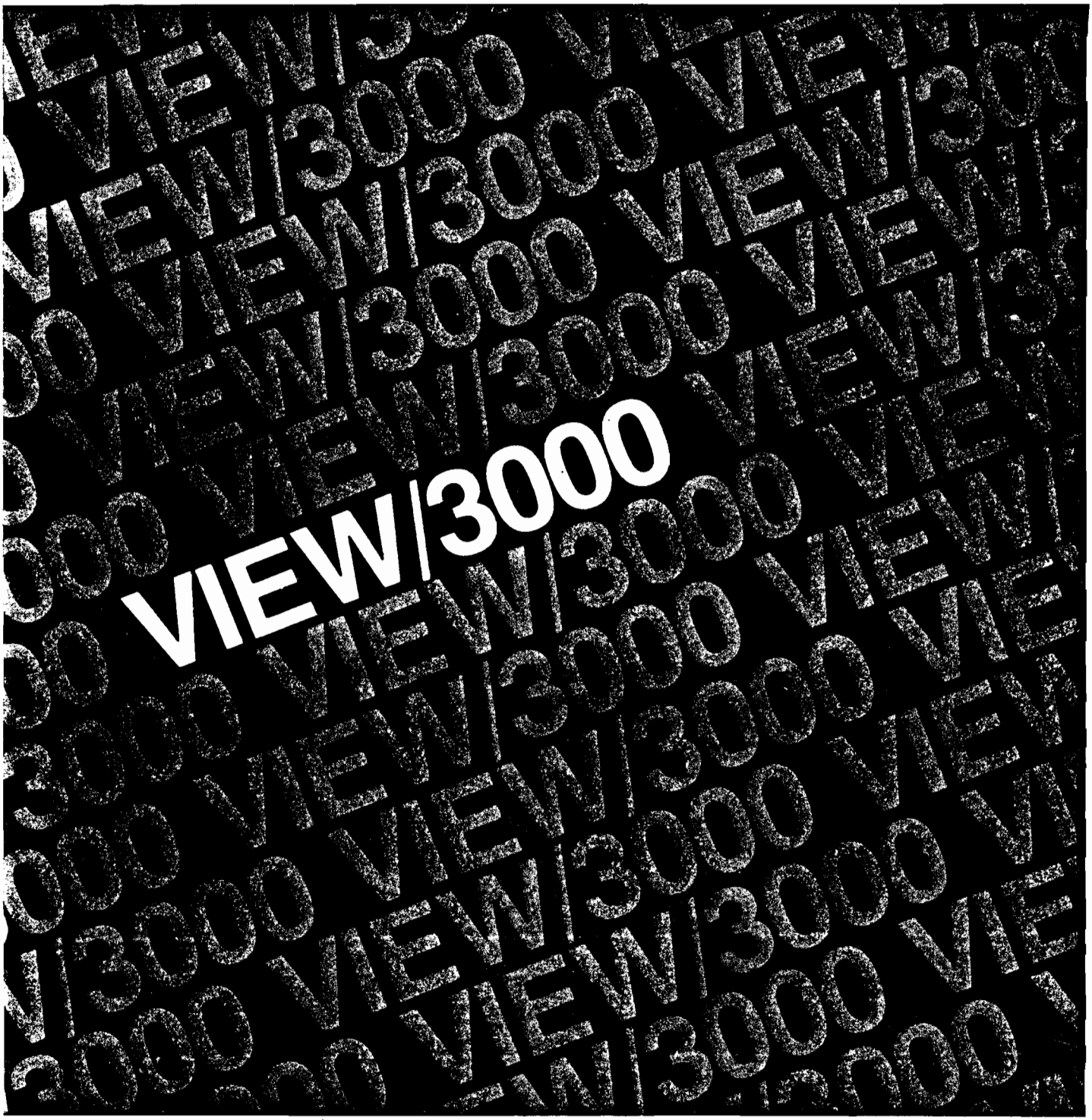
VIEW/3000

HEWLETT hp PACKARD

HP 3000 Transaction Processing Tools

Sales training manual

HP CONFIDENTIAL



VIEW/3000 SALES TRAINING MANUAL

HP-PRIVATE

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

PREFACE

This field training manual is substantially different from the May 1978 version. Changes affect every section and warrant your rereading the manual in detail. But again, if time is short, Chapters I and II provide a quick overview.

CONTENTS

Chapter I

Overview

A QUICK PREVIEW

*** WHAT IS VIEW/3000?

VIEW/3000 is a new software product designed for the HP3000 computer and the HP 264X series terminals. This product provides two major capabilities:

- A stand alone Data Entry Product, VIEW can get your customer started entering data quickly and successfully with NO PROGRAMMING IN A FORMAL PROGRAMMING LANGUAGE!

- A front-end to transaction processing application programs, VIEW provides an extensive library of terminal-handling routines called "procedures". These procedures are available to user programs written in RPG, COBOL, BASIC, FORTRAN and SPL.

In general, VIEW/3000 makes the difficulty of application design proportional to the complexity of the task. Simple applications are super easy to implement. Difficult applications require more effort but are significantly easier to implement with VIEW than with any previous application development tool. This is done through fill-in-the-blanks "menus" and a "parametric" non-programming approach throughout the subsystem.

*** WHAT ARE THE MAJOR FEATURES OF VIEW/3000?

1. A FORMS DESIGN FACILITY utilizing most HP 264X terminals, allows the creation of interactive screens from "fill-in-the-blanks" menus and the use of function keys. Simple edits are accepted by standard defaults and comprehensive data editing and validation can be specified by the use of a free-form field definition language.

2. A SOURCE DATA ENTRY FACILITY that allows immediate on-line entry and modification of data through forms created with the Forms Design Utility.

3. A DATA REFORMATTING FACILITY to change the format of entered data to meet the input requirements of existing application programs.

4. A PROGRAM INTERFACE which aids efficient and easy implementation of forms oriented interfaces for transaction processing applications. This library of high-level procedures is available to provide a simple programmatic interface between an application program on the HP 3000 computer system,the terminal,the forms and edits created by the forms design facility, the entered data, and the data file.

*** WHERE DO I SELL VIEW/3000?

VIEW is aimed at the online Data Entry environment for Transaction Processing applications. VIEW does not address the keypunch replacement market.

A. EXISTING HP3000 CUSTOMERS---
Existing customers may wish to add VIEW to increase programmer productivity, especially in source data entry environments. Sophisticated HP 3000 customers will also like the flexibility of being able to use the "procedures" to write transaction processing programs to extend the capabilities provided by the VIEW product (for example, to meet special editing or Data Base requirements).

B. POTENTIAL HP3000 CUSTOMERS---
VIEW may be the one thing you need to close a sale. An often asked question is "Does the HP 3000 have a complete package that addresses on-line data entry?" The answer now is an emphatic yes. The VIEW package can give your customer a fast and successful start in source data entry without writing a single program!

Chapter VI covers the target markets and anticipated competition for VIEW/3000 in detail but one thing is certain. Old, new and potential customers alike will be highly impressed by VIEW--just have them read the VIEW data sheet or show them how easy it is to use.

*** HOW DO I DEMONSTRATE VIEW/3000?

After September 15, VIEW software is available at all field demonstration centers. Most field S.E's know how to give VIEW demos. Work with your S.E. staff for customer demos.

*** WHEN IS VIEW/3000 AVAILABLE?

VIEW is orderable as of August 1, 1978.
We plan to begin VIEW deliveries on November 1, 1978.
The VIEW product number is 32209A.

Price is $1500.00 (US)
Monthly Software Fee is $75.00 (US)
Monthly Software Subscription is $25.00 (US)
Total Prepaid Purchase Price is $1350.00 (US)

Chapter II

Source data entry/transaction processing

SOURCE DATA ENTRY/TRANSACTION PROCESSING

The current trend toward decentralization of the data entry function increases the importance of the interface between the casual user and the data processing system. Source Data Entry systems allow their users to collect, edit and manipulate data at its source location and, since inaccurate data can cause costly re-entering and reprocessing, place special emphasis on timely and error-free data capture. Source data entry users may be non-technical clerical types but they are generally familiar with the data and how it is used.

Source Data Entry Systems attempt to make it easy for the user to:

- Enter data--often guided by "menus" and formatted screens.
- Resolve data errors--with the help of detailed diagnostic messages.
- Modify data--using "browse" techniques.
- Generate screen forms and incorporate error checking procedures in their own applications.

