PRENTICE COMPUTER CENTRE

University of Queensland

The QDATA Data Entry Package

Introductory Guide

Rob Cook

Mark Williams

MNT-4

24-Jan-79

MNT-4
24-Jan-79

The QDATA Data Entry Package

Introductory Guide

Rob Cook and Mark Williams
Prentice Computer Centre
University of Queensland
24th January 1979

This introductory guide describes the QDATA data entry package version 2(133)-2. No responsibility is accepted for any errors contained in this guide.

This manual has been authorized by
the Director of the Computer Centre

Acknowledgement

The Prentice Computer Centre wishes to acknowledge the contribution of Griffith University towards the initial developments for the Data Entry Package.

Table of Contents

CHAPTER 1

CONCEPTS OF DATA ENTRY AND VALIDATION

1.1 WHY USE DATA ENTRY?

An inevitable chore of a modern computer operation is the collection, codification and input of data for our hungry computer programs to digest. Many computerised tasks require extremely large amounts of data to be collected, often at frequent intervals and with a requirement for urgent processing. Naturally, the data that enters the computer store must be 100% accurate compared to the raw data and it should be possible to crosscheck the accuracy and reliability of the raw data too. It seems obvious that there should be some general solution to the problem of coding and entering large volumes of data.

Most of the more attractive solutions that spring to mind are still the preserve of science fiction writers and of researchers into computer technology. For example, it is not possible to enter data verbally, yet, nor can modern computer systems read manuscripts. There are devices in use today that can read typescripts or data organised as marks on paper, but they are usually require a special character font or method of marking that makes them relatively inconvenient for general use. You will have noticed that bank cheques have the cheque number coded in a strange looking way; these are read by a special reader.

In most cases the computer user has to fall back on a conventional method of data entry. Large jobs are typed using a key-punch machine onto punch cards, or into a magnetic tape or disk file using a terminal attached to a small data entry computer. The card, tape or disk file is later mounted on a larger computer for processing.

An advantage of the tape or disk system is that the data entry terminals are connected to a (usually small) computer, and the computer can be programmed to apply sophisticated checks to the data as it is entered, allowing the operator to correct simple faults when they are made, and to notice and report more complex ones.

The sort of check made by the data entry system can also be made by the program that eventually processes the data, but this is wasteful for two reasons. Firstly it is expensive to run a large program over "dirty" data only to have it abort in the initial input phase. Secondly, each program needs to have its own individual subprogram explicitly for performing the sort of checks that can be done just as well by the data entry process. This is not to mention the danger that the program may miss some vital check and process erroneous data. In general, it is best to catch errors in data as close to the source of the errors as possible. Errors created in the data entry process should be corrected there.

1.2 METHODS OF VALIDATION

There are various data checks that can be applied at the entry stage. Some check that individual numbers have been entered correctly, others check a total across all the data entered and still others provide a check on the whole of the data.

1.2.1 Verification

The traditional check on key-punch data entry jobs was verification. Once a card file had been created, the job was repunched on a machine called a verifier. Instead of punching new cards, the verifier checks that the punches in the existing card deck match the verifying punches. Each character punched is checked, but errors pass undetected if the operator makes the same mispunch twice.

Two disadvantages of the verification method are that it takes twice as long as punching alone, and the error rate is still unacceptably high. A more sophisticated approach is to use other checks wherever possible and to verify the uncheckable fields only.

1.2.2 Field Check

More sophisticated key-punching systems can be programmed to check that each individual number, or other data item, punched is in the correct format. For example, a positive integer number can only contain digits, so if the operator keys a comma, the error is detected instantly and can be corrected.

This type of check is effective since it detects and allows correction of a large number of errors as they are made, but it is obviously not sufficient to ensure correct transcription of data. Even if all the characters keyed were digits, they need not have been the correct digits.

1.2.3 Sum Check

Whether numeric data has been entered correctly or not can be checked by calculating the total of a set of data items before and after entry. The person providing the data calculates totals for rows or columns of data and the totals are entered along with the data. While the numbers are being entered the data entry system also calculates the total, and when entry is complete it compares its calculation with the total entered. Any discrepancy indicates a certain error. The entry of large amounts of numeric data can be checked in this way. If totals are kept and checked for each row and for each column, errors can be pinpointed accurately because they produce error signals for both column and row. This is especially useful when the row and column totals are produced automatically as a result of the data collection process.

Unfortunately, if the totals do not tally, it is still not safe to assume that the data is correct because of the possibility of making two mistakes which cancel each other out. By using multiple sum checks this error can be reduced to an aceptable level.

1.2.4 Check Digit Check

Numbers that are produced by computer, and are later entered back into a computer, such as Medibank numbers, can be checked by adding an extra digit to the number. This extra digit, called a check digit, is ccalculated from the other digits in the number when the number is produced. Later the number is entered into a computer, usually identifying a set of other data that has been collected. The check digit is calculated once more from the other digits of the number, and can be used to check that the keying was accurate. If used properly, check digits are almost infallible.

1.3 ORGANISATION OF A DATA ENTRY JOB

There are 5 phases to every data entry job and it pays to give some thought to planning it properly to maximise the chances of recording data on the computer store accurately and with the minimum of complication.

1. define the data to be collected: be careful to include items that may be used later in the analysis but avoid collecting entirely useless or redundant information unless it is for checking data accuracy;

2. decide the form in which each data item is to be entered into the computer: it is important to use a form that allows the data to be checked automatically, as far as possible, and a form that requires the minimum of manual or mental transformation before being keyed. Numeric data is easiest to check automatically, by providing sums or check digits. Alphanumeric data can be checked if a table of possible values for the data can be constructed. Names and addresses can really only be checked by verification;

3. define your data collection method giving some thought to layout that will aid keying accuracy, and the provision of redundant information that can be used for checking the entered data;

4. prepare a formal description of the key-punching job so that the keying can be performed easily and efficiently;

5. if the data entered is not guaranteed correct by the keying process, you must check the data manually or automatically if you are to have confidence in the results.

1.4 WHO IS INVOLVED IN A DATA ENTRY JOB

There may be as many as 4 distinct people involved in a single data entry job; the person who collects the data, the person who prepares the description of the data for entry to the computer, the person who actually performs the data entry, and the person who feeds the data to the processing programs once it has been keyed. Normally, the collector and the person who uses the data once it has been keyed are the same person.

The two important actors for data entry are the operator who keys the data and the supervisor who prepares formal descriptions of data entry jobs and checks that the job has been accomplished before releasing the data to the customer. In large data entry pools there would most likely be one supervisor and a number of typists, but for a small application the operator would also be their own supervisor.

The supervisor accepts the data from the collector and prepares a formal description, called a job definition, that contains all the information that the operator needs to key the data. The operator takes the data and the formal description and keys the data exactly as it is written. After the data has been keyed and corrected as far as possible, there may still be errors because the data as written disagreed with the job description. These errors are resolved by the supervisor, the collector or the user, and then the operator corrects the data already keyed. Afterwards the supervisor checks that the job has been done correctly and delivers the keyed data to the user.

1.5 DATA ENTRY TERMINOLOGY

A data entry job consists of the entry of a quantity of data organised in a way that can be easily understood by a computer program. Normally the data is organised into records. A record consists of one or more data items, and usually very many records of one or more types comprise a job. For example, a record may describe one person's answer to a questionnaire, or just a part of one person's answer if that is more convenient.

In the past, it was handy to treat the data that would fit into one (or one set of) punched cards as a record. But since cards have been largely superseded the concept of a record has become rather more flexible. Now a typical record might consist of the data that can fit onto the screen of a visual display terminal, or that can be digested by the processing program in one bite.

A record is divided into fields. A field is the name given to one data item, be it a number or a person's name. A record consists of one or more fields, and the record type and format can be described by defining its constituent fields.

