENT NAME          MANAGER
               314        Quality Assurance        A Bierstadt
               001        Administration           J S Sargent
               785        Engineering              J M Whistler
               551        Sales and Marketing      F Lane
               999        Security                 J S Copley





 Cursor TXT NOR Line 2 Column 23 Modes TXT OVS
```

In addition to the 5 lines displayed on the screen, there are 40 more lines of information that the operator might need to see. On this form, the operator can only read information from the form but cannot enter information.

Set up the scrolled region on your form definition to look like this:



```
  DEPT. NO.        DEPARTMENT NAME           MANAGER
 X  999        CCCCCCCCCCCCCCCCCCCC    AAAAAAAAAAAAAAA
 X  999        CCCCCCCCCCCCCCCCCCCC    AAAAAAAAAAAAAAA
 X  999        CCCCCCCCCCCCCCCCCCCC    AAAAAAAAAAAAAAA
 X  999        CCCCCCCCCCCCCCCCCCCC    AAAAAAAAAAAAAAA
 X  999        CCCCCCCCCCCCCCCCCCCC    AAAAAAAAAAAAAAA

Cursor TXT NOR Line 2 Column 23 Modes TXT OVS
```

Follow the same procedure as when you created the scrolled region in the previous section, adding a single-character field (using the X picture character) to the left of the field representing DEPT. NO.

The field represented by the single X is the "dummy" input field. The person creating the request maps the dummy field for *input* and maps the other fields for *output* (from the record to the form). At run time, the cursor moves to the field that is mapped for input (the dummy field), and the operator is able to scroll through the region and read any information available.

Remember the following when you create a scrolled region that includes no fields that are available for input:

1. Create a one-character field that will be used for input. You can place the field anywhere on the scrolled line, and you can use any picture character to designate the field. At run time, the cursor moves to this field, allowing the operator to use TAB, BACK SPACE, up arrow, and down arrow keys in the scrolled region.

2. Make sure that the record definition includes a provision for this field. Even though the field has no meaningful data, a record field must be provided for the input mapping.

3. Define the field as No Echo when you assign attributes to this field (in the Assign phase) so that the run-time form does not display any inadvertent operator input. Be sure that the field does *not* have a Response Required attribute.

4. Follow this procedure only if none of the fields in a scrolled region is available for input. If some of the fields in a scrolled region are output-only, but at least one field on each line is mapped for input (and not defined as Display Only in the form definition), you do not need to include the dummy input field.

5.10 Creating Indexed Fields in the Layout Phase

An indexed field is a single form field with two or more elements having the same name but a different numerical index for identification. You define the location of an indexed field and its individual elements in the Layout phase, and you define the field as being indexed in the Assign phase. Chapter 6 discusses indexed form fields, including rules governing their use, examples, and detailed procedures for creating them.

In the Layout phase, you should know the following about creating an indexed field:

- Each element in the indexed field must have the same length, picture type, video attributes, and field attributes.
- Indexed fields must be either entirely inside or outside of a scrolled region.
- The elements of a vertically indexed field must be aligned in the same column with no intervening blank lines.
- The elements of a horizontally indexed field must be on the same line with the same number of columns between each element.

5.11 Assigning Field Attributes from the Layout Phase

ASSIGN-FIELD-ATTRIBUTE key (GOLD-ENTER)	The ASSIGN-FIELD-ATTRIBUTE key allows you to go to the Assign phase from the Layout phase to assign attributes to a single field. To use the ASSIGN-FIELD-ATTRIBUTE key, press the sequence GOLD-ENTER when the cursor is positioned on the field to which you want to assign attributes.

When you press GOLD-ENTER, you enter the Assign phase in order to assign attributes to the field on which the cursor is positioned. (In Chapter 6, you learn the meaning and use of field attributes including field validators.) After you finish assigning attributes to the field, the form editor returns you to the Layout phase (unless you press the MENU key on the Attribute Assignment form.)

Assigning Field Attributes and Validators 6

In this chapter, you learn how to use field attributes and field validators in form definitions. You assign field attributes and validators in the Assign phase of the form editor.

6.1 Uses of Field Attributes and Validators

Field attributes and field validators provide specific characteristics to individual fields on a form. Each field has its own set of attributes and validators; you assign them during the Assign phase. If you do not use the Assign phase, TDMS provides default field attribute values for each field that you identify during the Layout phase.

Field attributes can do one or more of the following:

- Identify the field to TDMS by providing a field name
- Provide a default value
- Provide a help message
- Affect the appearance of the field and data entered into it
- Describe or place certain conditions on the data entered by the operator
- Inform TDMS that there are validators associated with the field

Field validators provide special validation procedures for data entered by the operator at run time. They supplement the input requirements that you specify with picture characters in the Layout phase. You can use field validators to identify:

- A list of valid entries
- A numeric or alphabetic range of valid entries
- A check digit, which applies a predefined algorithm to a numeric field
- A predefined size (for example, SIGNED BYTE or UNSIGNED LONGWORD) for a numeric field.

Any data entered by the operator at run time must conform to the requirements of:

- The picture string and maximum field length specified by picture characters during the Layout phase
- The field attributes specified during the Assign phase
- The field validators, if specified during the Assign phase

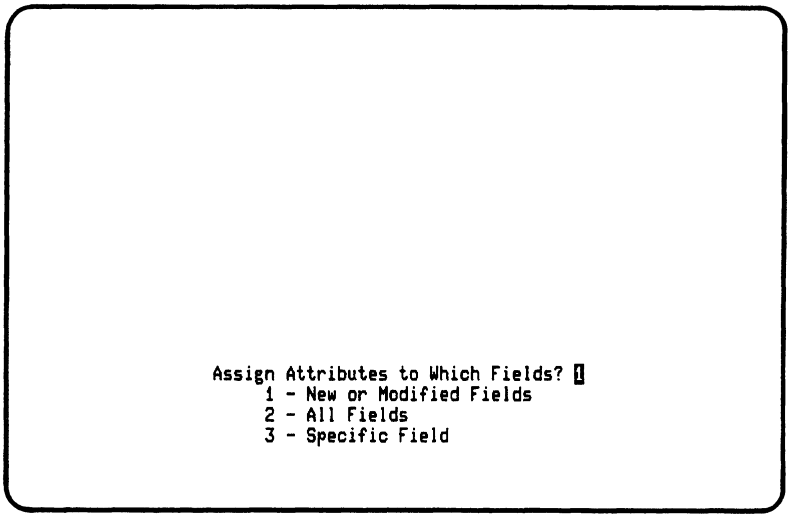
6.2 Assign Phase: Introduction

You can enter the Assign phase to assign field attributes and validators in either of two ways:

- From the Phase Selection menu, by typing ASSIGN at the Phase choice prompt. Entering the Assign phase from the Phase Selection menu gives you the choice of assigning attributes and validators to all fields, new or modified fields, or an individual field.
- From the Layout phase, using the ASSIGN-FIELD-ATTRIBUTE (GOLD-ENTER) key, with the cursor positioned in a field. This method allows you to assign attributes only to that field on which the cursor is positioned.

6.2.1 Assign Phase Menu

If you enter the Assign phase from the Phase Selection menu, the Assign Phase menu is displayed at the bottom of the Phase Selection menu form. The Assign Phase menu allows you to choose new or modified fields, all fields, or a specific field. The Assign Phase menu is shown here:



```
Assign Attributes to Which Fields? 1
     1 - New or Modified Fields
     2 - All Fields
     3 - Specific Field
```

- Type 1 or press RETURN. This is the default response to assign attributes to all new or modified fields.

  A field is considered to be new or modified if you have not assigned attributes to the field and the field was created or modified during the current form-editing session. Choose this selection when you are either:

  - Creating a new form and want to assign attributes to all of the fields that you have just created in the Layout phase
  - Modifying a form and want to assign attributes to fields that you have just added or modified in the Layout phase

  After you have assigned attributes to all new or modified fields, the form editor returns you to the Phase Selection menu.

- Type 2 to assign attributes to all fields on the form. Choose this selection when you want to assign attributes to all of the fields on the form and have previously assigned attributes (or accepted defaults) for some or all of the fields.

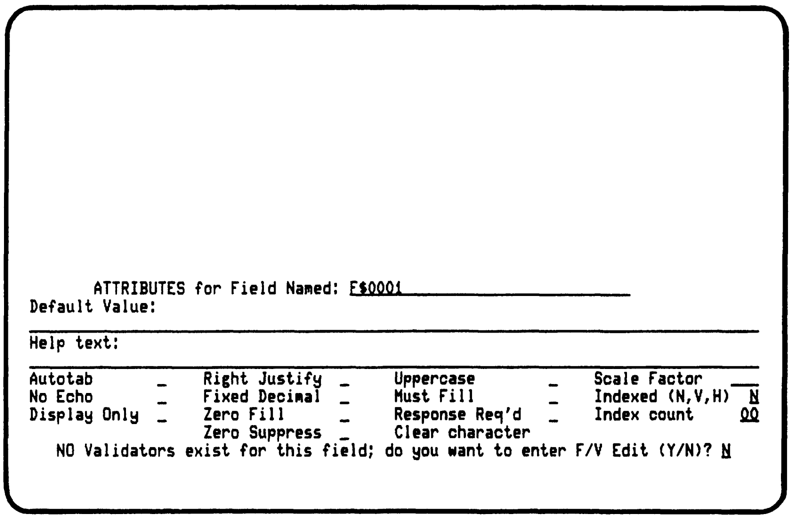
When you choose this menu selection, you can assign attributes (or accept the existing attributes) to every field on the form before the form editor returns you to the Phase Selection menu.

- Type 3 to assign attributes to a specific field on the form. When you choose this menu selection, the form editor prompts you with the message Enter field name: at the bottom of the screen. Type in the current field name and press RETURN. When you have assigned attributes to the field that you named, the form editor returns you to the Phase Selection menu.

If you want to assign attributes to a specific field but do not know the field name, you should enter the Layout phase, move the cursor to the field, and use the ASSIGN-FIELD-ATTRIBUTE key. The form editor places you in the Assign phase for that field only. When you have assigned attributes to the field, the form editor returns you to the Layout phase.

6.2.2 Attribute Assignment Form

The Attribute Assignment form, shown in Figure 6-1, is used to collect and display field attribute information for each field.