In contrast, one should note that keypunch replacement systems are generally used by professional operators entering data from forms filled in by others. Keypunch replacement systems often provide keystroke statistics, keypunch keyboards, and "zero" response times. VIEW/3000 is NOT a keypunch replacement.

Source data entry is an integral component of "On-line Transaction Processing Systems". However, Transaction Processing goes further than data entry as part of the decision making process of the business itself. Transaction Processing systems attempt to make it easy for the user to:

- Perform Source Data Entry at locations where the data is actually generated and used for business decision-making.
- Make complex inquiries against the information stored in the computer's data bases and files.

SOURCE DATA ENTRY/TRANSACTION PROCESSING

- Generate timely and accurate reports to dispersed locations in the firm in response to only one or a few input transactions.
- Interact with the computer in a non-technical, conversational way.

Chapter III

What is VIEW/3000?

WHAT IS VIEW/3000?

VIEW/3000 provides a self-contained, "stand-alone" source data entry capability which does not require the designer to write programs. It also permits programmatic access (through procedures) to a comprehensive library of terminal and forms management and data validation routines.

The package provides capabilities in four areas:

- EASY FORMS DESIGN AND EDIT SPECIFICATION

  Traditionally, data entry applications have been designed using a high-level programming language such as COBOL- with CRT screen formats laboriously coded into the program. Data editing was accomplished by building edit routines from scratch or by using routines stored in a program library. This was not an easy process!

  Now, with VIEW/3000, screen formats are typed or drawn on the CRT and additional editing specifications entered via "menus" and a special language. Forms may be "appended" to previously displayed forms and the sequence of forms may be altered dynamically. The program that is used to design forms and specify edits is called "FORMSPEC".

- BASIC DATA ENTRY

  For users wanting data entry and editing only, VIEW/3000 includes a source data entry program called "ENTRY". This program controls the flow of forms to a terminal and executes edits on the entered data before writing it to a data file. Designers need not write one line of code to begin entering data quickly and successfully.

  The ENTRY program is ready for production work and may be used to enter data into a batch file, to review or "browse" through the file and to make changes to the data in the file. Tested examples of this program are listed in the reference manual in five languages and can serve as good examples for programmrs who want to write their own special versions of ENTRY (for specialized edits, data base access or whatever).

- DATA REFORMATTING

  The reformatting utility is a stand-alone program which reads a specified data file, reformats the contents of the

data file according to a reformatting definition contained in a "reformatting specs" file, and writes the reformatted data to a sequential output file. This utility may be required to meet the input specifications of already existing processing systems without hindering the design of new interactive terminal formats. One program, called "REFORMAT", does the actual reformatting.

- PROGRAMMING PROCEDURES

A library of high-level procedures has been defined to facilitate the development of on-line transaction processing data entry applications in general. The procedures provide a simple programmatic interface between the HP 3000 computer, the forms and edits created by FORMSPEC, and a 264X series terminal.

A few of the functions which are callable from BASIC, COBOL, FORTRAN, RPG and SPL are:

GETNEXTFORM	- Retrieves the screen image and all editing characteristics in a single access.
SHOWFORM	- Displays the current form, any data in the data buffer and diagnostic error messages on the terminal.
READFIELDS	- Reads input from the terminal.
FIELDEDITS	- Edits all fields according to forms file specifications.
GETFIELD	- Returns the value of a single field to to the program.
PUTBUFFER	- Writes data from the application program to the data buffer.

FIGURE 1 presents a graphic overview of the VIEW/3000 package. It should be noted that, in addition to the stand-alone programs, VIEW provides an interface (the procedures) between a terminal (screens, edits, and data I/O) and ALL of the capabilities of the HP 3000 computer system.

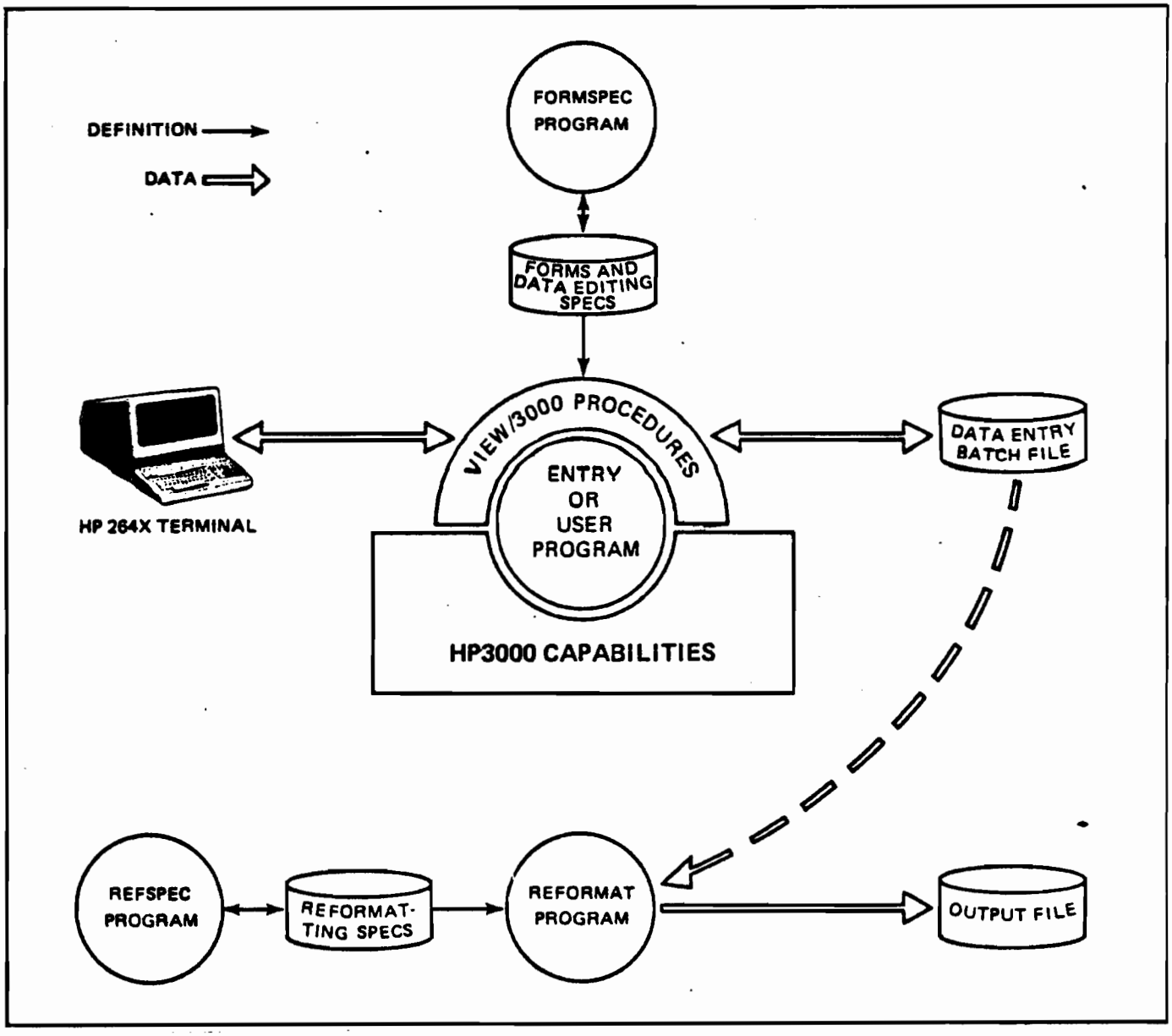


Figure 1. Overview of the VIEW/3000 Package.

WHAT IS VIEW/3000?

HOW DOES IT WORK?

The VIEW/3000 Data Sheet shows, step-by-step, how easy it is to use VIEW/3000 to create a simple PURCHASE ORDER application. Basically, the FORMSPEC program issues a series of menu screens on which the designer enters specifications to define the application. To display the first of these menus, you simply log onto the HP 3000 and enter the RUN command ":RUN FORMSPEC.PUB.SYS".

STEP 1: FORMS DESIGN

The first design step is to create the forms that are to be displayed on the terminal screen. It is important to remember that VIEW treats the data entered on each screen as a separate record. Thus, if the source document is to be represented on the terminal by two forms, two output records will be produced when the application is run. This should present no problem, however, since the REFORMAT program can reformat these records into a single (or many) record(s) as desired.