Suppose that a file of a mailing list for a society is being created. For each member, the file is to contain his name, address, telephone number, and the date to which he has paid his subscription. A suitable breakdown would be to consider one member's data as one record. Each of the name, the address, the telephone number and the date would be a field of that record. Of course, to take account of the variable number of characters in names and addresses some maximum field size would have to be set.

Some data entry jobs are very large, and have to be broken down into smaller, more manageable portions. Others are continuous, new data arriving to be keyed periodically. Each collection of data that arrives to be keyed is called a batch, and each batch is keyed, corrected and released to the processing program separately.

A data entry job is described by a job definition that describes each field in each different type of record exactly, including the checks that should be performed as the data is entered. The description of each type of record is called its record format.

CHAPTER 2

THE QDATA PROGRAM: WHO SHOULD USE IT AND HOW

2.1 THE QDATA PROGRAM: WHO SHOULD USE IT

QDATA is a data entry package that incorporates all of the concepts discussed in the first chapter. It can be used on a general purpose computer to enter and check data via ordinary computer terminals. It is a compromise solution to the data entry problem between the professional key-punching pool and the entry of data via a standard computer editing program. But it enables the data to be checked as it is being entered rather than in the processing program.

Therefore QDATA is suitable for use by those who need to enter a non-trivial amount of data but whose quantity is small enough to be handled by their own staff, or by themselves of course. Larger volumes of data should continue to be given to professional key-punch agencies. QDATA contains all the features found in agency key-punch systems. Since it runs on a general purpose computer, and is used via a normal computer terminal, its operation is not quite as smooth or fast as on special key-punch equipment, but it is more than adequate for all jobs that might be attempted by the collector of the data himself.

2.2 WHAT QDATA CAN DO FOR YOU

QDATA allows the operator to enter data on any normal computer terminal attached to the computer. The operator keys data in response to prompts displayed on the terminal. Checks, defined in the job definition, are applied

1. after each character
2. at the end of each field
3. at the end of each record for all fields within that record
4. at the end of a sequence of records, for all those records.

If errors are detected, they may be corrected immediately or on a subsequent edit of the batch. At the end of an input batch, the operator terminates data entry and the information is filed securely in the computer's store. At some later time, more data may be added, or the data already typed can be altered or verified.

When the data has been fully corrected it is released to the file system for use by processing programs. As it is released, the data can be reedited in certain standard ways defined in the job definition. For example, fields can be moved around within the record, standard character strings added and so on. Virtually any number of batches can be created under one job definition, and any number of jobs can be under way simultaneously.

QDATA is intended to be used by operators almost totally unfamiliar with computer concepts. It is a simple program to use, with extensive prompting, and help available at any stage for those who need it.

2.3 HOW TO APPROACH A DATA ENTRY JOB USING QDATA

There are 5 steps that are essential to use QDATA.

1. the job definition is created
2. data is collected, batched and given to the operator
3. the operator enters batches of data
4. the operator corrects batches of data
5. "clean" batches are released to the processing programs

2.3.1 Creating A Job Definition

There are two considerations in creating a job definition. The processing programs will read the files after they have been entered by QDATA, so that the format of the files produced must be suitable for the processing programs. One record entered through QDATA will appear as one record to be read by the processing programs. Therefore the records described to QDATA via the job definition must match the records expected by the processing programs.

The operator will enter the data via QDATA from instructions and lists given them by the data collectors. Therefore the job definition should ensure that the form of the collected data makes it easy for the operator. Once the format of the collected data and the format expected by the processing programs is known, it is time to write the job definition. The job definition tells QDATA:

1. what different record types will be keyed
2. what sequence those records appear in
3. the size, in characters, of each field in the record
4. what range of characters is permissible in each field
5. if the data is to be right or left justified within the field

6. if empty character positions in the field are to be filled and with what character

7. whether to total numeric fields for checking purposes

8. what prompts to issue to the operator

9. how the records should eventually be written to the file system for consumption by the user programs

10. what special purpose data checks are to be applied.

Job definitions are prepared by a supervisor using the QDATA program itself, using a special standard job definition tailored for creating job definitions. The user's job definitions are kept on his computer file store as ordinary QDATA batches.

Before use by QDATA, the job definition batch is 'assembled' by QDATA and appears again on the file store as a job file ready when an operator starts to enter data using that job.

2.3.2 Collecting And Batching Data

The way that batches of data are collected and assembled for entry to the computer system depends on the person who will eventually process the data. For example, one batch of data will probably be processed as one unit after entry. For data entry purposes it is convenient to keep batches fairly small so that they can be entered and corrected in one sitting, and so that sum checks can be done across cohesive groups of data.

2.3.3 Entering And Correcting Data

The operator enters data in discrete batches, using the job definition. According to the job definition, QDATA prompts for the operator to type in data, and the operator merely follows the prompts. The operator can choose whether to have all the prompts displayed, or to have only minimal prompts. Full prompts are useful to familiarise oneself with a new job definition or when using a high speed display terminal. Minimal prompts are best for high speed data entry. Following initial typing, batches are verified and altered until they are perfect.

2.3.4 Releasing Clean Batches

When the operator has finished entering and correcting the whole batch, the supervisor can instruct QDATA to write the batch into the computer file store as a normal data file ready for processing directly by user programs.

2.4 QDATA IMPLEMENTATION ON THE DECSYSTEM-10 COMPUTER

The rest of this guide describes the current implementation of QDATA on the DECsystem-10. It is numbered version 2, and contains all the basic features, but lacks some of the special data editing and checking methods described above.

CHAPTER 3

USING A TERMINAL TO ENTER AND CORRECT DATA

3.1 BASIC PROCEDURE

The various phases of entering and correcting data using QDATA are:

1. locate a vacant computer terminal and 'log-in'
2. start up the QDATA program
3. enter and correct data
4. stop QDATA
5. 'log-out' from the computer
6. go and have coffee

3.2 LOG-IN

Once you have located a computer terminal, turn the power on and type control-C. Wait for the computer to respond by typing a period. Then type a LOGIN command and answer the questions so that the computer system can check your identity. You will need to know your PPN, a pair of numbers that tell the computer who you are, the password associated with that PPN, so that the computer can verify that you really are who you say you are, and the amount of money that you are prepared to spend on using QDATA during this session.

```
.LOGIN 265/471          !or your PPN
PASSWORD:               !the password does not echo
COST: $10               !or your cost limit
```

In these example computer dialogues, the characters underlined are typed by the computer, while those without underlining are typed by the operator. Text following a '!' is explanation and is not part of the dialogue.

After this short exchange, the computer types back some information including any special messages from the computer centre and then types a period as a prompt for commands from the typist. Alternatively the computer messages may say that the log-in was unsuccessful because you made an error or didn't know the PPN or

password correctly or possibly because the cost specified exceeds what is left in the account. Try again.

3.3 STARTING QDATA

After a successful log-in, you will be faced with a period. To start QDATA, type:

```
.R QDATA
QDATA Data Entry Package version 2(107)-2
*
```

Now QDATA is waiting for instructions about what you would like it to do for you. You give instructions in the form of imperative commands that allow you to enter and correct batches of data, and to control the operation. Entering and correcting data are described below.

3.4 STOPPING QDATA

To leave QDATA, merely type EXIT when faced by an asterisk prompt:

```
*EXIT
.
```

and you can start another program or log-out.

3.5 LOG-OUT

If you have no more work to do, log-out before leaving the terminal. The simplest way is to type:

```
.K/N
```

or

```
.K/F
```

'K' stands for kill-job, and is the standard method of log-out. '/N' does the trick in the fastest possible way, but does not exist on older types of DECsystem-10. '/F' causes KJOB to type end of session messages including the net cost of the whole session.

3.6 INSTRUCTING QDATA

```
.R QDATA
QDATA Data Entry Package version 2(107)-2
*
```

QDATA's asterisk prompt is asking you what you want to do next. You can either tell QDATA that you want to enter or correct data, or you can give QDATA information, such as the number of the next batch to work on. After QDATA has finished the task, or processed the information, it types another asterisk prompt asking for more instructions. This level, ordering QDATA in answer to asterisk prompts, is called command level. You must always return to command level before giving QDATA this type of instruction.