```
      ATTRIBUTES for Field Named: F$0001_______________________
Default Value:
____________________________________________________________________
Help text:
____________________________________________________________________
Autotab       _   Right Justify _   Uppercase       _   Scale Factor  ___
No Echo       _   Fixed Decimal _   Must Fill       _   Indexed (N,V,H) N
Display Only  _   Zero Fill     _   Response Req'd  _   Index count    00
                  Zero Suppress _   Clear character
   NO Validators exist for this field; do you want to enter F/V Edit (Y/N)? N
```

Figure 6-1: Attribute Assignment Form (80-Column Form)

The Attribute Assignment form for an 80-column form has a slightly different format than the one for a 132-column form. The 80-column form is used to demonstrate examples throughout this chapter.

When the Attribute Assignment form is displayed for any field, it shows the current value for each attribute. The current value is determined by the attributes that you previously specified for the field in the Assign phase. If you have not used the Assign phase for this field, the current value is determined by:

- The formwide default attributes, if you specified them in the Form phase
- The form editor defaults, if you did not specify formwide default attributes

Note

In the Form phase, you cannot specify formwide attributes for Field Name, Help Text, Default Value, Scale Factor, Indexed, Index Count, or Field Validators.

Later in this chapter, you learn the meaning of each of the field attributes listed on the Attribute Assignment form.

6.2.3 Assign Phase Function Keys

The TDMS form editor provides several function keys that you can use during the Assign phase, as shown in Table 6-1.

Table 6-1: Assign Phase Function Keys

Key	Function
TAB	Moves the cursor to the next field on the Attribute Assignment form. Press TAB to move the cursor from field to field within the Attribute Assignment form. Pressing the TAB key does not change the contents of a field.
BACK SPACE (F12)	Moves the cursor to the previous field on the Attribute Assignment form. Use BACK SPACE to move the cursor backwards from field to field within the Attribute Assignment form. Pressing the BACK SPACE key does not change the contents of a field.
LINE FEED (F13)	Deletes the contents of the field where the cursor is positioned. The LINE FEED key is most useful when you want to replace the current contents of the Field Name, Help Text, or Default Value fields.

(continued on next page)

Table 6-1: Assign Phase Function Keys (Cont.)

Key	Function
HELP (PF2 or F15)	Displays help messages for individual fields during the Assign phase.
CTRL/W and CTRL/R	Refresh the screen. Press CTRL/W or CTRL/R to display the Attribute Assignment form and the current values for each attribute. (You can use CTRL/W or CTRL/R to clear the screen of disruptions such as broadcast messages.)
RESTORE (GOLD-R)	Restores all attribute values to their status when the Attribute Assignment form was first displayed. For example, if you change a Help text, then decide that you want to keep the original Help text, press RESTORE. All field values are restored including the Help text.
RETURN and ENTER	Tell the form editor to assign the attributes that you have specified to the field. When you press RETURN or ENTER, the form editor displays: • The Field Validator form if you indicate that you want to enter F/V Edit • The Attribute Assignment form if you are assigning attributes to more fields (after selecting either All Fields or New/Modified Fields at the Assign Phase menu) • The Phase Selection menu if you entered the Assign phase from the menu and have no more fields to which you are assigning attributes • The Layout phase if you entered the Assign phase using the ASSIGN-FIELD-ATTRIBUTE key
MENU (GOLD-KP7)	Saves the attributes that you have specified for a field and returns you to the Phase Selection menu. Use the MENU key when you are assigning attributes to a series of fields (All Fields or New/Modified Fields) and want to leave the Assign phase without assigning attributes to the remaining fields.

6.3 Field Attributes

This section describes:

- The meaning of each attribute

- The run-time effect of each response
- How to assign each attribute

6.3.1 Field Name Attribute

The Field Name is the most important field attribute. In order to collect or display information in the field, a TDMS request must associate, or map, the field name of the form field to a field name in a record definition. When you assign a field name to a form field, you should be aware of any naming conventions that are to be used in the application.

Usually, field names serve as self-documenting descriptions of the field. In the Employee sample application, for example, the field names on the Add form are:

EMPLOYEE_NUMBER	STREET	ZIP
FIRST_NAME	CITY	SEX
MIDDLE_INITIAL	STATE	BIRTH_DATE
LAST_NAME		

In addition, the form fields in the Employee sample have the same names as the record fields to which they are mapped. This simplifies the input and output mapping process, and it also allows the person who designs the request to take advantage of the "map to all fields" capability provided by RDU.

Field names must conform to CDD naming requirements. The field name can have up to 31 characters, and the first character must be alphabetic (A-Z). The remaining characters in the field name can include:

- Alphabetic characters (A-Z and a-z)
- Numeric characters (0-9)
- Dollar sign ($)
- Underscore (_)

The last character in a field name *cannot* be either a dollar sign ($) or an underscore (_).

The form editor assigns a default field name for each field. The first field that you create has the default name of F$0001, the second field has the default name of F$0002, and so forth. When the Attribute Assignment form is displayed, the current field name is shown; the current field name is the default field name until and unless you change the default field name.

To replace the current field name, press the LINE FEED key to delete the existing field name. Then, with the field name blank, type in the name that you wish to assign to the field.

You do not automatically delete the existing field name when you type in a new field name; you only delete as many characters as you type in. Therefore, always press LINE FEED to delete the existing contents of this field before replacing a field name.

For example, assume the field representing City in the Employee sample has a current field name of F$0006, the default field name assigned by the form editor. If you type City without first pressing the LINE FEED key, the resulting field name is City06. To make sure that the field is named correctly, first press LINE FEED and then type City.

6.3.2 Default Value Attribute

The Default Value attribute is a value that is displayed in the field at run time unless overridden by a request. A request overrides a default value and places a different value in the field, when either:

- The request includes instructions to display a form and to output a literal string to the form field
- The request includes instructions to display a form and to output data from a record field to the form field

If you map a field for input and you provide a Default Value, the Default Value is returned to the record unless the operator overrides it or unless the request outputs another value to the field. For example, if the Default Value for the field representing State is CA (for California), the field looks like this when it is displayed at run time:

```
State: CA
```

When the cursor is at the State field (assuming that it is mapped for input), the operator can either accept this Default Value (by pressing TAB) or override it by entering another value (for example, NY). When the operator completes the form, the current value displayed in the field is the value that is returned to the record.

Use the Default Value to save operator keystrokes by providing frequently used responses. However, keep in mind the following important notes about the use of Default Values:

- The form editor does not check Default Values against the field picture string or any field validators. If you use a Default Value that does not meet these requirements (for example, having a Default Value of ABC in a numeric field), you may have a run-time error when TDMS displays the form.
- The operator must delete the Default Value of a field by pressing the LINE FEED key *before* overriding the Default Value. If the operator neglects to delete the Default Value, the Default Value is overwritten but not necessarily erased. For example, if the Default Value for the City field is LOS ANGELES, and the operator types in RENO (without first deleting the Default Value by pressing LINE FEED), the field reads RENOANGELES. Therefore, remind the operator to delete the Default Value of the field before overriding the Default Value.

To assign a Default Value to a field, type in the Default Value in the Attribute Assignment form.

6.4 Help Text

The help text for a field is a message that TDMS displays at run time on the message line (line 24) when all of the following occur:

- The cursor is at the field
- The field is open for input
- The operator presses the HELP key (or PF2)

The help text can have up to 80 characters (for an 80-column form) or up to 132 characters (for a 132-column form). Use the help text to provide explanatory messages or special instructions to the operator. For example, you might tell the operator that only numeric input is valid in the field. Remember that the operator can read the message only when the field is open for input; do not use the help text if the field is not mapped for input in a request.

To replace the current help text, clear the field by pressing LINE FEED and then type in the help text. There is no default help text.

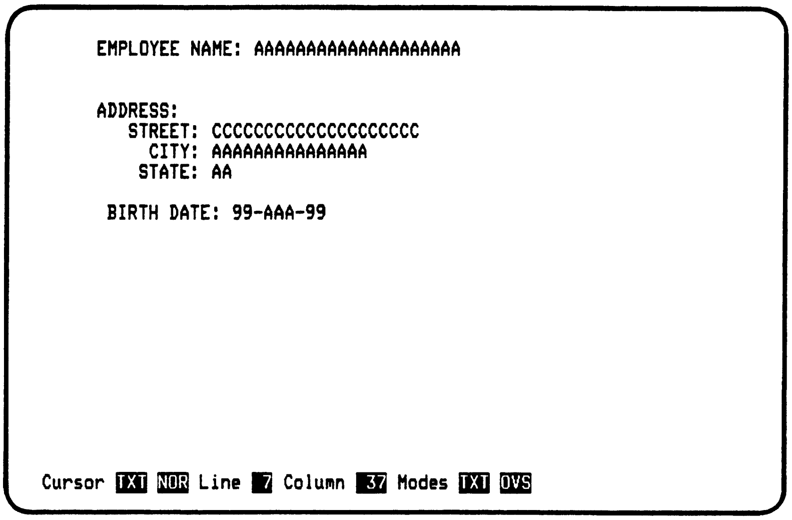
6.4.1 Autotab Attribute

The Autotab attribute automatically moves the cursor at run time to the next input field when the current field is filled. For example, assume a portion of a form looks like this:

```
State: __  Zip Code: _____
```

If you assigned the Autotab attribute to the State field (a two-character field), the cursor automatically moves to the Zip (Postal) Code field as soon as the operator types in two valid characters for the State field. (The characters are valid if they meet the requirements of the picture string, field attributes, and any field validators.)

When the Autotab attribute is assigned to the last field mapped for input on the form, the form is complete and the request completes its execution when the operator fills in the last field. In the following example, assume the BIRTH_DATE field is the last input field and has the Autotab attribute.



```
EMPLOYEE NAME: AAAAAAAAAAAAAAAAAAAA


ADDRESS:
   STREET: CCCCCCCCCCCCCCCCCCCC
     CITY: AAAAAAAAAAAAAAA
    STATE: AA

 BIRTH DATE: 99-AAA-99









Cursor TXT NOR Line 7 Column 37 Modes TXT OVS
```

At run time, typing in the full birth date (for example, 01-DEC-47) has the same effect as entering data in a field and then pressing RETURN at any other time.

Use the Autotab attribute to save operator keystrokes for fields that are frequently filled.

To assign the Autotab attribute, type an X to the right of Autotab (or press TAB if X is already entered). To deassign the Autotab attribute, press the space bar or the LINE FEED key. (Pressing the space bar or LINE FEED key erases the X.)

6.4.2 No Echo Attribute

The No Echo attribute prevents any data from being displayed in a field at run time regardless of whether the data is:

- The field default value
- A record field or literal string output by a request to the form field
- Data entered by the operator

Although the data is not displayed on the terminal, it is returned to the program (when the operator completes the form) if the form field has been mapped to a record field in the request.

Use the No Echo attribute when you do not want someone to accidentally see the data that is input or output to the field (for example, salary information). Keep in mind, however, that the operator's inability to see the field and to check input can significantly increase the likelihood of input error. Therefore, it is often a good idea to use field validators to verify operator input for No Echo fields. You learn about field validators later in this chapter.

To assign the No Echo attribute, type X to the right of No Echo on the Attribute Assignment form. To deassign the No Echo attribute, press the space bar or LINE FEED key.

6.4.3 Display Only Attribute

The Display Only attribute prevents the cursor from moving to the field for input at run time thus preventing a field from accepting operator input. In most instances, the person who designs the request determines which fields are mapped for input. However, there are some instances when you want to map all but a few fields for input (using a request feature that maps all form fields to all record fields). Here, it is helpful to use the form definition to prevent mapping a particular field for input.

See the Employee sample for an example of using the Display Only attribute. The request EMPLOYEE_ADD_REQUEST uses an INPUT %ALL instruction that would ordinarily accept input from all fields whose names match the record definitions named in the request. However, the field EMPLOYEE_NUMBER has the Display Only attribute so that it is not open for input at run time.

Type X to the right of Display Only in the Attribute Assignment form to assign the Display Only attribute; press the space bar or LINE FEED key to deassign it.

6.4.4 Right Justify Attribute

The Right Justify attribute fills in a field from the right margin rather than the left margin. The Right Justify attribute applies to:

- Input from the operator
- A default value for the field
- Output to the form field from a record field

For example, assume the Employee Number field is a seven-character numeric field:

```
Employee Number: _______
```

If you assign the Right Justify attribute and the operator enters the number 1234, the field looks like this:

```
Employee Number: ___1234
```

If you do not assign the Right Justify attribute, data entered by the operator is filled in from the left margin:

```
Employee Number: 1234___
```

To assign the Right Justify attribute, type X to the right of Right Justify on the Attribute Assignment form; to deassign it, press the space bar or LINE FEED key.

6.4.5 Fixed Decimal

The Fixed Decimal attribute provides specific run-time characteristics for an UNSIGNED NUMERIC field. When the operator enters data into a Fixed Decimal field at run time, the data is entered in the field:

1. To the left of the decimal point initially, until the operator presses the period (decimal point) key
2. To the right of the decimal point after the operator presses the period key

The following chart shows the run-time sequence for a Fixed Decimal field with picture characters 9999.99:

Operator Keystroke	**Form Display**
(Initial display)	____.00
1	___1.00
2	__12.00
3	_123.00
.	_123.00
4	_123.40
5	_123.45

Note that the Zero Suppress attribute does not change the display of zeroes to the right of the decimal point in a Fixed Decimal field. The Fixed Decimal attribute can be assigned to a field only when both occur:

- The field has a picture of all 9s
- There is one period (decimal point) in the field, with at least one picture character (9) on each side of the period. No other field identifiers or field constants can be used in a Fixed Decimal field. The following chart shows examples of valid and nonvalid Fixed Decimal fields:

Valid	**Nonvalid**
999.999	NNN.999
9.99	.99
	NNNN.XX
	%99.99

To use a punctuation character at the beginning or end of a Fixed Decimal field (for example, dollar sign ($) or percent sign (%)), define the punctuation mark as background text rather than a picture character. Remember, you can have only one field constant in a Fixed Decimal field, and that field constant must be a period (decimal point).

To assign the Fixed Decimal attribute, type X to the right of Fixed Decimal on the Attribute Assignment form. To deassign the Fixed Decimal attribute, press the space bar or LINE FEED key.

6.4.6 Zero Fill

The Zero Fill attribute determines whether the assigned fill character will be 0 or space.

At run time, TDMS uses the fill character to fill non-data positions of an internal representation of form field data. Any position that contains the clear character in the field on the screen will contain the fill character in the internal representation of the form field. The internal representation eventually is mapped to the record field. After the internal representation is mapped to the record field, the record field may or may not contain the fill character. This depends on the data type of the record field.

For example, if the record field is a NUMERIC STRING data type, the record contains a 0 in non-data positions. If the record field is a TEXT data type, the record field contains the fill character. Refer to Table 6-2 for examples.

If you type the clear character into a field, it echoes to the screen as the clear character and is stored in the internal representation of the field as the fill character. This also means that if you type the fill character into a field, it is echoed to the screen as the clear character and stored in the internal representation of the field as the fill character.

6.4.6.1 Effect of Fill Character on Clear Character — FDU allows you to specify a fill character of 0 with a clear character other than 0. However, at run time, if the fill character is zero, then the clear character is always treated as zero.

If the Zero Fill attribute is assigned, the form field should be defined as NUMERIC, right justified, and have a clear character of 0.

6.4.6.2 Assigning Zero Fill and Deassigning Right Justify — If you assign the Zero Fill attribute, you should also assign the Right Justify attribute. If you assign the Zero Fill attribute and deassign the Right Justify attribute, you run the risk of sending inaccurate data to the record.

For example, you type the value 1234 into a left-justified, seven-digit numeric field (for example, 9999999) with the fill character assigned to 0, the form field contains the value 1234000 and the value 1234000 is sent to the record field.

If you must assign the Zero Fill attribute and deassign the Right Justify attribute, you should be sure to assign the Must Fill attribute to prevent extra zeros from being added to the end of a number. A field defined in this manner cannot contain a zero anywhere in the field. TDMS determines whether a field is completely filled by checking that there are no fill characters in the internal form field representation.

Table 6-2 gives examples showing the effect the clear and fill characters have on the form field value and the value returned to the record field. In this table, the form field is defined with the picture string of 9999999 (UNSIGNED NUMERIC, seven digits) and the value 1234 is entered into the form field.

Table 6-2: Effect of Clear and Fill Characters

Field Attributes	Form Field Value	Record Field Definition	Record Field Value
Zero fill Right Justify Clear = 0	0001234	Unsigned numeric 7 digits	0001234
Zero fill Left Justify Clear = 0	1234000	Unsigned numeric 7 digits	1234000
Blank fill Right Justify Clear = space	1234	Unsigned numeric 7 digits	0001234
Blank fill Left Justify Clear = space	1234	Unsigned numeric 7 digits	0001234
Zero fill Right Justify Clear = 0	0001234	TEXT 7 characters	0001234
Zero fill Left Justify Clear = 0	1234000	TEXT 7 characters	1234000
Blank fill Right Justify Clear = space	1234	TEXT 7 characters	1234
Blank fill Left Justify Clear = space	1234	TEXT 7 characters	1234

To assign the Zero Fill attribute (setting the Fill Character to 0), type X to the right of Zero Fill on the Attribute Assignment form. To deassign the Zero Fill attribute (and thus setting the Fill Character to space), press the space bar or LINE FEED key.

6.4.7 Zero Suppress

The Zero Suppress attribute prevents a zero from being displayed on a form when a numeric record field with a null (zero) value is output to a form field. Use the Zero Suppress attribute to prevent a value of zero from being displayed in a form field.

Note that the Zero Suppress attribute does not change the display of zeros to the right of the decimal point in a Fixed Decimal field.

To assign the Zero Suppress attribute, type X to the right of Zero Suppress on the Attribute Assignment form. To deassign the Zero Suppress attribute, press the space bar or LINE FEED key.

6.4.8 Uppercase Attribute

The Uppercase attribute causes any alphabetic character the operator enters at run time to be displayed on the terminal and sent to the record in uppercase. Use the Uppercase attribute to save the operator from having to press the SHIFT key for fields that should always be uppercase (for example, two-letter state abbreviations).

To assign the Uppercase attribute, type X to the right of Uppercase on the Attribute Assignment form. To deassign the Uppercase attribute, press the space bar or LINE FEED key.

6.4.9 Must Fill Attribute

The Must Fill attribute requires that any input made by the operator (or any value returned by the request from a Default Value that the operator does not modify) fill all of the characters in the field. The Must Fill attribute does not require that the operator enter data; it requires that, if any data is entered, the data fill the entire field. For example, assume that your form has a five-character field for Employee Number:

```
Employee Number: _____
```

If all employee numbers in a company have five digits – no more and no less – you should assign the Must Fill attribute to this field. Then, if the operator enters any data for the Employee Number field, the data must be exactly five characters (digits, if the picture string specifies a numeric picture type). The operator cannot move the cursor to the next field until either filling the field or deleting the entire contents of the field.

When you assign the Must Fill attribute, the operator must also satisfy all other input requirements (for example, Response Required attribute, picture type, field validators) of the form definition. See the Response Required attribute for more information on combining the Response Required and Must Fill attributes. See the Zero Fill attribute for more information about how TDMS determines if the field is full.

To assign the Must Fill attribute, type X to the right of Must Fill on the Attribute Assignment form. To deassign the Must Fill attribute, press the space bar or LINE FEED key.

6.4.10 Response Required Attribute

The Response Required attribute requires that the operator type at least one character in a field or that a value is displayed in the field by the request or the form definition Default Value at run time. Otherwise, the operator cannot move the cursor beyond the field.

When you assign the Response Required attribute to a field in a *scrolled* region, input must be provided to every potential line in the scrolled region. For example, assume that a request maps a scrolled form field to a record field and lets the operator input up to 50 entries. If the form field has the Response Required attribute, then the operator must enter data in all 50 possible lines of the scrolled region before moving on to the next field or pressing the RETURN key.

When you assign the Response Required attribute to a field, the operator must also satisfy all other input requirements (for example, picture string, field validators) of the form definition. Use the Response Required attribute to make sure that necessary fields are not left blank by the operator. You can also use the Response Required attribute with the Must Fill attribute and/or field validators to ensure more specific responses.

If you assign the Response Required attribute and the Must Fill attribute to a field, the operator must respond (or the request must output a Default Value) and must completely fill the field.

If you assign the Response Required attribute and deassign the Must Fill attribute, the operator must enter (or the request must output a Default Value) at least one valid character into the field. However, the operator is not required to fill the field.

If you deassign Response Required and assign Must Fill to a field, the operator must fill the field *only if* he enters one or more characters into the field. That is, the operator has the choice of filling the field or leaving it blank.

When Response Required and Must Fill are used with field validators, the operator must satisfy the requirements of both the attributes that are assigned *and* the field validators.

To assign the Response Required attribute, type X to the right of Response Required on the Attribute Assignment form. To deassign the Response Required attribute, press the space bar or LINE FEED key.

6.4.11 Clear Character Attribute

The Clear Character attribute determines the appearance of the field at run time when the field is blank or partially blank. Usually, the Clear Character is a space ( ) or an underscore (_). For example, if the Clear Character for the seven-character Employee Number field is an underscore and there is no data displayed in the field, the field looks like this at run time:

```
Employee Number:  _______
```

If the Clear Character for the Employee Number field is a space (blank), the field looks like this:

```
Employee Number:
```

When a field is partially filled in, the Clear Character is displayed in that portion of the field that contains no data. For example, a partially filled, left-justified, seven-digit numeric field with the Clear Character defined to be space would look like this:

```
Employee Number:    1234
```

See the section on the Zero Fill attribute for more information about the interaction between the Clear Character and the Fill Character.

Using a space as a clear character often has the advantage of improving the appearance of a run-time form by reducing the amount of clutter on the screen. Using an underscore can have the advantage of showing the operator the number of characters that are permitted in the field. You can achieve a similar effect (of showing the length of the field) by using the input field highlighting feature in the Form phase (Chapter 4).

To assign a Clear Character to a field, type the character that you want to display in the Clear Character field on the Attribute Assignment Form. The default Clear Character is space.

6.4.12 Scale Factor Attribute

The Scale Factor attribute is a positive or negative integer that represents the location of a decimal point in a numeric field. (That is, the scale factor represents a power of 10 by which the number in the form field is multiplied.)

The examples show the effect of the Scale Factor:

Field Picture	Scale Factor	Data Entered on Form	Value Sent to Record
9999	0 (default)	5162	5162
NNNN	2	5162	516200
$999.99	−2 (default)	267.95	267.95
99.9%	−3	14.6	.146

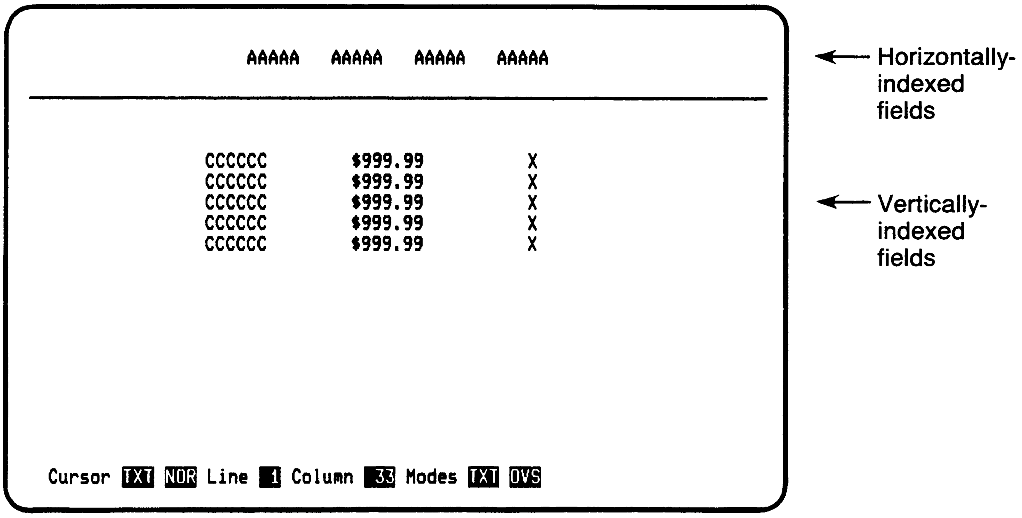
When a numeric field includes a decimal point (period), the default Scale Factor is the number of digits to the right of the decimal point, preceded by a minus sign. For example, in a field with picture 999.99, the default Scale Factor is −2. When a numeric field does not include a decimal point, the default Scale Factor is 0.

You cannot change the default scale factor for a field to which the Fixed Decimal attribute has been assigned. If you want to create a field that includes a decimal point and has a scale factor other than the default (for example, a field with picture 99.99% with scale factor −4), do not assign the Fixed Decimal attribute to that field.

To assign a Scale Factor, type in a positive or negative number next to Scale Factor on the Attribute Assignment form.

6.4.13 Indexed Field Attribute

An **indexed field** is a single form field defined with two or more elements aligned either vertically or horizontally. Indexed fields (like scrolled fields) are **form field arrays**. Form field arrays can be mapped to simple record fields, to record arrays, or to both. See the *VAX TDMS Request and Programming Manual* to learn about mapping between array form fields and records. Figure 6-2 shows examples of indexed fields.