The visual part of each form is created by typing it onto the terminal screen. Each data field can be demarked by brackets and identified to the system by typing its name within the field; [fieldname]. Figure 2 shows the example PURCHASE ORDER form just before the designer presses the ENTER key to store it into the forms file.

```
      * * *  ENTER A NEW PURCHASE ORDER  * * *

                                          DATE  [date          ]


SHIP TO  [name                                 ]
         [addr1                                ]
         [addr2                          ][zip ]



     ORDER #          QTY.          PART #          PRICE
   [ordernum  ]     [qty ]       [partnum    ]    [price  ]  _
```

Figure 2: CREATING A FORM

STEP 2: DATA EDITING AND VALIDATION

It is important to note that, by accepting default editing specifications for the fields, the forms file need only be compiled before the operator may begin entering data with the ENTRY program.

However, after each form is stored into the forms file, the FORMSPEC program requests editing specifications for ech field in the order they appear on the screen. The designer may qualify fields as to whether they are "display only", "required", or "optional". Input data can also be limited to "character", various "numeric" types or a "date". Initial field values and additional editing, formatting, and conditional logic can be specified with the free-form language described in Chapter VII of this manual.

Figure 3 shows how a "PRICE" might be qualified as a numeric, required field that "must be greater than or equal to 1". Customized error messages may be used where desired by enclosing the message within quotes as shown. Such error messages are displayed in a "window" on the terminal screen at run time. The location of this window is under the control of the designer.

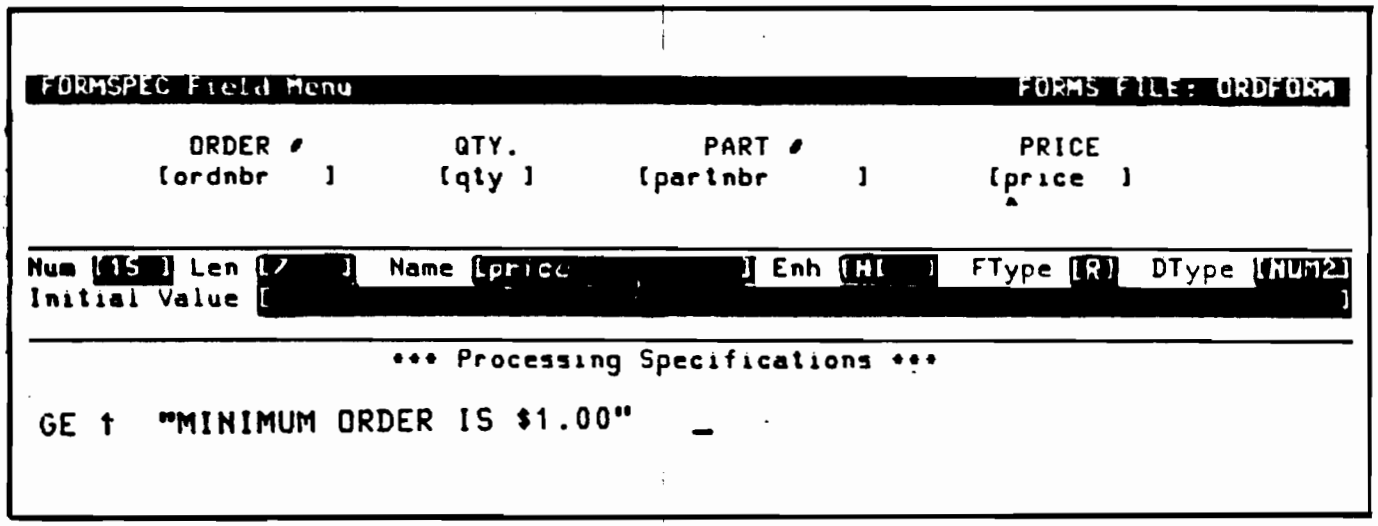


FIGURE 3: SPECIFYING EDITS

After defining one or more new forms (or modifying some old ones), the "source" version of the forms file must be compiled before it can be used in an application. Both the source and the compiled versions of the forms are kept in the forms file.

STEP 3: ENTERING DATA

The VIEW data entry program "ENTRY" can now be used to call the screens from the forms file and enter records into a specified data file. If errors are detected after the data is entered, the program enhances the field(s) in error and displays a diagnostic message on the terminal screen. The operator can then correct the errors and continue.

The ENTRY program also makes it possible for the operator to review or "browse" through the data entered on any screen and, if desired, change the data. A number of operator functions have been assigned to the special function keys of the HP 264X terminal: "display NEXT form", "display PREVIOUS form", and "EXIT" are a few.

STEP 4 - REFORMATTING THE BATCH FILE

Sometimes it is necessary to reformat the data entered to meet the input requirements of already existing application programs. Some of the reasons reformatting may be necessary include the need to:

- Combine data from several forms into a single record on the output file.
- Split data from a single form into two or more records on the output file.
- Rearrange the data within a record, insert constants, and generate check digits before writing it to the output file.
- Adjust data within fields (for example, justify or zero fill).

The program REFSPEC is used to specify how the data file is to be reformatted. Any number of data files may be reformatted and combined (one after the other) into one output file. This output file can then be used as input to the application program. The specifications are entered on menus much like those used by FORMSPEC and are stored in a "reformat definition" file.

The program called REFORMAT can be run any time and actually reformats the entered data. It runs non-interactively and requires only the names of the reformat definition file, data file and output file. Figure 4 presents an overview of the reformatting process:

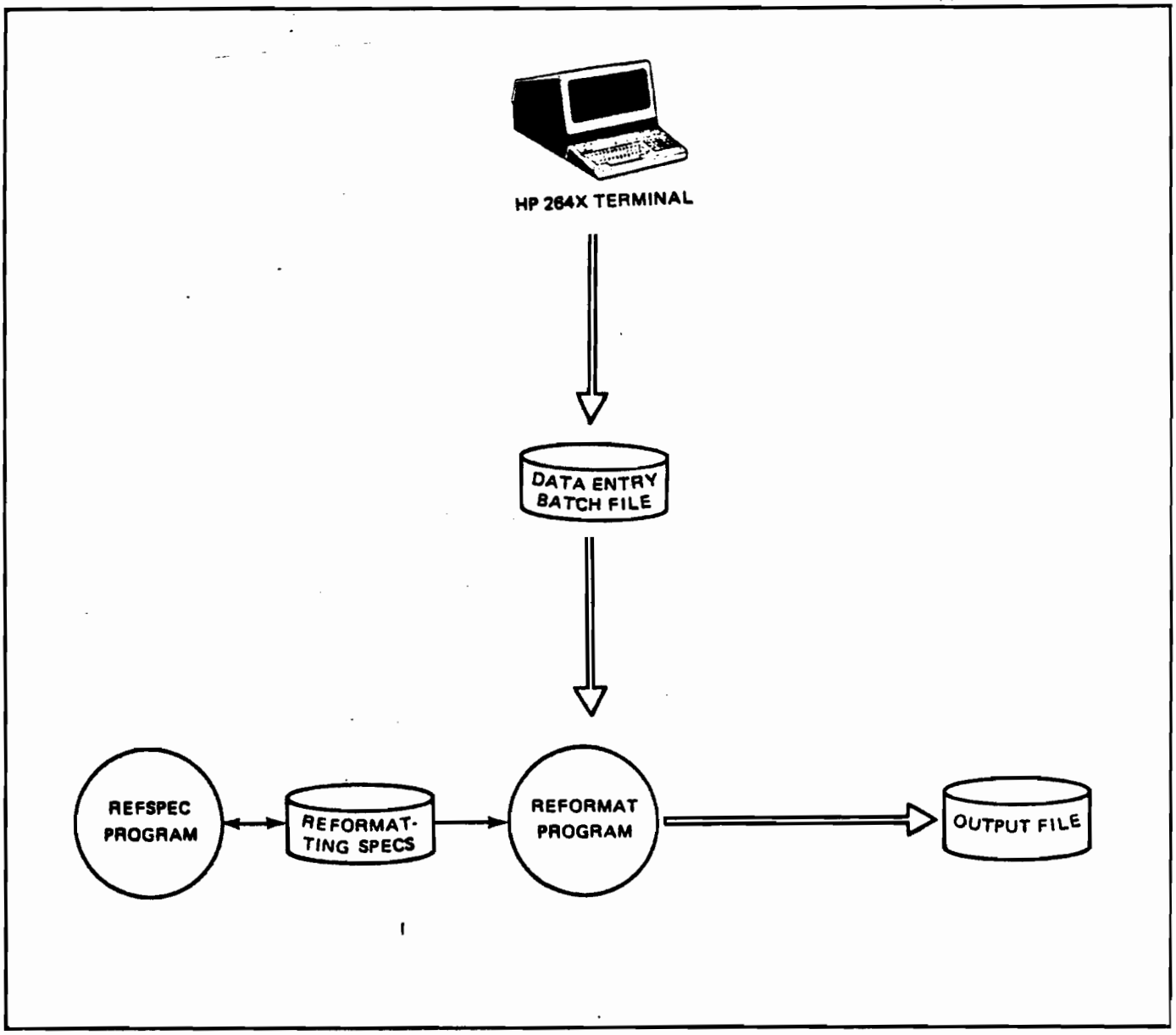


Figure 4: REFORMATTING

PROCEDURES

Given a more sophisticated requirement such as data base inquiry or update, it is possible for the programmer to write his own data entry program in the language of his choice.