There are many instructions that can be given at command level including EXIT and HELP. HELP will try to give you enough on the spot encouragement to get you to carry on when you are stuck. EXIT leaves QDATA and returns to DECsystem-10 monitor command level, so that you can KJOB or do some other processing.

Suppose that you want to start entering a new batch, to be numbered 37, for the job named MAILST. First tell QDATA about the batch:

```
*BATCH 37
Starting new batch 37
*
```

Now that you want to enter data using job definition MAILST:

```
*ENTER MAILST
Entering batch 37
<<MEMBER; 1>>
NAME:
```

QDATA prompts with the name of the format for the record to be entered, and the number of the new record within the batch, within << >> brackets. Then QDATA prompts for you to enter each field in the record one by one. After each record is completed, QDATA automatically prompts for the next record to be entered. While you are entering (or correcting) data in response to prompts which QDATA finds in the job definition, you are talking to QDATA at input level. You can leave input level and return to command level at any time.

It is important to get the distinction between command level and input level clear in your mind. At command level you give instructions in answer to an asterisk prompt. While at input level you enter data in response to prompts concerning the data in the particular fields and records.

3.7 COMMAND LEVEL INSTRUCTIONS

At command level you can issue instructions of several different kinds. Some will cause QDATA to descend to input level and some will stay at command level awaiting further instruction.

Each instruction comprises a verb followed by some clarification of how the verb is to be applied. The parts of an instruction are called the verb and its arguments. The verbs are predefined and are all english words. However, when you type verb names, you can shorten them as much as you like, provided that your abbreviation remains unambiguous. For clarity, full forms will be used throughout this guide.

A brief explanation of the available instructions is given below. More details are provided in later sections, and a full list in alphabetical order is given in Appendix A.

1. HELP and EXIT

We have already seen the effects of the HELP and EXIT commands. Actually HELP can be quite helpful. If you merely ask for help, HELP will give you a short explanation of how to obtain further help, such as a full list of all the possible instructions, or the details of how to use a particular instruction.

To obtain a list of all the instructions available, type 'HELP COMMANDS'; to find out how to use the ENTER command, try 'HELP ENTER'.

2. Instructions that Inform QDATA

These instructions can be given with an argument telling QDATA what is wanted, or they can be given alone, in which case QDATA will type back the current value.

1. BATCH tells QDATA which batch is to be worked on next. Without a BATCH command, QDATA either uses the last batch mentioned, or asks if none has been specified previously.

2. OPERATOR tells QDATA your name, so that you can be remembered as the last person to tamper with a particular batch if any disaster happens.

3. PROMPT tells QDATA how much prompting you require at input level. If you are a novice, or are dealing with a new job definition, then you may need all the prompts QDATA can give you. If you are already experienced, it is far quicker and more efficient to type the data with the very minimum of prompting.

4. TERMINAL tells QDATA what type of terminal you are using, in case QDATA can make use of special features for data entry.

5. DELIMITER instructs QDATA to use some other character for the field and record terminators than normal. For example, some terminals do not have a line-feed key, and it is more convenient to use a TAB perhaps.

3. Instructions that Descend to Input Level to Enter Data

ENTER creates a new batch, whereas APPEND adds to the end of an existing batch.

4. Instructions that Descend to Input Level to Verify Data

The VERIFY command causes QDATA to process an existing batch of data, asking the typist to retype all fields marked as 'to be verified'. Discrepancies are corrected and reverified.

5. Instructions that Search Existing Batches for Errors

1. TOP closes the batch and reopens it to begin searching from the top.

2. NEXT moves to the next record.

3. RECORD finds a record with a particular record number within the batch.

4. SEARCH finds a record by searching forwards looking for a match with a given string of characters.

5. ERROR finds the next record containing a known error

6. Instructions that Correct without Descending to Input Level

   1. DELETE will delete one record from the batch.

   2. INSERT enters one extra record immediately after the record found by a searching command.

7. Instructions that Enter Input Level To Correct a Record

   The UPDATE instruction enters input level and allows the typist to skip through fields correcting any mistakes.

8. Instructions that Open up an Extra Range of Supervisor Instructions

   The SUPERVISOR instruction makes available special supervisor functions, such as assembling job definitions and releasing batches.

3.8 INPUT LEVEL INSTRUCTIONS

Once you have entered input mode to type data, you continue entering data record by record until finished or you wish to stop for some other reason.

Everything typed at input level is understood by QDATA to be data, therefore special keys have to be used to indicate that you have reached the end of a field, the end of a record, the end of data to enter, or some other condition.

End of field is indicated by pressing the Return (RET or CR) key, and end of record by pressing the Linefeed (LF) key. RET is known as a field terminator and LF as a record terminator. Normally it is not necessary to type a record terminator as well as a field terminator for the last field in a record, but sometimes a particular Job Definition will insist.

Other instructions at input level are composed of the Escape (ESC or ALT) key followed by one or two letters indicating what is to happen. For example, <esc>T is an instruction to terminate input mode and return to command level. <esc>DR will delete the record just typed, and <esc>H will provide help. Detailed descriptions of input mode instructions are given in later sections and a full list is provided in Appendix B.

3.9 CORRECTING ERRORS DURING DATA INPUT

Mistakes that you make during typing can be corrected by striking the Delete (DEL or RUB) key to erase the bad character, and then type the correct character instead. For example if you realise that you struck G instead of H, then the mistyped character can be deleted using Delete, and retyped:

```
NAME: SMITG\G\H
          ^ ^
        DEL H
```

The character rubbed out is echoed preceded by a backslash and when the replacement character is typed, it too is preceded by another backslash so that the erased character stands out.

If the mistake is more complicated it can be erased by several Deletes or by using the Delete Field or Delete Record instructions at input level.

If you make an error that is detected by QDATA, such as an alphabetic character in a numeric field, the program rings a bell on your terminal and waits for you to correct the error. You can either strike the Delete (DEL) key to delete the erroneous character or the 'A' key to accept the error. It is best to accept the error if the data sheet is marked incorrectly and you do not know how to correct it without consulting the author. QDATA marks that record as containing an error and expects it to be corrected later.

Some errors are only detected at the end of a field or record, such as a sum of all the numbers in the record not matching the calculated sum of the numbers typed. If the Delete key is struck in answer to such an error, the whole field or record is deleted. 'A' accepts the error as one to be corrected later.

3.10 ENTERING DATA

Data is entered in batches according to a predefined job description. The job description is named, and the name must be quoted before any data is entered. Thereafter when adding data to an existing batch or correcting errors in an existing batch, it is enough to mention the batch number and QDATA remembers the job to which that batch belongs.

Batches are numbered, and batch numbers are unique across all jobs being entered under one PPN. After starting QDATA the typist should first quote the number of the batch to be entered:

```
*BATCH 37
Starting new batch 37
*
```

or

```
*BATCH 37
Batch 37 opened
    uses job MAILST and contains 103 records
    last appended by W. JOHNSON at 1003 on 25-Nov-78
*
```

depending on whether the batch had been created previously or not.

Once a batch has been opened, data can be entered or corrected until the batch is closed and a new batch opened by another BATCH command.

The ENTER command is used to enter data at the start of a new batch, whereas the APPEND command is used to add data to an existing batch.

3.10.1 Identifying Yourself

It is often helpful to remember the last typist to enter or alter the data in a batch, particularly if there are several people working at entering the same batch (this is not possible simultaneously). The OPERATOR command will remember your name along with the batch data:

```
*OPERATOR W. JOHNSON
*
```

If you type the OPERATOR command alone, it will tell you the name of the current operator, hopefully yourself!

```
*OPERATOR
Current operator is Rob Cook
*
```

QDATA only remembers the first 10 characters of the operator's name, so that D. Whitehouse will be remembered as D. Whiteho.