```
          AAAAA   AAAAA   AAAAA   AAAAA

      CCCCCC      $999.99      X
      CCCCCC      $999.99      X
      CCCCCC      $999.99      X
      CCCCCC      $999.99      X
      CCCCCC      $999.99      X

Cursor TXT NOR Line 1 Column 33 Modes TXT OVS
```

Figure 6-2: Examples of Indexed Fields

The following rules govern indexed fields on a form:

- Each element in the indexed field must have the same length, picture string, video attributes, and field attributes.
- Elements in a vertically-indexed field must be aligned in the same columns with no intervening blank lines.
- Elements in a horizontally-indexed field must be on the same line with the same number of columns between each element. (In the example shown in Figure 6-2, each element has a five-character alphabetic picture type, and there are three columns between each of the four elements.) You can include varying background text between elements in a horizontally-indexed field.

To define an indexed field, follow this procedure:

1. Enter the Layout phase to create a series of fields that adhere to the rules listed in previous sections. You can use the cut and paste feature in the Layout phase to make sure that each of the elements has the same picture type, video attributes, and (for horizontally-indexed fields) spacing between elements.

2. Enter the Assign phase to assign attributes to the top element in a vertically-indexed field or the leftmost element in a horizontally-indexed field. Assign those attributes that you want to apply to the indexed field (for example, Autotab or Right Justify), and type V (for vertically-indexed) or H (for horizontally-indexed) at the Indexed attribute on the Attribute Assignment form. (The default value N indicates that the field is not an indexed field.) The cursor then moves to the Index Count attribute.

3. Use the Index Count attribute to identify the number of elements in the indexed field. In the examples shown in Figure 6-2, the vertically-indexed field has an Index Count of 5 (for the five vertically-aligned elements) and the horizontally-indexed field has an Index Count of 4. The form editor does not allow you to assign an Index Count that is less than 2 or greater than the number of valid elements in the indexed field. If you assign N to the Indexed attribute (indicating no indexing), any value entered in the Index Count attribute is ignored.

   When you finish assigning attributes (including a field validator, if desired) to the first element in the indexed field, the form editor creates the indexed field based on the information that you supplied for the Indexed and Index Count attributes. That is, the form editor automatically assigns the field name and all other field attributes to each of the elements in the indexed field.

   Any subsequent attribute changes that you make to any one element of the indexed field apply to all of the elements of the indexed field. If, after initially defining the indexed field, you reenter the Assign phase for the indexed field, each of the elements in the field is shown blinking, bolded, and underlined.

6.5 Field Validators

A TDMS field validator allows you to specify valid data for fields, in addition to the picture string restrictions assigned during the Layout phase of the form editor. Field validators can include a range of choices, a list of choices, or certain other requirements. You can assign only one field validator to a field. Table 6-3 shows the types of field validators that TDMS provides.

Table 6-3: TDMS Field Validators

Validator	Function
Range	Checks operator data input against numeric or alphabetic range.
Choice	Checks operator data input against list of choices.
Size (Byte,Word, Longword, Quadword)	Checks that numeric data is within certain ranges (to avoid data type conversion errors).
Check Digit	Applies a predefined algorithm to validate a numeric field.

Range field validators let you specify a numeric or alphabetic range to a field on a form. Any operator input to that field must conform to the range that you specify. You may specify more than one valid range for a field.

Choice field validators let you identify a list of valid choices for a field. At run time, the operator can enter data in the field only if the data matches an entry in the list of choices. When you use this field validator, you can also specify:

- Abbreviations
- Case sensitivity (letting the operator enter either uppercase or lowercase or requiring exact case matches)

Size field validators let you specify predefined ranges for numeric form fields. Size validators can be used to specify:

- UNSIGNED BYTE (BYTE logical)
- SIGNED BYTE (BYTE integer)
- UNSIGNED WORD (WORD logical)
- SIGNED WORD (WORD integer)
- UNSIGNED LONGWORD (LONGWORD logical)
- SIGNED LONGWORD (LONGWORD integer)
- SIGNED QUADWORD (QUADWORD integer)

Check Digit field validators let you apply a predefined algorithm to operator input. You can assign a Check Digit validator only to an UNSIGNED NUMERIC field (picture string all 9s).

Characters from the DEC Multinational Character Set are valid for Range and Choice validators. Range and choice string validation is performed using the DEC Multinational Character Set.

6.5.1 Run-Time Effect of Field Validators

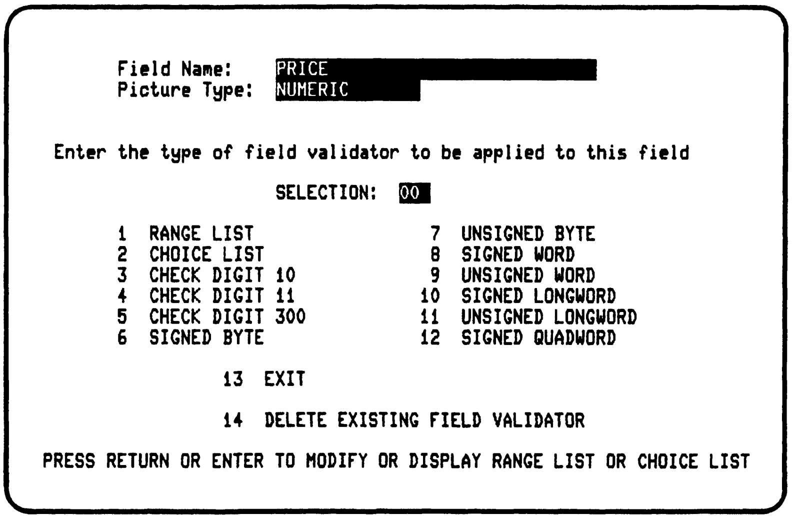
At run time, TDMS verifies that the values the operator enters are consistent with the values defined by the field validator. If the operator attempts to enter data that does not meet the requirements of the field validator, the terminal signals an error by ringing the bell and displaying a message on line 24 of the operator's terminal. When this occurs, the invalid data is not sent to the record, and the operator cannot move to the next input field until he or she enters a valid response into the field. (An empty field is considered to be a valid response unless the Response Required attribute has been assigned to the field.)

6.5.2 Assigning Field Validators

You assign validators when assigning attributes to a field in the Assign phase. The Attribute Assignment form tells you if a field validator is already assigned to the field with this message:

```
[NO] Validators exist for this field; Do you want to enter F/V Edit?
[Y/N]
```

To assign validators to a field, type Y to this question and press RETURN. The form editor displays the Field Validator form, as shown in Figure 6-3. In this example, the field is a five-digit UNSIGNED NUMERIC field (picture characters: $999.99) named PRICE.



```
        Field Name:       PRICE
        Picture Type:     NUMERIC

   Enter the type of field validator to be applied to this field

                      SELECTION: 00

        1  RANGE LIST              7  UNSIGNED BYTE
        2  CHOICE LIST             8  SIGNED WORD
        3  CHECK DIGIT 10          9  UNSIGNED WORD
        4  CHECK DIGIT 11         10  SIGNED LONGWORD
        5  CHECK DIGIT 300        11  UNSIGNED LONGWORD
        6  SIGNED BYTE            12  SIGNED QUADWORD

                  13  EXIT

                  14  DELETE EXISTING FIELD VALIDATOR

  PRESS RETURN OR ENTER TO MODIFY OR DISPLAY RANGE LIST OR CHOICE LIST
```

Figure 6-3: Field Validator Form

TDMS displays the field name, the field picture type, and the corresponding field validators that are currently assigned to the field. The cursor is positioned at the SELECTION prompt.

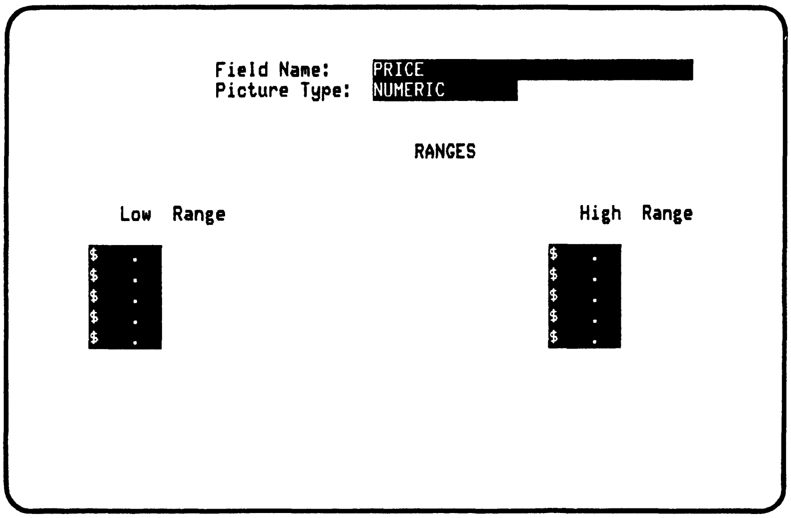
To assign a validator, or to change the current field validator, type the number of your selection and press RETURN. If you want to delete a field validator that has already been assigned to a field, enter 14 at the SELECTION prompt.

6.5.3 Assigning the Range Validator

The Range validator allows you to define one or more ranges (up to 127 different ranges) for valid operator input. You can include numeric ranges, alphabetic ranges, or both.

The range list is a scrolled region that lets you see up to five lines at a time.

To assign a Range validator, type 1 to the right of SELECTION on the Field Validator form and press RETURN. The form editor displays the Range List form, the name and picture type of the field, and any field constants that are part of the field. For example, Figure 6-4 shows the Range List form for a field named PRICE defined with the picture string $999.99:



```
                Field Name:   PRICE
                Picture Type: NUMERIC

                           RANGES

        Low  Range                          High  Range

       $   .                               $   .
       $   .                               $   .
       $   .                               $   .
       $   .                               $   .
       $   .                               $   .
```

Figure 6-4: Range List Form

The cursor is on the first line of the Low Range. Type in the Low Range, then press TAB to move to High Range. Type in a High Range, then either:

- Press TAB to enter another Low and High Range value
- Press RETURN to exit from the Field Validator portion of the Assign phase

For example, assume that the field PRICE represents the prices of either single pairs of shoes or cases of shoes with each case containing 10 pairs. The least expensive pair of shoes costs $19.95, and the most expensive pair costs $39.95. Therefore, valid entries for PRICE could be from $19.95 to $39.95 (for single pairs) or $199.50 to $399.50 (for cases of 10).

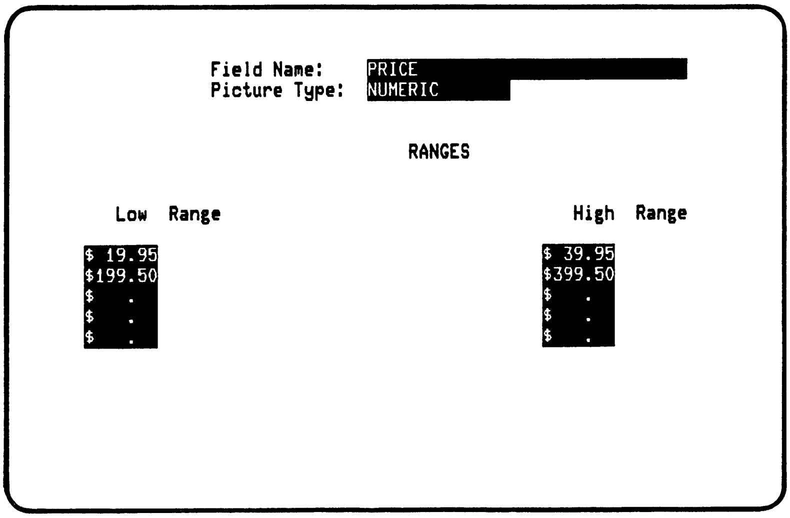
To assign the Range validator to a field with the Left Justify or Right Justify attribute, follow this procedure:

1. Select the Range validator on the Field Validator form (Type 1 in the space next to SELECTION.)

2. Enter the value 01995, or press right arrow and type 1995 with the cursor on the first line of Low Range. (The cursor automatically moves across the dollar sign and period field constants.)

3. Type 03995 (or press the right arrow key and type 3995) with the cursor on the first line of High Range and press TAB to move to the next line.

4. Type 19950 on the second line of Low Range, then TAB to High Range and type 39950. The Range List form now looks like this:



```
                 Field Name:     PRICE
                 Picture Type:   NUMERIC

                            RANGES

     Low  Range                                High  Range

   $ 19.95                                   $ 39.95
   $199.50                                   $399.50
   $   .                                     $   .
   $   .                                     $   .
   $   .                                     $   .
```

5. Press RETURN when you are done. The Range validator is assigned, and you exit from the validator portion of the Assign phase.

At run time, TDMS accepts only operator input that is between (and including) $19.95 to $39.95 or $199.50 to $399.50.

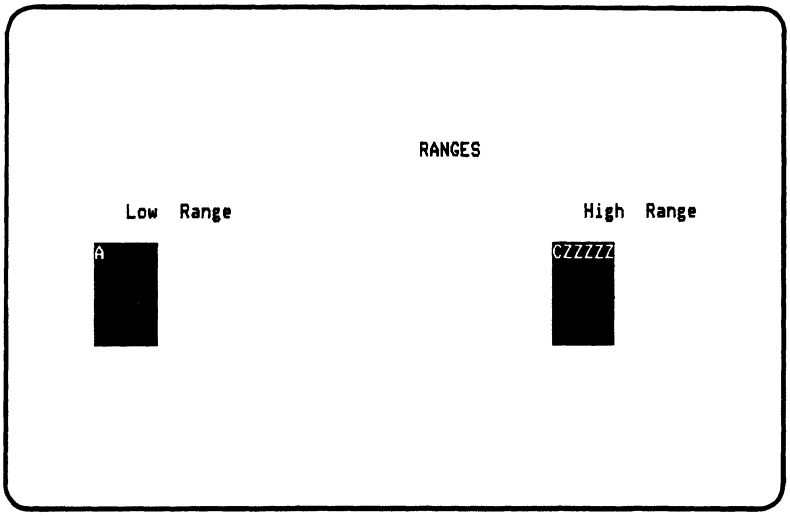
For a field with the Fixed Decimal attribute, the procedure is slightly different.

- Select the Range validator by entering 1 on the Field Validator form.
- Enter the value 19 when the cursor is on the decimal point of the Low Range field, then press the period (.) key and enter 95. Press TAB to go to the High Range.
- Enter the value 39 when the cursor is on the decimal point of the High Range field, then press the period key and enter 95. Press TAB to go to the next line.

- Enter the value 199 when the cursor is on the decimal point of the Low Range field, then press the period key and enter 50. Press TAB to go to the High Range.
- Enter the value 399 when the cursor is on the decimal point of the High Range field, then press the period key and enter 50.

At run time, TDMS accepts only operator input that is between (and including) $19.95 to $39.95 and $199.50 to 399.50.

You can also include alphabetic ranges in a field validator. For example, to define an alphabetic range that allows any entry up to six letters beginning with A, B, or C, the Range validator list would read:



```
                    RANGES

      Low  Range                    High  Range
      A                             CZZZZZ
```

Table 6-4 shows the function keys you can use on the Range List Form.

Table 6-4: Range List Form Function Keys

Key Sequence	Function
TAB	Moves from Low Range to High Range, or from High Range to next line.
BACK SPACE (F12)	Moves from High Range to Low Range, or from Low Range to High Range on previous line.

(continued on next page)

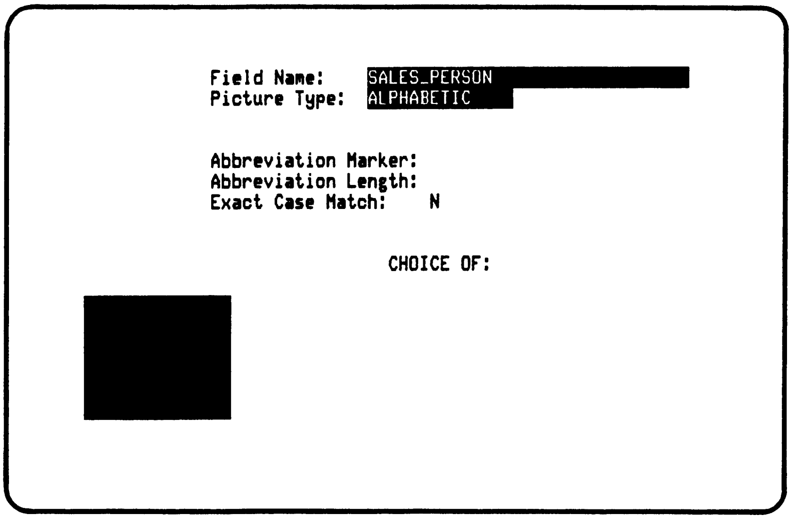
Table 6-4: Range List Form Function Keys (Cont.)

Key Sequence	Function
LINE FEED (F13)	Deletes the contents of the line.
KP0	Inserts a blank line.
KP-hyphen	Deletes the current line.
RETURN or ENTER	Saves ranges and exits from validators.
RESTORE (GOLD-R)	Restores range list to its status when you entered Range form.
CTRL/R or CTRL/W	Refresh the screen.

6.5.4 Assigning the Choice Validator

The Choice validator allows you to define a list of choices for valid operator input. You can specify abbreviations for the choices, and you can specify whether the operator must match the case (uppercase or lowercase) of the choices.

To assign the Choice Validator, type 2 to the right of SELECTION on the Field Validator form and press RETURN. The form editor displays the Choice List form, including the name and picture type of the field. For example, assume that you have a 12-character alphabetic field named SALES_PERSON. Figure 6-5 shows the Choice List form for this field.



```
Field Name:    SALES_PERSON
Picture Type:  ALPHABETIC

Abbreviation Marker:
Abbreviation Length:
Exact Case Match:    N

                    CHOICE OF:
```

Figure 6-5: Choice List Form

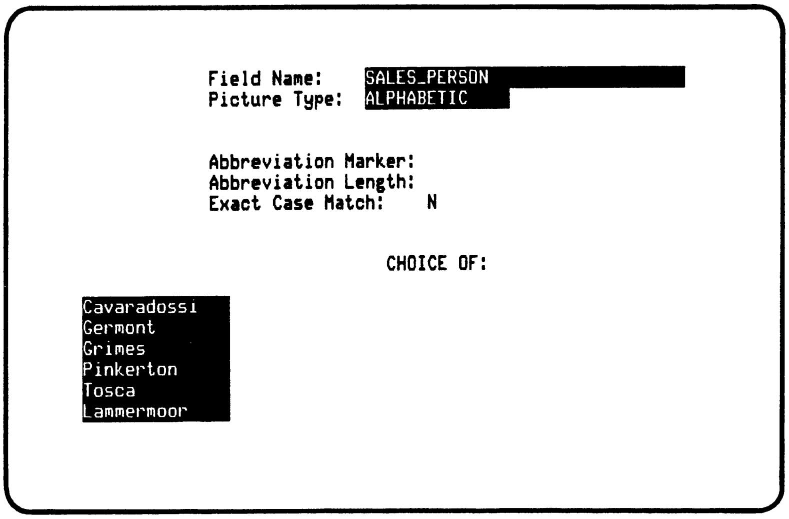
To identify valid choices, type in *one valid choice per line.* Move to the next line of the Choice List Form using the TAB key. Table 6-5 lists the function keys that you can use on the Choice List Form.

Table 6-5: Choice List Form Function Keys

Key Sequence	Function
TAB	Moves to next line.
BACK SPACE (F12)	Moves to previous line.
LINE FEED (F13)	Deletes the contents of the line.
KP0	Inserts a blank line.
KP-hyphen	Deletes current line.
GOLD-up arrow	Moves the cursor to Abbreviation/Case area.
RETURN or ENTER	Saves choices and exit from validators.
RESTORE (GOLD-R)	Restores choices to their status when you entered the Choice List form.
CTRL/W or CTRL/R	Refresh the screen.

You can enter up to 255 valid choices. The choice list is a scrolled region that lets you see up to six lines at a time.

For example, assume that the field SALES_PERSON represents the name of the individual responsible for a sale. You have six sales people whose last names are Cavaradossi, Germont, Grimes, Pinkerton, Tosca, and Lammermoor. Select the Choice validator on the Field Validator form and enter each name on the Choice List form. Type only one name per line and press TAB to move to the next line. Figure 6-6 shows the result:



```
                    Field Name:      SALES_PERSON
                    Picture Type:    ALPHABETIC

                    Abbreviation Marker:
                    Abbreviation Length:
                    Exact Case Match:    N

                                      CHOICE OF:

           Cavaradossi
           Germont
           Grimes
           Pinkerton
           Tosca
           Lammermoor
```

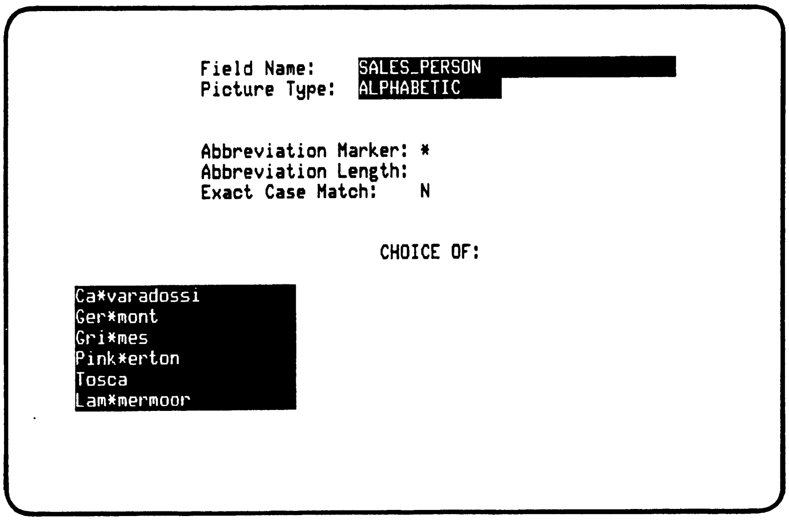
Figure 6-6: Example of a Choice List

If you press RETURN, the only valid entries at run time would be one of those six names. However, you can simplify operator input by allowing abbreviations as described in the next section.

6.5.4.1 Abbreviation Marker — The Abbreviation Marker feature allows you to insert a character within one or more items on the Choice list that denotes an acceptable abbreviation.

To assign an Abbreviation Marker, identify the marker that you want to use by typing it next to the Abbreviation Marker prompt. You can then include that marker once in each entry in the Choice list. You cannot use an Abbreviation Marker for a SIGNED or UNSIGNED NUMERIC field, and you cannot use an Abbreviation Marker for a field to which the Right Justify attribute is assigned.

For example, you could insert an asterisk (*) in each name on the Choice list as shown in Figure 6-7, and then identify the asterisk as the Abbreviation Marker.