The following examples show the format of calls to the VIEW procedures from each language:

LANGUAGE	INTRINSIC CALL FORMAT
COBOL	CALL "procedure name" USING parameter [, parameter2]...
FORTRAN	CALL procedure name (parameter1[,parameter2]...)
BASIC	label CALL procedure name (parameter1 [,parameter2]...)
SPL	procedure name(parameter1[parameter2]...);

where: procedure name - identifies the procedure being called.

parameter - At least one parameter is required for each procedure; the particular parameters are the same for each of the languages.

label - for BASIC calls only, a statement label must precede each call.

NOTE: RPG programs have their own interface with terminals and forms and do not call the procedures directly.

VIEW/3000 makes it possible to do simple applications easily and quickly, yet there is plenty of power (through the procedures) to create sophisticated applications. A VIEW user posesses access to all of the resources on the HP 3000 by writing his own version of the ENTRY program in any of five languages.

SUMMARY OF VIEW PROCEDURES

INTRINSIC	FUNCTION
VCLOSEBATCH	Closes batch file.
VCLOSEFORMF	Closes forms file.
VCLOSETERM	Closes terminal file.
VERRMSG	Returns message associated with VIEW error code.
VFIELDEDITS	Edits field data and performs other field processing .
VFINISHFORM	Performs final processing specified for form.
VGETBUFFER	Reads contents of data buffer into user program.
VGETFIELD	Reads field from data buffer into user program.
VGETTYPE	Reads field from data buffer to user program, converting data to specified type.
VGETNEXTFORM	Reads next form into form definition area of memory; window and buffer areas are not affected.
VINITFORM	Sets data buffer to initial values for form.
VOPENBATCH	Opens batch file for processing.
VOPENFORMF	Opens forms file for processing.
VOPENTERM	Opens terminal file for processing.
VPRINTFORM	Prints current form and data on offline list device.
VPUTBUFFER	Writes data from user program to data buffer.
VPUTFIELD	Writes data from user program to field in data buffer.
VPUTTYPE	Writes data of specified type from user program to data buffer, converting data to ASCII.
VPUTWINDOW	Writes message from user program to window area in memory for later display.
VREADBATCH	Reads record from batch file into data buffer.
VREADFIELDS	Reads input from terminal into data buffer.

VSETERROR	Sets error flag for data field in error; and supplies error message for that field.
VSHOWFORM	Updates terminal screen, merging the current form, any data in buffer, and any message in window.
VWRITEBATCH	Writes data from data buffer to batch file.

Chapter IV

Features, advantages & benefits

FEATURES, ADVANTAGES & BENEFITS

FEATURE	ADVANTAGE	BENEFIT
NO DETAILED KNOWLEDGE OF TRADITIONAL PROGRAMMING LANGUAGES REQUIRED.	Little training is required for the designer or user.	Increases programmer productivity.
EASY FORMS DESIGN	Forms are "painted" onto the terminal using full HP264X editing and video.	Quick acceptance by operators since screens can look much like the same old source document.
FORMS SEQUENCING	Screens sequence may be "chained" or programmatically controlled.	"Guided" entry speeds things up. Depending on the data entered, unnecessary screens need not be displayed.
VIEW Data Entry Program (ENTRY)	Stand-alone & ready to run!	User can enter data immediately.
COMPREHENSIVE EDITING	Provides "clean" data to the system	Reduces costs and time for correction & reentry.
-Length check -Single value comparison -Table check -Range check -Arithmetic expressions -Field initialization -Data movement -Pattern match -Conditionals		Errors caught at the time & place of entry
-Save fields -Check digits (mod10/11) -Custom error messages		See Chapter VII for more details on this subject!!!
REFORMATTING (REFSPEC/REFORMAT)	User can match output to existing record layouts.	No change needed to existing processing systems.
Program PROCEDURES	Callable from BASIC	Allows user to con-

FEATURES,ADVANTAGES & BENEFITS

	COBOL, FORTRAN, RPG and SPL.	centrate on his real (processing) needs.
"MULTI-DROP" Terminal Operation	(optional)	Lower communications costs since fewer phone lines required.

Chapter V

Performance

PERFORMANCE

The information presented in this section is intended as a general guide for evaluating the transaction handling capacity of VIEW/3000. The primary objective was to compare the performance of VIEW/3000 against that of DEL/3000. Comprehensive performance test results will be distributed in the form of a Performance Flyer as soon as the tests are complete. In the mean time, use good "HP judgement" in setting customer expectations.

SYSTEM ENVIRONMENT:

All tests were run on an HP 3000 Series III computer at General Systems Division. This system included 512K bytes of memory, two HP 7920 disc drives and 28 HP 2645A terminals. The terminals were run in point-to-point page mode at 2400 baud.

APPLICATION ENVIRONMENT:

To compare VIEW/3000 against DEL/3000, it was necessary to construct two functionally equivalent COBOL programs. One program called VIEW procedures to perform a screen management task; the second program called DEL procedures to do the same task. Calls to SPL routines were embedded in both application programs to provide appropriate timing measurements.

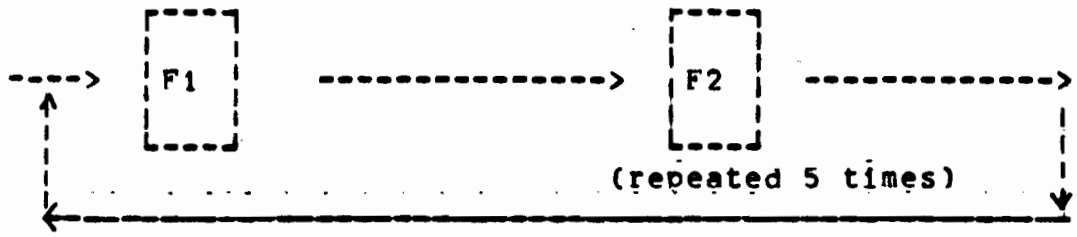
Both application programs performed the following functional steps:

1. RETRIEVE FORM F1 from a forms file
2. DISPLAY FORMF1 on terminal screen
3. INITIALIZE DATA
4. READ DATA from screen
5. PERFORM EDITING if specified
6. WRITE DATA to file

7. RETRIEVE FORM F2 from forms file
8. DISPLAY FORM F2 on terminal screen
9. (RE)INITIALIZE DATA
10. READ DATA from screen
11. PERFORM EDITING if specified
12. WRITE DATA to file

13. REPEAT STEPS 9 THROUGH 12 FIVE TIMES (REPEAT FORM) AND THEN GO TO STEP 1 TO START OVER.

These steps are shown graphically in the figure below.



In fact, two types of forms were used for testing. A "simple" form consisted of only one ten-character field and no editing. A "complex" form consisted of fifty ten-character fields with each field being edited for numeric only.

RESULTS:

With the HP 3000 "dedicated" to the above application, testing with 5, 15 and 25 terminals showed:

*** In terms of response time and transaction throughput, VIEW and DEL are basically comparable.

*** VIEW has a definite performance edge for those types of edits provided by the DEL procedures and for applications using repeating forms.

VIEW's advantage when it comes to simple editing is due to the fact that DEL's editing procedures access the disc once for each individual edit. In contrast, VIEW retrieves all associated edit specifications at the same time it retrieves the form.

VIEW's advantage when it comes to repeating forms is due to the fact that VIEW automatically recognizes fields that have not changed and thus does not rewrite them. DEL does not have a repeat-form capability built into its procedures. This function can be coded into the user's application but the cost of implementing a repeating capability as sophisticated and efficient as VIEW's requires considerable effort and must be recoded in each user application.

With the HP 3000 running several other processes (SORT, RPG, and COBOL) concurrently with the above applications, testing confirmed that:

*** In a mixed system environment, VIEW and DEL are basically comparable with respect to response time and transaction throughput.