3.10.2 Starting A New Batch

After specifying the batch number and being told that you are starting a new batch, use the ENTER command to enter input mode, and to specify to which job this batch belongs:

```
.R QDATA
QDATA Data Entry Package version 2(107)-2
*OPERATOR W. JOHNSON
*BATCH 37
Starting new batch 37
=> *ENTER MAILST
Entering batch 37
<<MEMBER: 1>>
NAME:
```

The job MAILST has already been defined by a supervisor. ENTER starts input mode, announces that the first record to be entered is a 'MEMBER' record and that its sequence number is 1. Then it prompts 'NAME:' for the typist to enter the first field in the record.

3.10.3 Adding Data To An Existing Batch

If the batch already exists, instead of ENTERing, type an APPEND command. It is unnecessary to specify the job name since the name is remembered along with the batch:

```
        .R QDATA
        QDATA Data Entry Package version 2(107)-2
        *OPERATOR W. JOHNSON
        *BATCH 37
        Opening batch 37:
            uses job MAILST and contains 103 records,
            last entered by W. JOHNSON at 1003 on 25-Nov-78
   =>   *APPEND
        Appending to batch 37 using job MAILST
        <<MEMBER: 103>>
        Name:    B. Hohnes
        Address: 31 Cleveland Av
        Suburb:  Arana
        P-Code:  4329
        Phone:   4391290
        Date financial: 21-Nov-78
        <<MEMBER: 104>>
        NAME:
```

Again the next record to be entered is a MEMBER record, and because the batch already contains 103 records, the first one to be added is numbered 104. APPEND types out the contents of the last record before asking the typist to begin entering data.

3.10.4 Data Input Level With Prompting

Following the prompt, the operator types the data for that field and strikes the field terminator key (RET) to move onto the next field. As soon as one field is finished, the prompt for the next field appears and so on until the full record is entered.

The layout of the prompt messages is the work of the author of the job definition. The field terminator appears on the screen or printer as a movement of the carriage to begin the next field and the typing of the prompt. This may be simply starting the next prompt on a new line or the author may have arranged several prompts on one line. A record terminator initiates the start of a new record, which always begins on a new line with the normal record prompt.

```
        <<MEMBER: 104>>
        NAME: B. HOHNES          ADDRESS: 31 CLEVELAND AV
                                 SUBURB:  ARANA
                                 P-CODE:  4329
                                 PHONE:   4391290
                        DATE FINANCIAL: 21-NOV-78
        <<MEMBER: 105>>
        NAME:
```

3.10.5 Data Input Level Without Prompting

After an operator has become familiar with a job definition, it can be tedious and slow to be prompted for each identical record. Operators commonly turn off prompting for each individual field and enter the data for each field remembering the order of the fields. Some operators also like to turn off the record format and number prompt at the beginning of each record, although this is a useful record of what was entered.

1. Prompting for each field can be turned off by typing

```
*PROMPT NOFIELD
```

at command level, and can be turned back on again by typing

```
*PROMPT FIELD
```

In 'no-prompt', the operator merely types the data field-by-field separated by field terminators.

```
<<MEMBER; 104>> B. HOHNES:31 CLEVELAND AV:ARANA:4329:4391290::
<<MEMBER; 105>>
```

In 'no-prompt' mode the field terminators appear on the screen or printer as ':' characters separating the fields. A record terminator, or the field terminator for the last field appears as '::'.

2. Prompting for each record can be turned off by

```
*PROMPT NORECORD
```

or

```
*PROMPT NONE                !field and record prompts off
```

and, of course, to turn them back on again:

```
*PROMPT RECORD
```

or

```
*PROMPT ALL
```

With no record prompts, QDATA still issues a brief prompt message to show when it is ready for more data to be entered:

```
>> B. HOHNES:31 CLEVELAND AV:ARANA:4329:4391290::
>>
```

3. To find out what prompting is currently in force:

```
*PROMPT
Prompting for fields and records
*
```

3.10.6 Special Instructions At Input Level

If anything other than data is to be entered at input level, that is special instructions such as 'terminate input mode' or 'blank to the end of this record' or 'help!', the operator must signal a special instruction using Escape (ESC or ALT) followed by characters indicating which special instruction. For example:

```
<<MEMBER: 105>>
NAME: <esc>T                    !return to command level
Terminating input level
*
```

These special instructions can be given at any sensible stage at input level. They are described in following sections and listed in Appendix B.

3.10.7 Error Detection And Correction

Errors may be detected after each character keyed, at the end of each field or at the end of a record. When the typist makes an error, the terminal's bell is rung and QDATA waits for the operator to correct the error. There are 5 possible actions for the typist.

1. type DEL (or BS) to erase the offending item

2. type A to accept the error. QDATA remembers that the record contains an uncorrected error and it must be fixed before the batch can be released to the file system.

   ```
   10.13p<bel>Accepted
   ```

3. type H to obtain an explanation of the error.

   ```
   10.13p<bel>H[Character p not valid in this field]
   10.13p<bel>
   ```

   A further request for help, with another H will prompt an explanation of the actions that the typist can take.

4. type <esc> to enter any reasonable input level instruction, just as during inputting

5. type any other character to ring a <bel>. After 3 false tries, QDATA suggests that the operator might like some help.

3.10.8 What To Do When The Bell Rings

If the mistake was merely an incorrect character in a field, it can be deleted or accepted easily, by typing <del> or A.

If the bell rings on typing a field terminator character, then something is amiss with the entire field. Either the job definition required that the filled be completely filled and the operator has typed too few characters, or QDATA has found that the data typed conflicts with the job definition. Striking the DEL key is equivalent to a Delete Field to erase all characters in the field and start again. It may not be obvious what was wrong with the field so it will often be necessary to use help to find out. 'A' will accept the field, but mark it as containing an unresolved error.

```
Course code: CS01<ret><bel>H[Field must be filled]<del>
Course code: CS301<ret>
```

After the field terminator the operator asked for help and was told that the field keyed contained insufficient characters, so <del> was used to delete the entire field and QDATA prompted for it to be retyped.

If the bell rings after typing a record terminator, there was a mistake in the record which was only detected at the end. For example, the sum of the fields in the record didn't balance with a total entered as the last field. A DEL is equivalent to a Delete Record and the operator must retype the entire record.

```
<<Vital Statistics; 2109>>
Chest: 42
Waist: 52
Hips: 38
Total: 105<ret><bel>H[Sum does not balance with total]<del>
<<Vital Stat; 2109>>
Chest:
```

Alternatively the operator may have typed the record terminator before typing all the fields of the record. In most cases, the job definition will insist that the whole record be entered. In this case, <del> will only erase the record terminator character and allow the operator to complete the record.

3.10.9 Correction Of Errors Recognised By The Typist

Single or multiple character errors within the current field can be corrected using the DEL (and BS) keys. Each time DEL or BS is struck one character is erased.

The whole of the current field can be deleted using Delete Field by typing <esc>DF

```
<<MEMBER; 105>>
Name: W. Johnson<esc>Delete Field
<<MEMBER; 105>>                    !retypes record entered so far
Name: J. Beveridge
```

At any time the whole of the current record can be deleted by using Delete Record, typing <esc>DR.

It is also possible to return to the previous record or field to correct that, but in this case the record or field currently being entered is lost. The instruction Backspace Field (<esc>BF) returns to the start of the previous field,

and Backspace Record (<esc>BR) to the start of the previous record to begin retyping.

```
<<MEMBER: 105>>
Name: W.Johnson<esc>Backspace Record
<<MEMBER: 104>>
Name:      J. Beveridge
Street:    31 Cleveland St
Suburb:    Arana
Postcode:  4239
Phone:     4391208
Date Financial: 31-Dec-49
<<MEMBER: 104>>
Name:
```

The typist can now skip through the record correcting it before continuing to append more records. The method of correcting records is described below.