```
          Field Name:     SALES_PERSON
          Picture Type:   ALPHABETIC

          Abbreviation Marker: *
          Abbreviation Length:
          Exact Case Match:    N

                          CHOICE OF:

    Ca*varadossi
    Ger*mont
    Gri*mes
    Pink*erton
    Tosca
    Lam*mermoor
```

Figure 6-7: Example of Abbreviation Markers

By inserting the asterisks in some of the choices and identifying the asterisk as the Abbreviation Marker, you define the following valid abbreviations:

Ca
Ger
Gri
Pink
Tosca (no abbreviation; marker not used)
Lam

The operator can enter the abbreviation, the full name, or a partial name that includes at least the abbreviation. These are some valid operator entries for this field:

Ca	Cavara	Cavaradossi
Ger	Germo	Germont
Gri	Grim	Grime

6.5.4.2 Abbreviation Length — As an alternative to the Abbreviation Marker, you can specify a valid Abbreviation Length, which is applied to each item on the Choice list.

To specify an Abbreviation Length, move to the Abbreviation Length prompt and type the appropriate number. For example, if you choose an Abbreviation Length of 3 for the list in Figure 6-6, a valid operator entry is one that includes at least the first three letters of each name. With an Abbreviation Length of 3, some examples of valid responses include:

```
Cav         Cavara         Cavaradossi
Ger         Germo          Germont
Gri         Grim           Grime
Tos         Tosc           Tosca
```

You cannot specify both an Abbreviation Length and an Abbreviation Marker.

6.5.4.3 Exact Case Match — The Exact Case Match feature allows you to require that operator input be the same case (uppercase or lowercase) as that which appears on the Choice list. The default is No Exact Case Match, which permits the operator to enter data in either uppercase or lowercase.

For example, if you required an exact case match for the list in Figure 6-6 (with no abbreviations allowed), the only valid choices would be:

```
CAVARADOSSI
GERMONT
GRIMES
PINKERTON
TOSCA
LAMMERMOOR
```

However, if you accept the default and do not require an exact case match, some examples of valid choices are:

```
cavaradossi
CAVARADOSSI
pinkerton
PINKerton
TOSCA
ToScA
```

You can use the Exact Case Match feature with Abbreviation Markers and Abbreviation Length.

If you require Exact Case Match for a field and all of the choices are uppercase, you should assign the Uppercase attribute to the field. Then, all operator input to the field is displayed, validated, and sent to the record as uppercase characters.

To require responses that match the case of the entries on the Choice list, move the cursor to the Exact Case Match prompt and type Y.

6.5.5 Size Field Validators

Size field validators are predefined ranges for numeric fields. On input, Size validators prevent the operator from entering data that is not within the range of the validator.

A Size validator determines the field data type. Therefore, the use of the Size validator can affect the validity of both input and output mappings to a field. See the *VAX TDMS Reference Manual* for tables showing valid input and output mappings for form field data types.

Table 6-6 shows the seven types of Size validators that you can use.

Table 6-6: Numeric Ranges for Size Validators

Size Validator	Range
SIGNED BYTE	−128 to 127
UNSIGNED BYTE	0 to 255
SIGNED WORD	−32768 to +32767
UNSIGNED WORD	0 to 65535
SIGNED LONGWORD	−2**31 to (+2**31)−1
UNSIGNED LONGWORD	0 to (2**32)−1
SIGNED QUADWORD	−2**62 to (+2**63)−1

TDMS allows you to assign a Size validator to a field only when:

- The field is numeric.
- The number of picture characters in the field picture is equal to or greater than the number of digits in the size of the validator. (You can include any field constants within the field picture.) Table 6-7 shows the minimum required field picture for each Size validator.

Table 6-7: Minimum Required Field Pictures for Size Validators

Size Validator	Minimum Field Picture Required
SIGNED BYTE	NNNN or 9999
UNSIGNED BYTE	999
SIGNED WORD	NNNNNN or 999999
UNSIGNED WORD	99999
SIGNED LONGWORD	11 Ns or 11 9s
UNSIGNED LONGWORD	10 9s
SIGNED QUADWORD	20 Ns or 20 9s

The Size field validator can be very useful in avoiding data conversion errors at run time. For example, if you identify a three-character NUMERIC input field (999), the operator would normally be permitted to enter any value between 0 and 999. However, the programmer may only provide one byte of space for this value in the program. In this instance, the UNSIGNED BYTE validator should be assigned to that input field to avoid a run-time data conversion error. If the UNSIGNED BYTE validator is assigned and the operator attempts to enter a value greater than 255, both occur:

- An error message appears at the bottom of the terminal screen
- The data is *not* sent to the program, avoiding a run-time data conversion error

To assign a Size field validator, type in the appropriate number at the SELECTION: prompt of the Field Validator form.

6.5.6 Check Digit Field Validators

In applications involving identification numbers, each number can be verified for accuracy by using a Check Digit. The Check Digit is an extra digit placed at the end of the normal number. A specific algorithm is applied to all but the final digit of the number; operator entry is verified by comparing the result of the algorithm to the final (check) digit. If the algorithm result and check digit do not match, the cursor does not leave the field and the operator must reenter a number that satisfies the Check Digit validator.

There are three Check Digit field validators available with TDMS, Check Digits 10, 11, and 300. A Check Digit validator can be assigned to a field only if the field has an UNSIGNED NUMERIC data type (picture string of all 9s) and is at least two characters long.

The algorithm and an example of each Check Digit validator is described in the following sections. The basic number identified in each algorithm is the number entered by the operator *excluding* the final digit. For example, if the operator enters the number 123456, the basic number is 12345 and the Check Digit is 6.

6.5.6.1 Check Digit 10 — The algorithm for Check Digit 10 is as follows:

1. Multiply the units position and every alternate position of the basic number by 2 (moving from right to left).
2. Add the digits in the products to the digits in the basic number that were not multiplied.
3. Subtract the sum from the next higher number ending in zero. The difference is the Check Digit.

For example, assume the operator enters 612481 in the field:

	Basic Number	**6**	**1**	**2**	**4**	**8**	
Step 1	Result of multiplication.	12		4		16	
Step 2	Digits in products added together plus digits that were not multiplied.	+3	+1	+4	+4	+7	= 19
Step 3	Next highest number ending in zero minus the sum obtained from Step 2. The difference is the Check Digit.	20	− 19	=	1		

The Check Digit is 1, so the entry 612481 satisfies the Check Digit 10 validator.

6.5.6.2 Check Digit 11 — The algorithm for Check Digit 11 is as follows:

1. Assign a weighting factor of 2 to 7 to each digit position of the basic number. The weighting factor begins with the unit position of the basic number and moves from right to left toward the higher order digits. (The 2 through 7 weighting factors are repeated if the basic number has more than six digits.)

2. Multiply each digit by its weighting factor.
3. Add the products.
4. Divide the sum by 11.
5. Subtract the remainder from 11.
6. If the difference is less than 10, the difference is the Check Digit. If the difference is greater than 9, subtract 10 from it; the resultant difference is then treated as the Check Digit.

For example, assume the operator enters 123455 in the field:

	Basic Number	**1**	**2**	**3**	**4**	**5**	
Step 1	Weighting Factor.	6	5	4	3	2	
Step 2	Result of multiplication.	6	10	12	12	10	
Step 3	Add the products.	6	+10	+12	+12	+10	= 50
Step 4	Divide the sum by 11.	50	− 11	= 4	with a remainder of 6		
Step 5	Subtract the remainder from 11.	11	− 6	= 5			
Step 6	Check Digit.	5					

The Check Digit is 5, so the entry 123455 satisfies the Check Digit 11 validator.

6.5.6.3 Check Digit 300 — The algorithm for Check Digit 300 is as follows:

1. Multiply the first (highest order) digit and every alternate digit of the basic number by 2 (moving from left to right).
2. Add the digits in the resulting product to the digits in the basic number that were not multiplied.
3. Subtract this sum from the next higher number ending in zero. The difference is the Check Digit.

For example, assume the operator enters 57645 in the field:

	Basic Number	**5**	**7**	**6**	**4**	
Step 1	Result of multiplication.	10		12		
Step 2	Digits in products added together plus digits that are not multiplied.	1	+7	+3	+4	= 15
Step 3	Next highest number ending in zero minus the sum obtained from Step 2. The difference is the Check Digit.	20	− 15	= 5		

The Check Digit is 5, so the entry 57645 satisfies the Check Digit 300 validator.

Assigning Field Order 7

Use the Order phase of the form editor to specify the order in which fields are accessed for input at run time. You can use the Order phase to:

- Check the existing access order of fields. The **access order** is the order in which the cursor moves from input field to input field at run time. The form editor keeps track of the field access order on the access order list.
- Change the existing access order of fields.

The Order phase of the form editor is optional. If you do not use the Order phase, fields mapped for input (by a request) are accessed in the default order (that is, the order in which you created the fields). Only the form definition can specify the access order of fields on a form; reordering of fields cannot be done in a request.

When a form is displayed at run time, the cursor is positioned at the field that is both:

- Mapped for input
- At the beginning of the field access order list

In the Add form of the Employee sample, for example, the Employee Number field is first on the access order list and the First Name field is next. However, since the Employee Number field is not mapped for input (it is a Display Only field, and the Employee Number is mapped from the record to the form based on previous operator input), the cursor is positioned at the First Name field when the form is displayed.

Note that while the Order phase determines field access order for the form, the input instructions within the request, not the order list, determine which fields are available for input.

7.1 Default Field Access Order

When you define fields in a form definition (in the Layout phase), the form editor establishes a default field access order that is the order in which you create the fields. The first field that you create is the first field on the access order list, and the last field that you create is the last field on the access order list.

Any time you add a field, it is placed at the end of the access order list. Similarly, a field is placed at the end of the access order list any time that you insert the field on the form using either the UNDELETE-LINE or PASTE key.

7.2 Using the Order Phase

You can enter the Order phase of the form editor only from the Phase Selection menu. At the Phase Choice prompt of the Phase Selection menu, type Order (or just O) and press RETURN; the form editor displays the form layout. (The form layout is the background text, picture characters, and video highlights that you defined in the Layout phase.)

The form editor provides several function keys that you use in the Order phase, as listed in Table 7-1. The remainder of this chapter discusses how to use these function keys to reorder fields.

Table 7-1: Order Phase Function Keys

Key	Function
TAB	Moves cursor to the next field on the access order list.
BACK SPACE (F12)	Moves cursor to the previous field on access order list.
SELECT (KP period)	Identifies the first, then subsequent, fields on the access order list.
STANDARD-ORDER (GOLD-C)	Resets all fields to be ordered from left-to-right, top-to-bottom. The access order of scrolled and indexed fields is based on the first field that is scrolled or indexed.

(continued on next page)

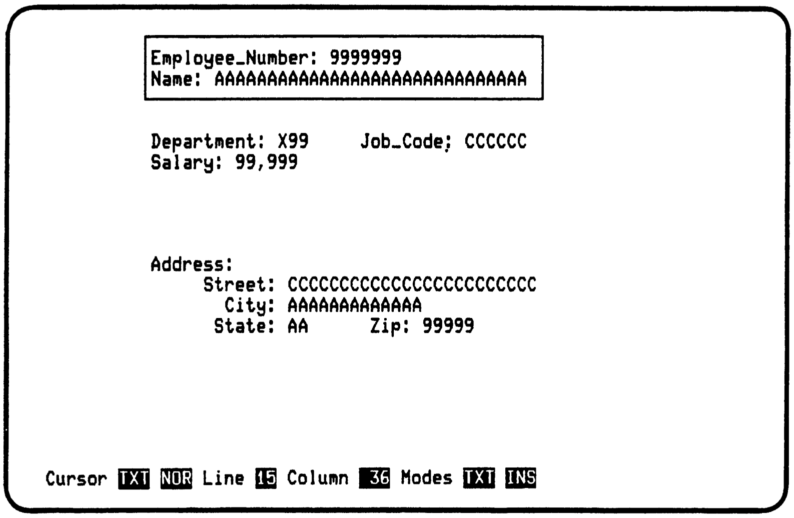
Table 7-1: Order Phase Function Keys (Cont.)

Key	Function
RESTORE (GOLD-R)	Restores the access order list to the same status as when you entered the Order phase.
MENU (GOLD-KP7)	Saves current access order and returns you to Phase Selection menu.
RETURN	Revises the access order list and stays in the Order phase.
GOLD-KP period	Restores the access order list to its status as when you last used the RETURN key.

7.2.1 Determining the Current Field Access Order

When the form editor displays the form layout at the beginning of the Order phase, the cursor is positioned at the field that is currently first in the access order list. Press TAB to move to the second field on the list; each time you press TAB the cursor moves to the next field on the field access order list. (Press BACK SPACE to move the cursor to the previous field on the access order list.) The form editor signals an error if you press TAB when the cursor is at the last field on the access order list, or if you press BACK SPACE when the cursor is at the first field on the access order list.

For example, examine the following form definition as it appears during the Order phase:


```
Employee_Number: 9999999
Name: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Department: X99     Job_Code: CCCCCC
Salary: 99,999


Address:
     Street: CCCCCCCCCCCCCCCCCCCCCCCC
       City: AAAAAAAAAAAAA
      State: AA       Zip: 99999




Cursor TXT NOR Line 15 Column 36 Modes TXT INS
```

Assume that you created the fields on the form from left-to-right, top-to-bottom. When you first enter the Order phase (after having used the Form, Layout, and Assign phases), the cursor is at the first 9 in the Employee_Number field. Press TAB and the cursor moves to the Name field; each time you press TAB, the cursor moves to the next field on the access order list. When the cursor is at the Zip field, the form editor signals an error if you press TAB, because Zip is the final field on the access order list.

7.2.2 Creating a Left-to-Right, Top-to-Bottom Field Access Order

The Order phase includes the STANDARD-ORDER key (GOLD-C), which automatically reorders the fields on a form to be left-to-right, top-to-bottom. You can use the STANDARD-ORDER key at any time while in the Order phase. Press the sequence GOLD-C. All fields are ordered from left-to-right, top-to-bottom, and the cursor moves to the first field in the new access order list.

7.2.3 Changing Field Access Order

If you want a field access order *other than* left-to-right, top-to-bottom, follow this procedure:

1. Enter the Order phase from the Phase Selection menu.

2. Move the cursor to the field that is to be the *first* field in the new access order. Press TAB to move the cursor to the next field on the current access order list and press BACK SPACE to move the cursor to the previous field on the access order list.

3. Press SELECT (KP period). When you first press SELECT, you tell TDMS that the field on which the cursor is positioned is at the beginning of the new access order list.

4. Move to the field that is to be second on the access order list using TAB or BACK SPACE. When the cursor is positioned on the second field, press SELECT. Each time you press SELECT, you tell TDMS to insert the field on which the cursor is positioned next on the access order list.

5. Continue to move the cursor to successive fields on the access order list by pressing SELECT to add a field to the access order list.

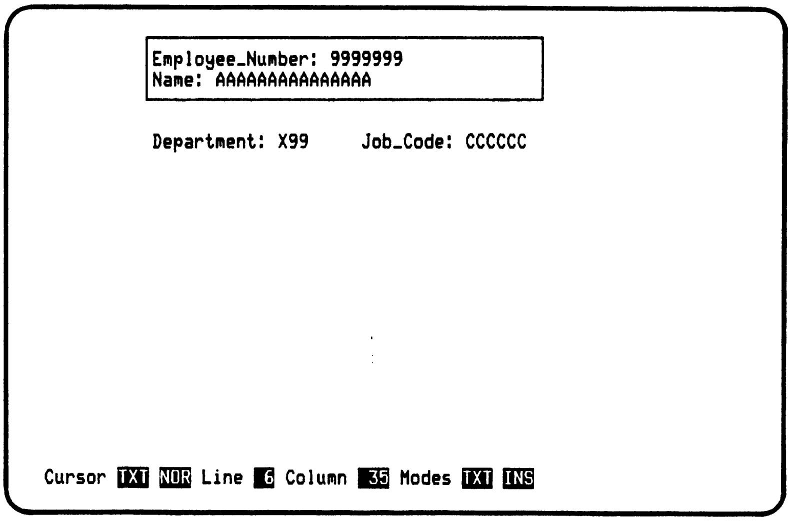
6. Press MENU (GOLD-KP7) when you have gone through each field on the form and have finished reordering fields. This tells TDMS to rewrite the access order list and return to the Phase Selection menu.

   When reordering fields, it is good practice to explicitly order *each* field on the form. It is possible to reorder a subset of fields (for example, you can reorder the first, second, and third fields on a form to be third, first, and second, respectively). However, some combinations of subset reordering can produce unintended results.

7. Check the new access order by returning to the Order phase from the Phase Selection menu, then pressing the TAB key to move through the fields in the new access order.

   It is good practice to check the new field access order each time you reorder fields on a form.

The following example demonstrates the reordering process:



```
Employee_Number: 9999999
Name: AAAAAAAAAAAAAAA

Department: X99     Job_Code: CCCCCC








Cursor TXT NOR Line 6 Column 35 Modes TXT INS
```

In this form, the fields were created from left-to-right, top-to-bottom. Therefore, the original access order list is as follows:

```
Employee_Number
Name
Department
Job_Code
```

To reorder the fields so that you enter the Department first, then Job_Code, Name, and Employee_Number, you would follow these steps:

1. Enter the Order phase from the Phase Selection menu.
2. Press TAB twice to move the cursor from the first character in Employee_Number to the first character in Department.
3. Press SELECT (KP period) to indicate that Department is the first field on the access order list; then press TAB to move the cursor to the field Job_Code.
4. Press SELECT to indicate that Job_Code is next (second) on the access order list; then press BACK SPACE twice to move the cursor (backwards) to field Name.
5. Press SELECT to indicate that Name is next (third) on the access order list; then press BACK SPACE once to move the cursor to field Employee_Number.

6. Press SELECT to indicate that Employee_Number is next (fourth and last) on the access order list.

7. Press MENU (GOLD-KP7) to tell TDMS to rewrite the new access order list and return to the Phase Selection menu.

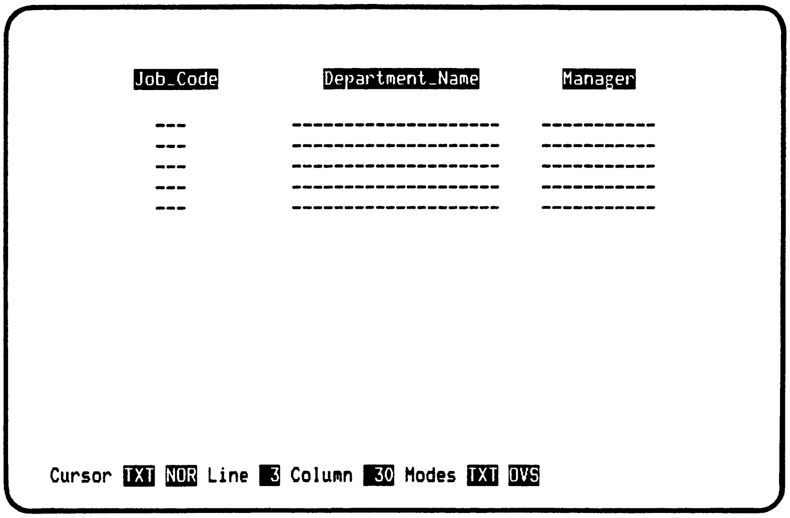
The access order list is now:

```
Department
Job_Code
Name
Employee_Number
```

7.2.4 Run-Time Cursor Movement and Access Order in Scrolled Fields

When you identify a field as scrolled (in the Layout phase), the run-time cursor movement and access order are affected. In a scrolled region that includes only one field, the cursor moves from top to bottom through the field at run time if the field is mapped for input. When the operator completes the first line of the field (by pressing TAB or filling a field to which the Autotab attribute has been assigned), the cursor moves to the second line of the scrolled region. To leave the scrolled region, the operator presses the sequence GOLD-down arrow to move from the scrolled region to the next field (outside the scrolled region) or the sequence GOLD-up arrow to move to the previous field.

In a scrolled region that contains more than one field, the cursor normally moves from left to right on each line, then down through the scrolled region. The following example shows a scrolled region with three fields:



```
          Job_Code          Department_Name           Manager

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





  Cursor TXT NOR Line 3 Column 30 Modes TXT OVS
```

If all the fields are mapped for input in the request, the cursor first moves to the field Job_Code, then to the first row of field Department_Name, and then to the first row of field Manager. The cursor then moves to the second line of the scrolled region, continuing to move left-to-right, then top-to-bottom. The operator leaves the scrolled region using the GOLD-down arrow sequence to move to the next field outside the scrolled region, or GOLD-up arrow to move to the previous field outside the scrolled region.

You can use the function keys in the Order phase to change the access order of the entire scrolled region, but you cannot change the order of the individual fields within a scrolled region. For instance, in the previous example, if Job_Code, Department_Name, and Manager had been fields number 3, 4, and 5 on the access order list, the form editor allows you to reorder the scrolled region so that the three fields could be 1, 2, and 3 on a new access order list. You could *not*, however, reorder Job_Code, Department_Name, and Manager to be 1, 4, and 7; you also could not reorder them within the scrolled region to be 3, 2, and 1 (Manager, Department_Name, Job_Code).

Saving the Form 8

Use the Exit phase of the form editor to leave the form editor and return to FDU command level. To enter the Exit phase, type EXIT (or E) at the Phase Selection menu and press RETURN. The form editor displays the question:

```
Do you want to save this form? Y
```

To save the form definition that you have just created or modified, press RETURN to accept the default value Y (yes). The form definition is saved and stored in the CDD (in the location that you specified in the CREATE, MODIFY, or REPLACE FORM command), and you return to FDU command level (FDU> prompt).

If you want to leave the form editor without saving the edits that you made in this session, type N (to replace the default value Y) and press RETURN. The form definition is not saved, and you return to FDU command level. If you entered the form editor by issuing either the MODIFY FORM or REPLACE FORM command, the form definition that originally existed in the CDD location is not modified or replaced.

Using VAX TDMS with VAX DATATRIEVE 9

If both VAX TDMS and VAX DATATRIEVE are installed on your system, you can use a TDMS form for input and output with a DATATRIEVE domain. This chapter describes how:

- To prepare TDMS form definitions for use in a DATATRIEVE application
- To convert FMS forms to TDMS forms for use in a DATATRIEVE application

The *VAX DATATRIEVE User's Guide* describes how to develop a DATATRIEVE domain that can use the forms to collect and/or display information.

Before reading this chapter you should be familiar with the concept and use of request library definitions, request library files, and the Request Definition Utility (RDU) as discussed in the *VAX TDMS Request and Programming Manual.*

9.1 Preparing a TDMS Form for Use in a DATATRIEVE Application

In order to use a TDMS form definition in a DATATRIEVE application, you must create a request library definition and build a request library file that names the *form definition(s).* In a regular TDMS application, request library definitions contain only requests and, optionally, the file specification to which the request library file is to be written. In a TDMS/DATATRIEVE application, however, the request library definition contains form definitions using the FORM IS instruction; the FILE IS instruction can also be used.

For example, in order to use the form definitions PAYROLL_FORM and SALARY_FORM in a DATATRIEVE application, you create a request library definition in the RDU environment as follows:

```
RDU> CREATE LIBRARY DTR_LIBRARY
RDUDFN> FORM IS PAYROLL_FORM;
RDUDFN> FORM IS SALARY_FORM;
RDUDFN> FILE IS "FORMLIB";
RDUDFN> END DEFINITION;
RDU>
```

You then build a request library file (assuming that PAYROLL_FORM and SALARY_FORM are valid form definitions) using the BUILD command in RDU:

```
RDU> BUILD LIBRARY DTR_LIBRARY
```

In this instance, the information from the form definitions PAYROLL_FORM and SALARY_FORM is built into the request library file named FORMLIB.RLB (.RLB is the default file type for request library files). VAX DATATRIEVE can now access either of the form definitions for use in a DATATRIEVE application.

If any Help forms are identified in any of the form definitions, the Help forms are automatically built into the request library file when the form is built, as long as the Help forms exist in the CDD location that has been specified. You should not include the Help forms in the request library definition.

You can include both requests and form definitions in a request library definition, using the REQUEST IS instruction for requests and the FORM IS instruction for forms. However, remember that:

- In a DATATRIEVE application, you can use a TDMS form definition if it is specified either:
  - In a FORM IS instruction in a request library definition
  - In a request, which is listed in a REQUEST IS instruction in a request library definition

  After a request library file is built from the request library definition, the form can be associated with a DATATRIEVE domain as a form in a form library. For example:

  ```
  DTR> DEFINE DOMAIN PAYROLL USING PAYROLL_RECORD ON PAYROLL.DAT-
  CON> FORM IS PAYROLL_FORM IN FORMLIB.RLB;
  ```

- In a regular TDMS application, you can use a form definition only if it is identified and used in a request. The request that identifies and uses the form must then be listed in a request library definition.

9.2 Modifying a TDMS Form Used by DATATRIEVE

A TDMS form must be modified before it can be used by DATATRIEVE:

- If the form is moved from one CDD location to another by an FDU COPY FORM command, a DMU COPY command, or a DMU BACKUP command. (The form does not need to be changed. Simply enter the form editor and save the form.)
- If a TDMS form is created from an FMS form file, and the FMS form has a different name than the TDMS form.

If a TDMS form is not modified after either of these two operations, TDMS displays the following message when DATATRIEVE tries to use the form:

%TSSFDV-E-FRM, invalid form description

9.3 Converting VAX FMS Forms for Use with TDMS and DATATRIEVE

This section shows how to convert existing VAX FMS forms to TDMS forms for use in either a DATATRIEVE application or a TDMS application.

When VAX DATATRIEVE is installed on your system, it can be linked with either (but not both) the VAX FMS form driver or the VAX TDMS run-time shareable image. If DATATRIEVE is linked with the VAX TDMS run-time image, then you cannot use existing FMS forms in your DATATRIEVE applications.

To use existing FMS forms, convert them to TDMS forms using one of two methods:

1. A command procedure provided by the VAX TDMS software installation procedure that converts FMS form libraries to TDMS library files
2. A series of FMS and TDMS commands that extracts forms from the FMS form library and places them in the CDD

Both procedures are described here. You must have the appropriate FMS V1 or V2 utilities on your system to run the procedures.

9.3.1 Using a Command Procedure to Convert FMS Form Libraries

TDMS provides a command procedure that:

1. Extracts the binary FMS form file from an FMS form library.
2. Places the FMS form file in the CDD as a TDMS form definition.
3. Builds the form definition into a request library file that you can refer to in your DATATRIEVE domain definitions. This step is optional.

To execute this command procedure:

1. Make sure that the FMS form is in an FMS form library. See the VAX FMS documentation for information about creating a form library.
2. Type the following command at DCL level:

```
@SYS$COMMON:[SYSEXE]:FLBTORLB
```

The command procedure prompts you for the following five parameters:

1. The FMS V1 or V2 form library name. Type a standard VMS file specification:

   Device:[Directory]filename.FLB

2. Whether or not the form library is a V2 library. Type Y or N.
3. A CDD path name for the new TDMS form and request library definition. You can supply the entire path name starting with the topmost CDD directory (CDD$TOP), or you can enter a relative path name or given name. TDMS converts the name you enter to a full path name using the default CDD$DEFAULT path name.
4. Whether you are creating a new form in the CDD or replacing an existing TDMS form. Type C or R. The default is R.
5. Whether you want to create a request library file that contains only the form you are converting. Type Y or N. The default is Y. If you type N, the command procedure creates the new TDMS form in the CDD and then stops executing. It does not create a request library file. If you type Y, TDMS creates a request library file in your default VMS directory containing the form.

The request library is stored in your CDD$DEFAULT directory and takes its name from the form library name. For example, if the form library name is DTRFORMS.FLB, the request library is stored as DTRFORMS in your CDD$DEFAULT directory, and the request library file is named DTRFORMS.RLB.

Note that the following features, which may be included on an FMS form, are removed when you convert an FMS form and store it in the CDD:

- Named data.
- User action routines.
- Supervisor-only attributes.
- Character-set information.
- Indexed attributes of fields on VAX FMS V2 forms. TDMS stores the field as a nonindexed field. (Indexed fields on VAX FMS V1 forms are supported.)

You can submit the command procedure in batch mode by using the following DCL command:

```
$ SUBMIT/QUEUE=QUEUE-NAME/PARAM=(P1,P2,P3,P4,P5).
```

The parameters are those identified in the preceding paragraph.

9.3.2 Converting an FMS Form to a TDMS Form Using Specific Commands

To convert FMS forms to TDMS forms yourself, rather than using the command procedure provided by the installation procedure, follow these steps:

1. Extract the FMS forms from the form library.

   - For V1.0 FMS forms, enter the FUT forms utility and specify the form library file specification from which you want to extract the FMS V1.0 form:

     ```
     $ FUT :== $FUT <RET>
     $ FUT
     FUT> FORM-LIBRARY-FILE-SPEC/FF
     DBA1:[DIR]form-library-file-spec Form Name? FMSFORM
     FUT> EXIT
     ```

Note that you do not specify an output form file. Rather, the form utility displays the full form library file specification and prompts you for the name of the form that you want to extract. The name you specify is the name you gave the form when you created it using the FMS form editor, (FED). The form utility uses this form name as an output file name, adding the file type .FRM. FUT stores the extracted binary FMS form in a file called FMSFORM.FRM.

- For V2.0 FMS forms enter the following DCL command:

```
$ FMS/LIBRARY/EXTRACT form-library-file-spec-
$_/FORM_NAME=fmsform/OUTPUT=Fmsfrmfile.FRM <RET>
```

  The parameter fmsform represents the name you gave the form when you created it in FED. Fmsfrmfile.FRM is the name of the new binary form file you create. The .FRM file type is optional.

2. Invoke the TDMS Forms Definition Utility and place the form file in the CDD using the TDMS CREATE FORM command. Note that you specify a new name for the form using the CDD path name. The binary form file is input to the CREATE FORM command with the /FORM_FILE qualifier. If the form file contains a V1.0 FMS form, use the /V1 qualifier as indicated:

```
$ FDU :== $FDU
$ FDU
FDU> CREATE FORM -                               <RET>
FDU>_ CDD-pathname/FORM_FILE=Fmsform.frm/V1      <RET>

FDU> CREATE FORM -                               <RET>
FDU>_  CDD-pathname/FORM_FILE=Fmsfrmfil.frm      <RET>
```

   Note that when you store an FMS form in the CDD, the name of the form is the CDD path name. When you reference this form in a DATATRIEVE domain definition, you use the final name in the CDD path name, the given name. When you assign a CDD path name to a form in the CREATE FORM command, be sure that the last name you enter is unique from any other definition in that CDD directory. Any form name information that was part of the FMS form is ignored.

   After the form is stored in the CDD, you can modify the form definition as you would any other TDMS form definition.

3. Place the form in a request library file using the steps discussed in section one of this chapter.

Note

Your FMS library may contain Help forms that you are also converting to TDMS forms. After they are stored in the CDD, they must be associated with the main forms for which they provide help. You must explicity make this association, whether you convert forms using the command procedure or individual commands. To associate a Help form with the appropriate main form, modify the main form with the TDMS form editor. Enter the Form phase in FDU and type the full CDD path name of the Help form following the appropriate prompt. FDU stores the main form in the CDD with the pointer to the Help form that it will use at run time.

Then enter RDU and build a new library file. RDU extracts both the main form and its associated Help form and places them in the library file.

Index

In this index, a page number followed by a "t" indicates a table reference. A page number followed by an "f" indicates a figure reference.

How to Order Additional Documentation

If you live in:	Call:	or Write:
New Hampshire, Alaska	603-884-6660	Digital Equipment Corp. P.O. Box CS2008 Nashua, NH 03061-2698
Continental USA, Puerto Rico, Hawaii	1-800-258-1710	Same as above.
Canada (Ottawa-Hull)	613-234-7726	Digital Equipment Corp. 940 Belfast Road Ottawa, Ontario K1G 4C2 Attn: P&SG Business Manager or approved distributor
Canada (British Columbia)	1-800-267-6146	Same as above.
Canada (All other)	112-800-267-6146	Same as above.
All other areas	—	Digital Equipment Corp. Peripherals & Supplies Centers P&SG Business Manager c/o DIGITAL's local subsidiary

Note: Place prepaid orders from Puerto Rico with the local DIGITAL subsidiary (phone 809-754-7575).

Place internal orders with the Software Distribution Center, Digital Drive, Westminster, MA 01473-0471.

VAX TDMS
Forms Manual
AA-GS13B-TE

Reader's Comments

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement. ______________________________

____

____

____

Did you find errors in this manual? If so, specify the error and the page number.

____

____

____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) ______________________________

Name ______________________________ Date ______________________

Organization ______________________________________

Street ____

City ______________________________ State ________ Zip Code or Country ____________

Do Not Tear - Fold Here and Tape

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO 33 MAYNARD MASS

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN: DISG Documentation
ZK02-2/N53
Digital Equipment Corporation
110 Spit Brook Road
Nashua, NH 03062-2698

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line