*** VIEW impacts other user processes in a fashion similar to DFL.

As mentioned above, a complete performance analysis to characterize VIEW/3000 will be published prior to product

Chapter VI

Target markets and competition

TARGET MARKETS AND COMPETITION

The marketplace for data entry hardware and software ranges from keypunch replacement machines, key-to-disc or key-to-tape, dedicated data entry and edit systems with communication facilities, and data entry an editing packages for general purpose mainframes.

KEYPUNCH REPLACEMENT MACHINES tend to have little or no CRT screen formatting capability or the ability to check the data as it is entered in. These machines tend to have "instantaneous" response so that the operator can enter data as fast as possible. Validation of that data is generally accomplished by rekeying the data a second time and comparing that with the data originally entered.

KEY-TO-TAPE OR KEY-TO-DISC MACHINES tend to have some screen formatting and data editing capability built in. Their primary purpose, however, is to prepare data for batch processing on a mainframe. Little or no table look-up or other types of validation are performed. Response time for the operators tends to be close to instantaneous.

DEDICATED DATA ENTRY SYSTEMS have grown in capability over the past few years. These systems generally support up to a dozen operator terminals. The system may display source forms on the terminal screen as created by an in-house appliction programmer. Many of these systems have a special screen formatting language to perform this task. System designers may specify lists of complex data edits including such features as logical operation, table searching to match an entered value against one of the table elements, and in some cases, validating the entered data directly against the contents of files. This data is stored on disc files and later sent to a host mainframe through RJE or HASP communication facilities.

In addition, many of the vendors of dedicated systems have started to include general purpose utilities such as COBOL compilers and additional file system utilities. This gives users the ability to process entered data further before passing it along to a host computer, and in some cases doing data retrieval and update at the local site while passing only summary data to the central site. The operating systems for these machines are often not general purpose enough to serve as time-sharing or multiprogramming systems.

The final classification is that of GENERAL PURPOSE COMPUTER SYSTEMS WITH DATA ENTRY AND EDITING FACILITIES. These facilities are generally used in conjunction with other data processing activities such as data base inquiry and update, batch processing, and on-line program development. Because these

systems are general purpose and tend to service many users performing diverse tasks, the response time to users performing data entry and editing is not instantaneous. The response time depends upon the overall system load, languages used, complexity of the data editing, and the structure of the data files.

VIEW/3000 -- WHERE DOES IT FIT?

VIEW/3000 is designed to address the latter two marketplaces. VIEW provides a screen formatting, data entry and edit facility for customers who wish to upgrade from a dedicated data entry system to a more general purpose computing system. VIEW allows source data entry by individual users on the system, either as the front end to a transaction processing system or as a stand-alone data entry package collecting to batch files.

VIEW is similar in function to many facilities currently used in the small business and supermini marketplace. The table below lists products offered by some of Hewlett-Packard's competitors

VENDOR	*PRODUCT	*PRICE
DEC	DECFORM	$2500(bundled)
	TRAX	(bundled with hardware)
DATA GENERAL	IDEA	$3000
PRIME	FORMS	$5000
INTERDATA	ITRAC	$6500
FOUR-PHASE	VISION	(bundled with hardware)
	DATA IV	(bundled with hardware)
	STARTER	(bundled with hardware)

****
DIGITAL EQUIPMENT CORPORATION -- DECFORM.
****

DECFORM is a screen formatting and data editing facility designed to run primarily on the CTS 300 and CTS 500 series of commercial systems. These systems run a version of the RSTS/E timesharing operating system, and are generally programmed in either Basic or a "Cobol-like" language known as Dibol.

DECFORM is a screen formatting, data editing and stand-alone data entry package. Users may enter data into serial or ISAM files.

In general, users specify screen formats by typing a source screen description of the form into a DEC text editor, and then compiling that description into a Dibol program. This program is then executed just like any other program on the system. The program can call user-created subroutines to perform special processing. These subroutines are written in Dibol.

The designer specifies both field labels (protected information) and unprotected data fields in which to display or enter data. These fields are described in the screen format by using a line and column number. For example, if the user wishes the label 'NAME' written on the first line of the form starting in column five, he would code:

1,5,'NAME'

The designer also specifies unprotected fields in the same manner. When the form program is executed, DECFORM displays dashes for each alphanumeric character in the field, or cross hatches (#) for each decimal digit in a field. The two allowable data types in DECFORM are alphanumeric and decimal (real numbers).

Along with the placement of labels and entry fields, the user also specifies a list of edits to be applied to the field. These edits include:

- Alpha only (A-Z)
- Field equal to a constant
- Check digit (modulo ten or eleven)
- Duplicate a previous field
- Display only field (not for entering data)
- Reformatted field (performed by a user-written subroutine)
- Full value entered (specific number of characters).
- File validation against an ISAM file.
- Increment the value in the field by one.
- Cross field validation (Edit based on the value of other fields).
- Display leading zeros.

- Arithmetic (multiply values by a constant)
- Do not clear (leave the value displayed when the current record is posted to the file).
- Override (Allow the operator to post the record to the file even if the value does not pass the edit specification.)
- Pass control to a subroutine (written by the designer).
- Required field (may not leave the field blank).
- Range check.

- Save and duplicate the value for the next record.
- Verify a value against the values in a table.
- Numeric only.

The absolute capacity of a single data entry job is 999 data fields, 999 text fields (labels). The sum of data fields must be less than or equal to 9998 bytes. Up to 30 screens may be chained together in a single data entry program. Many data entry jobs may be linked together logically by creating a menu program. When the menu program is run, it displays a list of data entry jobs. After the user chooses the appropriate job from the menu, the menu program transfers control to the selected job.

To understand how VIEW compares to DECFORM, remember the following points:

- DECFORM is not designed to use display enhancements such as line drawing sets or inverse video, etc..

- Designers must specify the form by using line and column numbers. VIEW designers "paint" the form on the screen using the 264x keyboard and step-by-step menus.

- Each terminal requires its own copy of the data entry job. This typically uses up from 4 to 6K WORDS of memory for each terminal. The HP 3000 shares code between users.

- The DECFORM Users Guide warns people that is a DECFORM program aborts or crashes while updating a serial file, the file may be lost. This does not hold true for ISAM files.

- If a user attempts to add a record to a serial file and the file is full, DECFORM automatically builds another file and copies the old file into the new larger one. This may take a great deal of time if the file is large. There seems to be no way to abort this process once it has started.

- Although a designer may call user-written subroutines to process entered data, the designer may not use the screen formatting and data entry capabilities of DECFORM programs as a front end to a general transaction processing system. This is a key wekness of DECFORM agains VIEW. VIEW not only runs stand-alone for data entry and edit, but allows designers to use VIEW capabilities from general programs written in either COBOL, RPG, FORTRAN, BASIC or SPL.

- DATA BASE MANAGEMENT (DBMS-11) does not run on the RSTS/E operating system. Therefore, DECFORM cannot be used as a front end to a transaction processing application using a data base. The HP 3000 uses IMAGE, VIEW may be used as the front end to a transaction processing application program which in turn calls the IMAGE subroutine library to access an IMAGE data base.

****
DIGITAL EQUIPMENT CORPORATION -- TRAX
****

TRAX is DEC's newly-announced transaction processing monitor which runs as a task under the RSX-11M operating system. RSX is a real-time operating system similar in nature to the RTE operating system on the HP 1000 family.

TRAX is designed to perform several major tasks. The first task is to design screen formats for DEC's VT 62 intelligent terminals The second task is to control processing flow for transactions entered on the terminal screens.

TRAX reportedly works with DECNET and can pass processing control to another machine or to other processes within the same machine.

Terminal support consists of both asynchronous (point-to-point) and synchronous multidrop terminals.

TRAX reportedly runs on any machine from an 11/34 to an 11/70. Application modules are written in either Cobol or Basic-Plus-2. The idea of TRAX is to supply a tool which allows designers to dedicate a system to specific, well defined transactions. This is in direct conflict to the current DEC commercial environment which uses the RSTS/E operating system in timesharing mode. RSTS/E is a general, friendly, operating system which can be programmed easily with relatively little pre-planning. TRAX, under RSX-11M requires a great deal of planning, is much less friendly, but is built to use Cobol and intelligent terminals in a dedicated environment. Although little is known yet about product specifics, the following points should be kept in mind:

- Terminals are either dedicated to transaction processing or to "support activities". Support activities include such tasks as a program development or running utility programs. All of the terminals on a single terminal controller must be dedicated to one task or the other. Programs must be development. This is rather inflexible compared to the HP 3000 and MPE.