3.10.10 Copying And Filling Fields And Records

Fields and records can be filled in 3 ways

1. by copying the previous record's contents into the current record in the matching fields

2. by filling with fillers as specified for each field in the job definition. For each field in the job definition it is specified that empty characters in the field should be filled with a particular character. For numeric fields, zero is usually used, whereas for alphabetic fields blank is common.

3. by filling with blanks

The instruction to fill a field or a record can be given at any stage during the input of a record. Sometimes the record format in the job definition will explicitly forbid copying or filling, and in that case an attempt to do so will be greeted by a bell. If the previous record was of a different type, then the record cannot be filled by copying.

To fill by copying, issue a Copy Field (<esc>CF) or Copy Record (<esc>CR) command. For example if record 104 is an entry for Mr J. Beveridge, and record 105 is for Ms K. Beveridge of the same address:

```
<<MEMBER: 105>>
Name:      Ms K.<esc>Copy Record Beveridge:
Street:    31 Cleveland Street:
Suburb:    Arana
Postcode:  4239
Phone:     4392109
Date Financial: 31-Dec-43::
<<MEMBER: 106>>
Name:
```

Similarly, to fill with blanks, use Skip Field (<esc>SF) and Skip Record (<esc>SR) and to fill with fillers, use Null Field (<esc>NF) and Null Record (<esc>NR).

3.10.11 Automatic Copying And Skipping

Sometimes fields are always left blank, for a number of records, or always contain the same string as the previous record. Then it is convenient to have QDATA fill or copy the field automatically without the operator having to press any keys. As soon as the typist reaches that field, QDATA can copy or fill it and proceed to the next field.

Two conditions have to be met before automatic copying and skipping can be used. The Job Definition must explicitly permit either automatic copying, or automatic skipping for the field. And the operator must turn on the automatic copy/skip switch, by typing <esc>A as an input level instruction. For example, suppose a number of members to be entered all live in Arana. Assuming that the suburb and postcode fields were defined so that automatic copy is allowed, the operator can turn on automatic copy after entering the first member living in that suburb. Having entered the first one, our old friend Mr Beveridge:

```
<<MEMBER; 105>>
Name:      <esc>Auto
Name:      Ms K. Hohnes
Address:   28 Acacia Grove
Suburb:    Arana
P-Code:    4239
Phone:     4391208
Date financial: 21-Nov-67
<<MEMBER; 106>>
```

To turn the automatic copy/skip switch off again, type another <esc>A, to reverse the sense.

3.10.12 Using A Different Record Format

The format of the next record can be determined by default or from the job definition or by the operator. If no contrary instruction is pending, the format of the next record will be the same as the current one.

Often the job definition is written so that one record follows another in a specific order. For example, if the total information on a member of a mailing list is split into 3 records for convenience, because there is so much detail, the records will always be input together and in order. So the job definition arranges that the typist is requested to enter the 3 records in the correct order, and then return to the first record format for the next member.

Sometimes the order of the records is only determined by the data sheets given to the typist from which to work. Then the typist has to instruct QDATA when the record format is to be changed.

The first record format defined in the job definition is known as the default format, and is used if no other format has been specified.

The typist can change the record format for one record only, by using the Format Alter (<esc>FA) instruction. After one record of the alternate format, QDATA restores the default record format, but normally the new record format would contain the name for the next record format to be used.

```
<<Vital Statistics; 34>>
Chest: <esc>Format Alter Legs<ret>
<<Legs; 34>>
Length: 77
<<Vital Statistics; 35>>
Chest:
```

The record format can be changed permanently using Format Keep instead of Format Alter. If Format Keep or Format Alter is used without a format name, the format reverts to the default format.

The default format can be changed by using Format Default (<esc>FD) at any time. It is an advantage to have the record format that is most often the target of a Format Alter or a Format Keep as the default, because it is quicker to change formats to the default.

3.10.13 Typing The Current Record

The instruction Record (<esc>R) can be given at any time to type out the entire contents of the current record. If the new record has not been entered yet, <esc>R will type out the previous record.

3.10.14 Finding The Status At Input Level

It can be quite confusing at input level, especially when using no-prompt, to find out the state of the job, whether automatic copy/skip is set or what the current format is, for example. At any time <esc>W, for What, can be typed to obtain a status report.

```
>> <esc>What
QDATA is appending to batch 101 using job MAILST;
     now at record 6, format MEMBER;
     default format is MEMBER;
     prompt = none, auto copy/skip = off.
>>
```

3.11 CORRECTING AN EXISTING BATCH

Batches are corrected by verification, by finding records with errors that were noted during input, or by scanning a listing of the batch, noting errors and correcting them. various commands are available at command level that allow the typist to Verify a batch, or to locate records within a batch and then correct

them. It is also possible to insert a record between two existing ones, and to delete records.

3.12 VERIFICATION

The supervisor specifies which fields of which records are to be verified when the job definition is written. After the operator has entered data into a batch, the batch should be processed again to verify all the verifiable fields. Once verification is complete then theoretically the batch is ready to be released. Until verification is complete, QDATA will ring an alarm if the supervisor tries to release the batch.

The batch can be verified in one pass or the operator make take several bites. Each time verification is started, QDATA will find the first unverified record and begin there.

To verify a batch:

```
    .R QDATA
    QDATA Data Entry Package version 2(107)-2
    *OPERATOR W. JOHNSON
    *BATCH 106
    Batch 106 opened
         uses job MAILST and contains 103 records,
         last appended by W. JOHNSON at 1003 on 25-nov-78
=>  *VERIFY
    Verifying batch 106 using job MAILST
    <<MEMBER: 1>>
    Name:
```

3.12.1 Verifying With Prompts

QDATA types the prompts for all fields, and the values for fields that are not to be verified. For fields that are to be verified, it waits after typing the prompt, for the operator to retype the field.

3.12.2 Verifying Without Prompts

QDATA types a record prompt and data values for non-verify fields if the job definition asks for them to be typed. When QDATA waits, the operator types in the value of the field to verify it.

3.12.3 If The Bell Rings

If QDATA detects a verify error, it may be either the original data or the verify data that is incorrect. If the operator has made a mistake typing the verify data, it can be corrected using <del> or Delete Field. If the mistake is in the original data, the operator should Accept the mistake, and QDATA will

replace the old field contents in the record by the verification data. At the end of verifying the record, if any corrections have been accepted, QDATA will ask the operator to reverify the record.

```
<<MEMBER: 104>>
Name:  K. Hohnes
Street: 31 Cleveland St
Suburb: Arana
P-Code: 432a<bel><del>a9
Phone:  5391208<bel>H[Verification error]Accepted
Date financial: 21-Nov-48
Reverifying <<MEMBER: 104>>
Name:  K. Hohnes
Street: 31 Cleveland St
Suburb: Arana
P-Code: 4329
Phone:  5391208
Date financial: 21-Nov-48
<<MEMBER: 105>>
```

3.13 CORRECTION

A batch is corrected by searching through the batch for errors and then correcting them. The corrected batch has its records renumbered to take account of records inserted or deleted. Therefore it is most convenient to correct a batch in one pass from beginning to end. Every time TOP is used to start again at the top of the batch, the batch will be closed, renumbered if necessary, and reopened.

3.13.1 Finding A Record Containing A Mistake

The commands available to find a particular record are:

ERROR	locate the next record containing an error
NEXT	move onto the next record
RECORD	find the record with a given record number
SEARCH	find a record containing a given string
TOP	go to the beginning of the batch

The usual method of correcting a batch is to start at the top and correct all known errors on one pass through the batch. For example, the typist can use the TOP command to position to the beginning of the batch, and then repeatedly use the ERROR command to find all the records throughout the batch that contain known (to QDATA) unresolved errors. After ERROR has found a record, it is typed, and then the typist can use the UPDATE command to enter input mode to correct that record.

```
*TOP
Starting at top of batch 37
*ERROR
<<Vital Statistics; 48>>
Chest: 44
Waist: 52p
Hips:  38
Total: 105
*
```

Another method is to locate records by number or by context. If the numbers of the faulty records are known, they can be found using the RECORD command.

```
*TOP
Starting at top of batch 37
*RECORD 34
<<Legs; 34>>                    !34 happens to be format Legs
Length: 77
*
```

NEXT can be used to advance through the batch record by record. SEARCH is used to find a record containing a certain string of characters. For example, if you know there is an error in Beveridge's record, but you've forgotten the number, you can request.

```
*SEARCH /Beveridge/
<<MEMBER; 105>>
Name: Mr. J. Beveridge
......
*
```

The two '/'s are delimiters which are used to show where the string to be searched for begins and ends. Any characters may be used instead of the '/' so long as they are not contained in the string.