- All interaction with transaction processing terminals takes place through ATL. ATL is designed to build CRT screens on the VT 62 intelligent terminal. The terminals themselves are capable of very limited stand-alone editing within the terminal itself. However, ATL DOES NOT PERFORM ANY EDITING FUNCTIONS. Consequently, ATL is not a replacement for DEC's DECFORM (see above) nor is it a competitor for VIEW/3000.

- In conjunction with the last paragraph, DECFORM is not supported by TRAX. This takes away one of DEC's commercial tools, because users cannot perform stand-alone data entry and edit. Any source data entry applications must be programmed using the ATL language and either Cobol or Basic-Plus-2.

- DEC does support both ISAM and DBMS with TRAX. However, there are several holes in those products. Datatrieve-11 works in conjunction with RMS-11K (ISAM) and is an on-line stand-alone query program used to retrieve and format data from ISAM files. Datatrieve works only on the "support" terminals, not on the transaction processing terminals. Therefore, Datatrieve will not be available on smaller DEC CPUs and limited in use on the larger ones. In addition, DBMS-11 (data base management system) does not currently possess a query facility. This is a major weakness on any DEC system.

In conclusion, TRAX is seemingly a transaction processing tool similar to Interdata's ITRAC. VIEW/3000 is not a total transaction processing system, but is used as the front end to a transaction processing system. The concept of a TRAX system may appeal to large companies willing to pay the price of technological complexity, but designers of such systems must give up some of DEC's better tools to use it.

****
DATA GENERAL -- IDEA
****

IDEA is used to design screen formats for stand-alone data entry and editing jobs. IDEA works in conjunction with DG's INFOS file management system which is similar to HP's KSAM. Like VIEW, designers "paint" the screen format using the terminal keyboard. Once entered, the IFMT program stores the screen in a file for

later use. After the designer builds the screen IFMT carries on a dialog with the designer about each field to specify such things as "value required", display only field, full value needed, etc.

Designers control the additional editing of fields and interaction with INFOS file structures through programs written in a language called IFPL which is "Cobol-like". The instruction set is:

Arithmetic: ADD,SUBTRACT,MULTIPLY,DIVIDE

Move: MOVE,RIGHTJUSTIFY,LEFT JUSTIFY

Comparison: COMPARE,RANGE CHECKING, TABLE LOOK-UP

Branch of Condition: EQUAL,NOT EQUAL,LESS THAN,GREATER THAN, WITHIN-RANGE,OUT-OF-RANGE,FOUND,NOT FOUND

Display Communication: STORE,DISPLAY,MESSAGE

INFOS File Handling: FIND..USING,FIND NEXT,FIND PREVIOUS, FIND BEGINING WITH,FIND NEAREST,FILE-NEW, REFILE,DESTROY,FIND AND HOLD,RELEASE,and INVERT

Forms Linking: LINK USING,PASS,ACCEPT

Unconditional Branching: GO TO,GO TO USING,RETURN,RETURN USING

As competition to VIEW, IDEA posesses the following characteristics:

- Designers "paint" the screen using the terminal keyboard and special characters like VIEW. Lacking, however, is the display enhancements and extra character sets available to owners of the 264x terminal.

- Executing a data entry job requires writing a program in IFPL. DG has basically invented another Cobol which requires a programmer to use. This is poor compared to VIEW since simple data entry jobs are run under the control of ENTRY, which is supplied by HP. No programming effort is required with VIEW.

- IDEA works with INFOS. When used with multiple indexing levels, INFOS is not easy to use. DG warns salespeople that using INFOS requires a great deal of support. This takes away DG's claim that IDEA is for non-experts. VIEW enters data into MPE files. Building these files is handled by VIEW if the user requests it, and is therefore, much easier

to use. IDEA cannot use direct access files.

- IDEA comes with transaction logging, but there are many weaknesses. For one, there is no utility to use the transaction tape to recreate the blown files. In addition, the transaction log is a five-level INFOS file. Every transaction is indexed by terminal number, operator number, etc.. Writing your own recovery program is not necessarily easy.

- With INFOS, the key files are kept in memory, and are totally wiped out in the event of a system crash. The log file created is then totally useless. This invalidates the main data files used in the application. The system is totally unable to survive an operating system crash.

- IDEA does not multiprogram. This means that if a user presses the ENTER key, the system is dedicated until all of the file accessing and calculations for that transaction have finished. With multi-level INFOS files, tracing down hierarchical chains can "bog-down" the system.

- Also, I believe IDEA's field processing is in screen order, so if screen changes, must change program. VIEW maintains independence of fields and screen order.

- Screen formats and edit lists are not available for other programming languages. Thus, IDEA cannot be used as the front end to a transaction processing system as VIEW can.

Although IDEA is touted as a non-programmer's tool, there is programming involved, and the interaction with INFOS does not make it a simple product.

****
PRIME --FORMS
****

The Prime Forms Management System (FORMS) operates under the control of the PRIMOS level III,IV, and V operating systems. FORMS communicates with the ANSI 1974 Cobol, Fortran, IV, RPG II and Macro Assembler languages.

FORMS main purpose is screen formatting and limited data editing. Designers create screen formats for any device used for listing or inputing data such as a CRT or Line Printer. These forms are stored in a file for later use. Like Digital's DECFORM product (described above), data fields and their labels are specified by line and column number. Validation through edit masks is also allowed which gives some simple editing capability.

Once the screen has been described in the Prime text editor, the source description is compiled using the FORMS Description Language Translator. The forms are then linked to specific device drivers using the FORMS Administrative Command Processor.

User programs interface with the formats using standard I/O calls appropriate to the language. Formats can be changed without necessarily changing the application program.

In general, FORMS exhibits several weaknesses compared to VIEW--namely, it cannot run stand-alone for data entry and edit into a batch file, and secondly, the editing features are primitive compared to VIEW. Points to remember are:

- FORMS cannot supply stand-alone data entry without programming -- VIEW can.
- Formats are designed for hard-copy or CRT devices. The designer links the form to a driver for the appropriate device. Designers may use protected fields, blinking, and inverse video. Lacking is a complete line drawing or alternate character set available for VIEW users.
- Besides Cobol-like edit masks, designers may center,left or right justify entered data.
- Designers cannot paint the screen as they can with VIEW. The position of fields and labels on the screen is accomplished through line and column numbers.
- The same form may be used for different device types. However, the designer must link the drivers to the forms using the Administrative Command Processor. VIEW can write forms to a line printer.
- For transaction processing, FORMS can be used in conjunction with Prime's DBMS and ISAM products. As with VIEW, all of this must be controlled through an application program.

****
INTERDATA TRANSACTION CONTROLLER--ITRAC
****

ITRAC is a transaction-oriented software monitor for business application systems. It runs on Interdata's 32-bit minicomputer under control of their real-time operating system.

The system allows screen formats and data editing specifications to be defined separately from Cobol application programs. These programs manipulate data entered on the CRT screen by referencing the name of the data field. Consequently, like VIEW the screen can be changed without reprogramming the application. ITRAC

compares to VIEW in the following ways:

- Designers paint the screen and specify edit specifications by typing the form on the screen of the CRT and then picking specifications from a menu. This is almost exactly the same way users of VIEW do it.

- The Model 1200 terminal is semi-intelligent compared to the HP 264x models. The 1200 may protect fields, do inverse video, half-bright, and blinking. Lacking is an alternate character set to do line drawing.

- Each field in the screen is given a name which is used to reference the field during Cobol program execution. In this way, the fields can be rearranged on the screen without reprogramming the application. This is exactly what VIEW does.

- ITRAC includes a TEST utility which exercises the display and edit checks for a CRT screen. This does not involve any programming effort. If the form is incorrect, the user runs the MODIFY program to modify the form. VIEW can test forms by using the ENTRY program to enter data into an MPE file. If incorrect, VIEW users can change form layouts more easily than ITRAC can.

- Edit errors are indicated by reverse video for the offending data field. Designers do not get a choice of how to display bad data. With VIEW, the designer can choose what kind of display enhancement to use for error indications. Both ITRAC and VIEW provide diagnostic messages on the CRT screen.

- ITRAC includes a job menu facility. When the console operator initiates ITRAC, all of the user terminals display a menu. All the operator has to do is select a job name from the menu. The menus can be nested if desired. VIEW does not have such a facility.

- Interdata's Cobol includes a record locking scheme for their ISAM file access method. This offers multiple users protection against updating the same record. This is of course, the responsibility of the programmer.

- Inters ISAM includes a logging and recovery system which includes user facilities for entering checkpoints on the log tape. This tape can be used to recover ISAM files in the event of a crash. This is currently not available on HP's KSAM (but will be when MPE supports file logging.)