A SEARCH command written this way will locate the string no matter what field in the record it is in. However, another form of SEARCH allows the typist to be more specific.

```
*SEARCH 7 /Beveridge/
```

tells QDATA to search for a record containing the string Beveridge in the 7th field of a record. This is particularly useful when a file is full of numeric data that may have similar strings in different fields.

Once a search command has been given, QDATA will search for the next occurence of the same string if given a SEARCH command without any explicit string.

3.13.2 Correcting A Record

Once a record has been found it can be corrected by entering Input level using the UPDATE command. UPDATE allows the typist to skip through the fields within one record, correcting any faulty fields by retyping them. For each field the typist has the opportunity to type:

1. a field-terminator, to leave the contents of the field unchanged
2. corrected data to fill the field
3. <esc>SF to delete the previous contents of the field and leave the field full of spaces
4. <esc>NF to leave the field full of fillers

So, suppose we have to correct a Vital Stat record with the waist measurement entered as 52 instead of 25:

```
*SEARCH /52/
<<Vital Statisitics: 38>>
Chest: 44
Waist: 52
Hips:  38
Total:105
*UPDATE
Chest: <ret>44
Waist: 25                    !alter faulty data
Hips: <ret>38
Total: <ret>105
*
```

And to correct a Member record, that had the house number entered as 31 instead of 33, and had a phone number entered by mistake:

```
*RECORD 105
<<MEMBER: 105>>
Name: Mr. J. Beveridge
Street: 31 Cleveland Road    !should have been 33
Suburb: Arana
Postcode:4329
Phone: 4392109               !should have been blank
Date Financial: 30-Nov-49
*UPDATE
Name: <ret>Mr. J. Beveridge
=> Street: 33 Cleveland Road
Suburb: <ret>Arana
Postcode: <ret>4329
=> Phone: <esc>Skip Field
Date Financial: <ret>30-Nov-49
*
```

3.13.3 Typing A Record

Once a record has been found or updated it can be typed out by giving a RECORD command at command level without specifying the record number.

3.13.4 Deleting A Record

Once a record has been found by searching it can be deleted by entering UPDATE and using <esc>DR, for Delete Record. Instead there is a shorthand method; the DELETE instruction at command level will act the same way and delete the record. The next record becomes current.

```
*ERROR
<<Legs; 34>>
Length: 77
*DELETE
Deleted record 34
*
```

3.13.5 Inserting A Record

It is sometimes necessary to insert one (or more) records into the middle of a batch, for example if you have just deleted the wrong record by accident. Using the INSERT command, 1 record can be inserted after the record that has just been found by searching. The inserted record becomes the current record. If you need to insert several records, you must give several insert commands.

```
*SEARCH /Beveridge/
<<MEMBER; 105>>
Name:    Mr. J. Beveridge
Address: 31 Cleveland Road
Suburb:  Arana
P-Code:   4239
Phone:   4391208
Date financial: 30-Nov-49
*INSERT
<MEMBER; 106>>
Name:      Ms. K. Beveridge
Address: <esc>Copy Record
etc
```

CHAPTER 4

THE SUPERVISOR'S ROLE

4.1 INTRODUCTION

It is anticipated that the supervisor will be responsible for the data collection and processing, and will be rather more familiar with the data and with the operation of the computer than will the typist. The supervisor provides the data in a form that can be typed from directly, opens an account at the Computer Centre, obtaining a PPN and a password as well as sufficient space in the computer's file store; creates a job definition describing the format of the data to QDATA and releases completed batches from QDATA to the system file store for processing.

4.2 CREATING A JOB DEFINITION

The supervisor uses QDATA itself to build a job definition of the data entry job. The job definition is created initially as a normal QDATA batch. Then it is assembled by QDATA to make it usable as a job definition. The standard job definition, JOBDEF, is used to create job definitions. JOBDEF is described in detail in the next chapter.

4.3 ASSEMBLING A JOB DEFINITION

Once the job definition batch has been entered and corrected, it must be assembled, to change it into a form that can be understood quickly and easily by QDATA. To assemble a job definition, or to perform any other specialised supervisor's function, the supervisor must tell QDATA that they have supervisor status. Then the command to assemble a batch can be given.

```
.R QDATA
QDATA Data Entry Package version 2(107)-2
=> *SUPERVISOR
*BATCH 29
Opened batch 29
      uses job JOBDEF and contains 5 records
      last updated by J. SIDEBOTHAM at 2158 on 30-Nov-78
=> *ASSEMBLE GIRLFR                          !girlfriends
Assembling batch 29 to create job GIRLFR
*EXIT
.
```

Note that, due to limitations in the file system on the DECsystem-10 computer, job definition names are restricted to 6 characters.

4.4 WHERE CAN JOB DEFINITION FILES BE STORED

Job definitions are normally stored in files on the operator's disk area so that they can be used easily by merely quoting the job name. If the supervisor and the operator are different people with different PPNs, the supervisor may find it convenient to keep job definition files on his own area and have the operator use them from there. This can be done if the operator declares that the supervisor's PPN is their library PPN. If QDATA cannot find a job definition on the operator's own PPN, it will look on the library. The operator can declare a library PPN when they log-in:

```
.LOG 262,153/LIB:[262,121]
```

Standard job definition files, such as JOBDEF, are held on the TED: system library of standard text editing files. If QDATA fails to find a job definition on the operator's area or on their library, it will look on TED:. A list of standard job definitions can be found in Appendix C.

4.5 WHEN IS A BATCH COMPLETED?

A batch is completed and ready for output when all the data has been entered, when all the entered data has been corrected, so that no records are marked as containing errors, and when all fields marked as to be verified have been verified in all records. In fact, the data should also have been checked visually, using the PRINT instruction, to see whether there are any obvious errors that have not been detected in any automatic way.

4.5.1 Re-Verification

It is possible to clear the flags marking records as having been successfully verified if a second check is required for any reason. Maybe the batch was verified and then modified heavily. The supervisor can use the CLEARVERIFY command to clear all verify flags.

```
*SUPERVISOR
*CLEARVERIFY
Batch 29 must be verified again
*
```

4.5.2 Printing The Contents Of A Batch

The supervisor can print the whole of a batch onto the lineprinter complete with all prompts, so that the batch can be visually checked for errors.

```
*SUPERVISOR
*PRINT
Printing batch 29
*
```

For each record, PRINT gives the record format and record number followed by all the fields of the record keyed by the operator, separated by colons. If the supervisor wants to see the record as it would be output, with transfers, fills and constants included, and without the field separators, it is best to use 'OUTPUT LPT:' instead of 'PRINT'.

4.6 RELEASING BATCHES FOR PROCESSING

Once a batch has been completed it must be released to the file system before any future processing. The supervisor asks QDATA to output the batch and nominates a filename for the released batch.

```
*SUPERVISOR
*OUTPUT GIRLS1.DAT
Batch 29, 96 records, using job GIRLFR written as GIRLS1.DAT
*
```

In the output file each record appears as one line terminated by a carriage return/line feed. The length of the record is as specified in the job definition, and each field is written in the columns specified with no intervening inter-field marks.

If the batch has not been completely verified before the OUTPUT command, QDATA will object and ask if the supervisor wants to change his mind.