- ITRAC is NOT a stand-alone data entry facility as VIEW is. Although it is powerful, it does require a great deal of programming effort. As a front end to a transaction processing system, it is as good as VIEW. Remember, that VIEW can be used from Cobol,RPG,Basic,Fortran, and SPL. ITRAC can be used from Cobol, Assembler, and Fortran(?)

- Remember also that HP has a Data Base Management System (IMAGE/3000). Interdata has no DBMS, only ISAM. IMAGE and Query are powerful tools that Interdata does not have.

- Interdata's operating system is a real-time task monitor. Terminal communication and progrm development terminals run as separate tasks on the system. MPE is better in that it handles rapidly fluctuating loads by dynamically allocating resources as needed. On the Interdata, users must reconfigure the machine if the current configuration does not meet peak load demand. Interdata does allow different SYSGENS to reside on disc for easy reconfiguration, but changing this requires shutting down the system for awhile.

In conclusion, ITRAC is a well-conceived transaction processing tool. The two major disadvantages are the lack of stand-alone data entry and the relative inflexibility of the operating system relative to MPE and lack of DBMS.

****
FOUR PHASE SYSTEMS
****

Four-Phase Systems offers a line of dedicated source data entry systems which may be used as keypunch replacements (complete with operator key-stroke statistics), source data entry stations (with screen formatting and edit specification) and on-line data entry and retrieval systems with IBM 3270 emulation. Two products are of interest to you--Data IV and Vision. Data IV is a stand-alone keypunch replacement or a stand-alone source data enry package which includes screen formatting and data validation and edit. Vision is a complete transaction processing package which supports concurrent source data entry, on-line inquiry and retrieval, batch communications, and multi-station file processing.

DATA IV

Data IV looks very much like the DEAL programming language on the HP 2026 product. Using Data IV, designers set up screen formats set up edit lists, and define data entry jobs. These jobs are then executed by operators for data entry functions. These functions include data entry, verification (either visually or by rekeying), and supervisory functions.

Editing and data validation includes:

- Alphabetic, general, hexedcimal, integer, left-zero fill, numeric, prompt, and text field validation.
- Field modification including balance accumulation, zero balance accumulation, auto duplication, auto skip, auxiliary skip, auxiliary duplicate, check digit, display only, generate a value from an expression, generate a constant, must fill, must enter, must release, right justify.
- Cross-field checking including equal, not equal, greater than less than.
- Field arithmetic including add, subtract, multiply, and divide.
- Logical operators including and, or, if, then, else

In general, Data IV operates very much like VIEW/3000. It is used entirely for stand-alone source data entry, and is not used as a front end to a Cobol program. (See the discussion of the STARTER product at the end of this chapter.)

VISION

Vision is an entire transaction processing system which runs on a dedicated Four-Phase system. It supplies the following features:

- Source data entry. Designers specify source documents on the terminals for data entry. They also specify data edits. Included in this is data validation against local ISAM files and host IBM data bases through IBM 3270 emulation.
- On-line inquiry and retrieval. Operators type in a key value on a form and Vision retrieves the associated record from an ISAM file. If the data does not exist locally, provision is made for transparent (to the user) access of a host IBM dta base.
- Systems may be linked together for batch communications with automatic answer and disconnect facilities. Data may be down-loaded from an IBM host or passed from system to system.
- On-line report generation. Operators may manipulate serial files for report generation with headings, arithmetic, and manipulation of data with conditional logic. Data may be sorted on-line.

STARTER

Starter is a screen generation package which interfaces with Four-Phase Cobol. It enables CRT format generation using the terminal keyboard and menus (like VIEW). These forms are then manipulated using a Cobol program. Data editing may also be specified at the time the form is generated.

Although Four-Phase looks quite formidable for specific applications, the biggest weakness is that none of the application packages mentioned above run concurrently. Generally, the system must be dedicated to one application or another. The operating systems are disc-based but batch in nature. Although some of the application packages such as Vision do perform timesharing functions, the user is responsible for timesharing in any application that he writes himself without the aid of Four-Phase application packages.

If users wish to do general processing with some source data entry, the HP 3000 supplies screen formatting and data editing with VIEW/3000, along with the power of the MPE operating system.

Chapter VII

More on the forms design language

MORE ON THE FORMS DESIGN LANGUAGE

VIEW/3000 provides a form design language that can be used to do very sophisticated data editing, data movements, data formatting and conditional control. Each of these four areas are summarized in the tables below:

DATA EDITING

EDIT TYPE	COMMAND	MEANING
Length	MINLEN integer	Checks that the entered value has at least the number of non-blank characters specified by integer.
Single Value Checks	{GT, LT, GE, LE, EQ, NE} operand	Checks that entered value has the specified relation to another value (operand) where: GT = Greater Than LT = Less Than GE = Greater Than/Equal To LE = Less Than/Equal To EQ = Equal To NE = Not Equal To
Table Check	{IN, NIN} value[,...]	Checks that entered value is identical to one of (IN), or differs from all (NIN), the values in list, constants or other fields
Range Check	{IN, NIN} lowval:highval [,lowval:highval]	Checks that entered value is within (IN) or is not within (NIN) any of the specified ranges.
Pattern Match	MATCH pattern	Checks that entered value matches a "pattern"
Check Digit	CDIGIT {10, 11}	Modulus 10/11 Check Digit

A FEW EXAMPLES OF DATA EDITING:

```
MINLEN 5            ---The minimum number of characters entered into
                       this field is five (regardless of how big the
                       field is).

GT 12               ---This field must have a value greater than 12.

IN 12,14,16         ---The entered value must be one of the three
                       listed values.

NIN 12:45           ---The entered numeric value must not have any
                       value between 12 and 45 inclusive.

MATCH (A,AB,BCD)ddd  ---"A123" or "AB999" or "BCD562" among
                          others are acceptable matches for this
                          pattern.
```

DATA MOVEMENT:

Data movement falls into two basic categories;setting of fields and movement of data between fields. Each of these categories uses the SET statement. By default, all fields are initialized to all blanks. The following syntax may be used (as an optional field qualifier) to specify a particular value for any field:

```
SET TO source

      where source may be a { field
                            { save field
                            { constant
                            { arithmetic expression
                            { indexed retrieve
```

The following syntax may be used to move data to a field from some other field or save field:

```
SET destination [TO source]

      where destination is either a field or a save field
      and source is listed above.
```

A FEW EXAMPLES OF DATA MOVEMENT:

Assume a date field of the form MDY. The following statement initializes the field to the value JUL 10, 1978:

```
SET TO ! JUL 10,1978!
```

Assume the value resulting from an arithmetic expression is to be moved to a numeric data field AMOUNT. This statement multiplies the field VALUE by 3 and then sets AMOUNT to 6 percent of the result.

```
SET AMOUNT TO 6% (3 * Value)
```

Special fields, called "save fields" can be defined for the entire forms file. These fields are globally, defined and can be used anywhere in the forms file where field references are allowed. Save fields are useful primarily in order to pass values between forms.

DATA FORMATTING:

In some cases it is desirable to format data as it is being collected as opposed to a separate pass with the REFORMAT program. The following table shows four such formatting commands:

COMMAND SYNTAX	DESCRIPTION
STRIP {TRAILING / LEADING / ALL} "character"	Replaces any specified character in the field with a blank
JUSTIFY {LEFT / RIGHT / CENTER}	Shifts the data within the field to the left or right or center of the field boundary
FILL {TRAILING / LEADING} "character"	Replaces all blanks between the field boundaries and the first or last non-blank data with the designated character.
UPSHIFT	Causes all alphabetic characters to be replaced with their uppercase counterparts.

A FEW EXAMPLES OF DATA FORMATTING:

COMMAND	DATA ENTERED	AFTER FORMATTING
STRIP LEADING "0"	[002050]	[2050]

```
JUSTIFY RIGHT          [ SMITH  ]          [   SMITH]

FILL TRAILING "*"      [  250   ]          [  250***]

FORM SEQUENCING:
```

The default form sequencing defined in the Form Menu can be changed using the following commands:

COMMAND SYNTAX		
CHANGE NFORM TO	CLEAR	FIELD
	APPEND	SAVE FIELD
	FREEZEAPPEND	CONSTANT
		INDEX RETRIEVE
CHANGE CFORM TO	NO REPEAT	
	REPEAT	
	REPEAT APPEND	

CONDITIONAL FIELD PROCESSING:

When a command is to be executed only under certain conditions, this can be specified with the IF statement. The general syntax of the IF statement is:

```
IF condition THEN [command]
   command
    .
    .
   command
ELSE [command]
   command
    .
    .
   command
```

A FEW EXAMPLES OF CONDITIONAL FIELD PROCESSING:

IF QUANTITY GT 100 THEN...	If the current value of the field QUANTITY is greater than 100, any commands following THEN are executed.
IF MINLEN 1 THEN	If at least one character was entered in the current field, execute any commands associated with THEN.

Chapter VIII

What you need for VIEW/3000

WHAT YOU NEED FOR VIEW/3000

The following items are required to run VIEW/3000:

- HP 3000 computer running MPE-III Operating System Software
- KSAM subsystem software
- VIEW/3000 subsystem software
- HP 2645A, 2640B, 2641, 2644A, 2648A or equivalent terminals. Both point-to-point and multi-point terminal operation are supported by VIEW/3000.

  While not absolutely required, we recommend lower-case characters, Display Enhancements, and 8K of memory in the terminals.

Chapter IX

How you order VIEW/3000

HOW YOU ORDER VIEW/3000

VIEW/3000 has been orderable by the field as of August 1, 1978. Delivery is scheduled to begin November 1, 1978.

It is NECESSARY for the customer to already have KSAM software or this must also be ordered.

VIEW/3000

VIEW/3000 PRODUCT #	32209A
VIEW/3000 PRICE:	$1500.00 (US)
MONTHLY SOFTWARE FEE (22823A-020)	$75.00
SOFTWARE SUBSCRIPTION COST	$25.00
TOTAL PREPAID PURCHASE PRICE (32209A-002)	$3750.00

KSAM/3000

KSAM/3000 PRODUCT #	32208A
KSAM/3000 PRICE:	$1500.00
MONTHLY SOFTWARE FEE (22823A-0100	$25.00
SOFTWARE SUBSCRIPTION COST	$10.00

Appendix

FREQUENTLY ASKED QUESTIONS

QUESTION 1: WHAT TERMINALS ARE SUPPORTED BY VIEW/3000???

Today, the 2645A, 2640B, 2641, 2644A and 2648 terminals are supported. As we continue to enhance VIEW, additional terminal types will be tested.

QUESTION 2: WHY IS KSAM REQUIRED???

KSAM is used for its directory management capabilities to keep track of fields and forms within the Forms file on disc (using a single "key". These are variable-length records--and instead of writing the same thing again, we chose KSAM.

QUESTION 3: BUT I NEED A "SPECIAL" EDIT???

You can splice the special code into ENTRY or write a completely new ENTRY program.

QUESTION 4: JUST HOW WOULD I DO A DATA BASE VALIDATION???

For example, assume that you wish to verify that an entered partnumber exists in an IMAGE Data Base. The VIEW procedure, "GETFIELD", is used to get the field value entered by the operator. Then the IMAGE procedure, "DBFIND", is used to check if the entered value is a valid partnumber in the data base. If the partnumber does not exist in the data base, the VIEW procedure called "SETERROR" would be used to return an appropriate error message to the operator.

QUESTION 5: WHAT IS AN "INTRINSIC"???

An "procedure" is simply an HP-supplied and HP-supported subroutine to ease the task of interfacing programs to the operating system.

QUESTION 6: WILL VIEW RUN ON SERIES I 3000 SYSTEMS???

No. VIEW/3000 requires MPE/III to run.

QUESTION 7: HOW DOES VIEW COMPARE TO DEL???

A. Capabilities:

1. VIEW/3000 provides easier application design and maintenance.

2. VIEW/3000 provides more extensive editing capabilities.

3. VIEW/3000 has a stand-alone data entry program; no programming effort involved!

4. VIEW/3000 offers a data reformatting capability.

5. VIEW/3000 provides a library of high-level program procedures which are much easier to use.

B. Performance

DEL and VIEW provide comparable performance (within 10%) with VIEW having a slight edge for applications requiring editing and repetitive forms. However, one should not forget that "ease of use" is also a measure of performance. (See Chapter V for performance details.)

QUESTION 8: CAN I CONVERT MY PROGRAMS FROM DEL TO VIEW?

DEL and VIEW are "not" compatible. No conversion program is anticipated. In general, the VIEW procedures are more powerful than those of DEL and a rethinking (and reprogramming) approach should be considered whenever a DEL to VIEW conversion is desired. (Also, see Appendix C for more information.)

QUESTION 9: WHAT IS THE FUTURE OF DEL???

DEL will be obsoleted on January 1,1979 when VIEW becomes available. But, don't forget our five year support commitment for all of our products including DEL.

SCHEDULE OF EVENTS

DATE	ITEM
AUGUST 1	VIEW/3000 APPEARS ON CORPORATE PRICE LIST
SEPTEMBER 15	FIELD TRAINING MANUAL SHIPPED TO FIELD
SEPTEMBER 15	DATA SHEET AVAILABLE AND SHIPPED TO FIELD
SEPTEMBER 15	SEND VIEW DEMO SOFTWARE TO FIELD DEMO CENTERS
SEPTEMBER 15	PRESS RELEASE APPEARS IN TRADE JOURNALS
SEPTEMBER 15	VIEW APPEARS IN SERIES III COMPUTER ADVANCES
OCTOBER 1	CUSTOMER TRAINING COURSE MATERIALS AVAILABLE
OCTOBER 23	FULL S.E. TRAINING BEGINS
NOVEMBER 1	VIEW DELIVERIES BEGIN
NOVEMBER 1	VIEW REFERENCE MANUAL AVAILABLE (FINAL COPY)
NOVEMBER 1	VIEW PERFORMANCE FLYER SHIPPED TO THE FIELD

DEL TO VIEW CONVERSION

Due to the advanced capabilities provided by VIEW/3000, the programmatic and user interfaces for VIEW/3000 are radically different from that of DEL. The two systems are not compatible. Even for the same appliction, the program structure using VIEW/3000 is much different than the one using DEL.

It is anticipated that most existing DEL applications will continue using that interface, leaving new applications for VIEW/3000. However, it is possible to convert the visual form by the following process:

1. Use DEI to display the form of interest on a terminal that has cartridge cassette capability.

2. Dump each form onto cassette.

3. Go into VIEW/3000 (FORMSPEC) and load the form from the cassette onto the terminal screen. Put the terminal into format mode (control f4) and type a "name" for each unprotected field. Then take the terminal out of format mode (control f5) and press ENTER.
   This will store the form into the forms file.

4. After transferring the form(s) to VIEW/3000, it is still necessary to qualify each data field for type, editing, etc. if default are not acceptable.

Since the VIEW/3000 procedures in general "do more" than those for DEL, program conversion from DEL to VIEW/3000 is not recommended. Instead, the user should rethink and then rewrite the programs in the context of the more powerful VIEW/3000 procedures.

SUPPORT STRATEGY SUMMARY

Initial tools to assist the S.E.'s in their task will include demo software at area offices by 8/15/78. Additional S.E. tools will include: preliminary reference manuals, performance data, and comprehensive field demonstration software.

Support requirements for VIEW/3000 will generally consist of S.E. consulting and assistance in getting the customer started. This will include customer (programmer) training and design guidance.

SYSTEM ENGINEER TRAINING - After VIEW/3000 is released for shipment to customers, GSD training will provide the S.E. with sufficient technical knowledge to support VIEW/3000 in both a pre-sale and post-sale environment. This two-day module will train the S.E. to:

1. Demonstrate the use of ENTRY, REFSPEC, REFORMAT.
2. Design a screen with FORMSPEC utilizing the advanced editing features.
3. Utilize VIEW/3000 procedures in a user program.
4. Provide basic guidelines for modification of ENTRY, conversion from DEL and implementation planning.

NOTE: New S.E.'s will receive a general overview of VIEW/3000 in the Level I S.E. Training Course.

CUSTOMER TRAINING - Customer training will be provided by field S.E.'s on-site or at HP facilities.

TRAINING is provided on:

- Using advanced forms design and processing capabilities
- Using VIEW/3000 as a stand-alone data entry facility
- Reformatting of entered data
- Interface between application programs and VIEW callable procedures.

In general, VIEW/3000 will be supported in the same manner as other software subsystems.

DISTRIBUTION - Object code will be distributed on the HP 3000

SUPPORT STRATEGY SUMMARY

MIT tape.

INSTALLATION - The product will be a "store" tape of VIEW/3000 group in field.support. It will include all program files, necessary data files, USL, a guide file, an install file and a test file. Installation may be performed by C.E. or S.E.

SUPPORT - Standard HP 3000 software support will be available.