4.7 ERASING BATCHES AND JOBS

The OUTPUT command does not delete a batch from the QDATA system. Both batches and jobs must be deleted explicitly using the REMOVE command.

```
*SUPERVISOR
*REMOVE 29
Batch 29 removed from QDATA
*
```

QDATA will not allow removal of a batch that has not been output without asking the supervisor if that is what is really wanted.

Jobs are removed by quoting the job name instead of the batch number. QDATA will always query the removal of a job if it still contains batches entered under that job.

```
*SUPERVISOR
*REMOVE GIRLFR
Job GIRLFR removed from QDATA
*
```

or

```
*SUPERVISOR
*REMOVE GIRLFR
** DANGER ** Removal may affect batches 1, 2, 6, 10
Continue? No
Job GIRLFR retained
*
```

If there are batches remaining in the file store that were entered under the job definition that the supervisor has asked to be removed, those batches will never be able to be completed, without reentering the job definition. Although QDATA does not prevent removal, it does give the supervisor a second chance to save the job definition.

4.8 FILES USED BY QDATA

While processing batches, QDATA keeps the data that has been entered in a file, along with information about the batch such as the job name used. These files appear on the user's disk area with names of the form Bnnnnn.QDB. For example, batch 103 would be kept in a file called B00103.QDB.

Job definitions that have already been assembled are kept in files named with the name of the job. For example, a job called GIRLFR when it was assembled, would be found in a file called GIRLFR.QDJ.

These batch and job files are written in a format suitable for QDATA and are not easily readable using any other DECsystem-10 program.

The data entry process is managed by QDATA in such a way that the files should only be deleted by using the REMOVE command to QDATA. If they are deleted using the ordinary DECsystem-10 DELETE command, to delete a file, batches may be left stranded without a job definition, or batches may be deleted before they have been fully corrected and released for processing.

After a batch has been released by QDATA, it appears on the file system as an ordinary data file with whatever name was specified by the supervisor in his OUTPUT command.

While a batch is being altered by appending or correcting, a copy of the unaltered file is kept in case any disaster should happen during alteration. The previous copy of the batch has a name of the form Bnnnnn.BAK, matching Bnnnnn.QDB. To go back to the previous copy of the file in case of emergency, remove the batch file, then return to monitor mode and rename Bnnnnn.BAK to Bnnnnn.QDB.

```
*REMOVE 103
Batch 103 removed from QDATA
*EXIT
.RENAME B00103.QDB=B00103.BAK
Files renamed:
B00103.BAK
.
```

All QDATA's .QDB and .QDJ files are given a file protection that prevents them from being deleted accidently using a monitor mode DELETE command, outside QDATA. Normally the only way they can be deleted is by a QDATA REMOVE instruction.

CHAPTER 5

JOB DEFINITIONS

5.1 INTRODUCTION

The job definition describes to QDATA each different type of record that can be keyed as part of a job, and each different field within each record. So the job definition consists of a series of record definitions or formats, and for each record definition there is a series of field definitions. A record definition contains such information as the name of the format, the length of the record, the number of fields and what record is to be entered after this one. An individual field definition tells QDATA what type of information will be keyed, what checks to make and so on.

Job definition batches are created using the standard job definition, JOBDEF.

```
.R QDATA
QDATA Data Entry Package version 2(107)-2
*OPERATOR J. SIDEBOTHAM
*BATCH 121
Starting new batch 121
=> *ENTER JOBDEF
<<Record: 1>>
Format name:
```

Job definitions can be entered, appended and corrected just like any other batch. After they are completed, they are assembled as described in chapter 4.

Job definitions are not simple to write, or to write correctly, until you have had some practice and are thoroughly conversant with the job definition's way of describing jobs. It is essential to give each new job definition an exhaustive test before letting an operator enter data using it.

5.2 THE RECORD DEFINITION

The following items must be specified for each record defined:

1. name

2. record length

3. number of fields in the record

4. whether the record is to be included in the data ultimately released or whether it is merely for checking other records

5. if an explicit record terminator is needed to end the record just entered, or whether the field terminator for the last field is sufficient

6. whether any special record checks are to be performed at the end of record (none are available in the current version of QDATA)

7. what record definition should be used for the next record

8. field definitions for each field in the record

5.2.1 Record Name

The name of the record can be up to 10 characters long. It appears in the prompt message requesting the typist to enter or correct a record, and it is used in the record definition to tell QDATA which record is to be entered after the current record.

5.2.2 Record Length

The record length is the length of the record output to the file when the batch is ultimately released for further processing.

It is <u>not</u> the sum of the lengths of the fields entered for the record. The two lengths may be different because fields may be moved around in the record between entering and output, may be lengthened or shortened, and constant fields can be added before output. Also some input fields may be used solely for checking other input fields and will never be output.

5.2.3 Number Of Fields In The Record

This is the total number of field definitions following the record definition, and includes fields entered as well as fields indicating changes to the input data record to be made when the record is released.

5.2.4 Whether The Record Is To Be Output

Some records are never output because they only exist to act as a check on other the records. For example, a series of records may be entered containing sums of money. Afterwards another type of record may contain the total of those amounts and QDATA can check that the total was correct. The record containing the total is not output because its only reason for being entered was to check the correctness of the sums entered, and this function has been completed inside

QDATA.

Data or check:

This field is coded:

D if the record is data to be sent to the output file when the batch is released (default)
C if the record is comment or check information only;

5.2.5 Mandatory Record Terminator

If the operator types sufficient fields and their terminators to make up a record, there is really no need to type a record terminator too. The last field terminator is quite enough.

Explicit terminator:

Answer:

I if the last field terminator is to be taken implicitly as the record terminator (default)
E if an explicit record terminator is required

5.2.6 Special Checks At End Of Record

None of these special checks have been implemented in this version of QDATA.

5.2.7 Selecting The Next Record Format

The name of the record format for the record to be entered following the current record can be specified. If it is given, as soon as the record is finished, a record of the specified format will be requested.

The operator may want to enter a record of that type, or may want to switch to another type depending on the data being entered. As well as giving the name of the next format, the record definition can insist that the given format is used, or allow the operator to override and choose the type using Format Alter, Format Keep or Format Default.

Next record format: Vital stat
Force next format:

Answer:

F the format given in the record definition is compulsory
N the format may be selected by the operator

5.2.8 Example - The Vital Statistics Record

```
<<Record: 1>>
Format name: Vital Stat        !only 10 characters
Record length: 6               !total for checking only
Number of fields: 4            !chest,waist,hips,total
Data or check: D               !the record is data
Explicit terminator: I         !implicit is o.k.
Next record format: Vital Stat
Force next format: F           !only one record type
<<Field: 2>>
```

5.2.9 Example - The Mailing List Member

Suppose there are 6 fields, whose lengths are:

Name of member	35
Street	35
Suburb	35
Postcode	4
Phone	10
Date financial	9

making 128 character positions altogether.

```
<<Record: 1>>
Format name: Member
Record length: 128
Number of fields: 6
Data or check: D
Explicit terminator: I
Next record format: Member
Force next format: F
<<Field: 2>>
```

5.3 THE FIELD DEFINITION

The following items must be specified for each field defined.

1. the type of the field, whether it is an entered field or a field that manipulates the data before output
2. where the field is to appear in the output record
3. what characters are permitted within the field
4. how the remaining characters that are not entered are to be filled
5. whether to left or right justify the entered data within the field

6. whether verification is required for the field

7. whether copying and skipping can be enabled

8. the prompting message asking the typist to enter the field

5.4 FIELD TYPES

There are 5 distinct field types. 2 specify how data should be entered for that field, and the other 3 specify how data should appear in the final output record.

1. I - Input field

   Input fields specify how data is to be entered for that field and how it is to appear in QDATA's internal copy of the record. Because of the transformations that QDATA can make on the record as it is output, as directed by the other types of field, the internal record may not be the same as the output record.

2. V - Verify field

   A verify field is essentially identical to an input field except that it must be verified before it can be output. An input field is not verified, but relies on some other form of checking.

3. T - Transfer field

   Transfer fields specify the contents of a given field in the output record. If no transfer fields are used, the output record will be the same as the internal record, but transfer fields can be used to move data around within the record before output. The transfer field specifies the source and destination for a field of characters.

   Transfer fields allow a field to be entered once, but to occur in several places in the output record. They can also be used to split up input fields to appear in different parts of the output record. For example, it is convenient to enter a date as day-month-year, but the programs that process the record may require the year at the beginning and the day near the end of the record.

4. F - Fill field

   Fill fields specify that some portion of the output record is to be completely filled with the selected character. For example, the processing program may expect as input a series of 6 digit numbers each separated by 5 spaces. Rather than enter 5 spaces after each number, it is quicker to code fill fields for the gaps, and enter the numbers separated only by field terminators. Actually, just one fill field of 5 spaces could be coded, and transfer fields used to generate the others.

5. C - Constant fields

Constant fields are like fill fields, but instead of filling with one character, a constant field fills with a given string of characters. For example, the processing program may require that a record containing a total appears as 'TOTAL=' followed by the number. It is much quicker to enter the number alone and to add the string 'TOTAL=' using a constant field. Another application is when all the records in a job, or batch, need some identifying string attached to them, for example the origin of the batch of data. The job definition can be set up with a constant field containing the origin name and it need never be keyed again although it will appear in each output record.

The field type must be coded as I, V, T, F or C. Any other code will provoke a protest from the job assembler in QDATA.

Field type: C

5.4.1 Field Position Within The Record

QDATA needs to know the first and last character position of the field within the record so that it can position the data accurately, justify the data and fill empty positions in the field. For T fields it needs to know the source of the data to be moved, and for F and C fields it needs to know the character positions to fill.

The record starts at character position 1 and ends at the limit of the record as specified in the record definition.

First character position: 29
Last character position: 34

5.5 INPUT AND VERIFY FIELDS

5.5.1 Field Type Requirements

In answer to the question 'Field type requirements:' in the JOBDEF dialogue, input fields require a specification of the characteristics of the data to be entered for that field. The specification must be supplied in the form of a rather cryptic string of characters, one for each characteristic. For example,

Field type requirements: ABL#

says that only upper case alphabetics (A) and blanks (B) are acceptable, that the data entered must be left-justified within the field (L) and that any empty characters to the right of the data are to be filled with blanks (#).

5.5.2 Acceptable Characters In The Field

There are 7 codes for acceptable characters which can be combined. Any character entered that does not meet the specification is greeted by a bell and the operator must delete or accept the character.

A	alphabetics, A-Z, upper case only
O	lower case alphabetics
C	lower case alphabetics are converted to upper case
N	digits, 0-9, and +,-
.	period (so that real numbers can be specified as 'N.')
B	blanks
S	all other characters on the terminal keyboard, such as #, @, &

5.5.3 Justification

Data must be specified as being left (L) or right (R) justified within the field. Left justification means that the data is aligned to the left margin of the field, and the field is filled to the right margin. Right justification is the opposite.

5.5.4 Field Filling

After the data has been set into the field, either left or right justified, any character positions left empty in the field are filled. The fill character must be speciifeid. It can be given as:

0	fill with 0's
#	fill with spaces
*	fill with asterisks
$	fill with dollars

Alternatively the typist can be required to type an entire field leaving no empty spaces. To do this use the code F, instead of a fill character code.

5.5.5 Subsequent Verification

If prompting is disabled during verification, it is often convenient to have fields that are not to be verified printed anyway, so that the typist can always be sure what field comes next. If P is coded the field will be printed during verification if it is not to be verified. If prompting is enabled, all the not-to-be-verified fields are printed anyway with their prompts.

5.5.6 Copying, Skipping And Nulling

It is often convenient for the typist to copy a field directly from the previous record, or to skip over a field. Sometimes this can be done automatically for each record, at least for a portion of a batch or job. However sometimes this cannot be allowed, for example, if the course code must be supplied for each student, the least that should happen if the data does not contain a course code, is that the typist should type a skip field, hear the error bell and accept the error. Then the record is marked by QDATA as containing an error which must be corrected later.

If any of copying, skipping or nulling are to be forbidden for a field, it must be explicitly specified:

K	forbid <esc>SF to skip field
U	forbid <esc>CF to copy field
Z	forbid <esc>NF to null field

Automatic copying or skipping must be permitted for fields explicitly. Then to use automatic copying or skipping the operator must turn it on using the <esc>A command in input mode.

"	allows automatic copying of the field
/	allows automatic skipping of the field

Naturally only one of the two can be specified. It is not possible to copy and skip at the same time.

5.5.7 Automatic Field And Record Termination

As with the record definition, if sufficient characters have been entered to fill a field entirely, it is not really necessary to enter a field terminator. If I is coded, QDATA will automatically generate the field terminator implicitly if the field is completely filled. Otherwise QDATA will require an explicit field terminator.

If T is coded, QDATA allows a record terminator to be used to terminate the field and the record. If the field is not the last field in the record, then the record terminator is equivalent to an <esc>SR to skip the rest of the record.

5.5.8 Some Examples Of Field Type Requirement Specifications

1. for an integer number

   Field type requirements: NOR

   Only digits, plus and minus are permitted. The number will be aligned to the right margin and filled with zeros to the left.

2. for a real number (with a decimal point)

   Field type requirements: N.OR

As for the integer number, but a period is allowed as well to separate the integer part from the fractional part.

3. for a sum of money such as $10.32

Field type requirements: N.S#R

The number itself will contain a dollar sign (special character) as well as a real number. The sum is right justified and filled with blanks, because we cannot have zeros to the left of the dollar sign.

4. for a date in the form dd-mmm-yy

Field type requirements: NACF

A date consists of digits, minus signs which are included in the N format, letters which could be input as lower case, but we want converted to upper case. The field is exactly 9 characters wide, and it should be filled to ensure that the operator has not missed leading zeros on dates before the 10th.

5. for the name of a person

Field type requirements: AO.BL#

Names consist of initials and last names, so that upper and lower case alphabetics, periods and spaces should be allowed. Names are usually left justified within a field and filled with blanks.

5.5.9 Prompt Messages

The prompt message is given as a character string optionally preceded by characters controlling the starting position of the message.

Prompt message: Chest:

Characters controlling positioning are:

\+ start the prompt on a new line

If the prompt does not start with a '+' then the prompt will appear on the same line as the last field.

Prompt message: +Chest:

5.6 TRANSFER FIELDS

All the additional information required under 'Field type specification' is the destination of the transfer. Only the starting position need be given, because the length of the field is already known from the First and Last Character specifications.

```
Field type: T
First character position: 29
Last character position: 34
Field type requirements: 105 !last position is 110
```

5.7 FILL FIELDS

'Field type requirements' for fill fields are merely the character used to fill the field:

```
Field type: F
First character position: 34
Last character position: 50
Field type requirements: *    !fill with asterisks
```

5.8 CONSTANT FIELDS

As with fill fields, the requirements are merely the character string used to fill the field:

```
Field type: C
First character position: 1
Last character position: 7
Field type requirements: Total:
```

5.9 EXAMPLE OF FIELD DEFINITIONS

5.9.1 Enter A Date In The Form dd-mmm-yy

The field is to be verified. It stretches over 9 characters from column 32 to column 40.

```
<<Field: 2>>
Field type: V                    !verify
First character position: 32
Last character position: 40
Field type requirements: NACF
Prompt message: +Date:
<<Field: 3>>
```

5.9.2 Transfer The Year To Another Field

The year entered in the previous field definition is in columns 39 and 40. Move it to columns 1 and 2.

```
<<Field; 3>>
Field type: T                    !transfer field
First character position: 39
Last character position:  40
Field type requirements:  1      !move to column 1
Prompt message: <ret>            !none for transfer field
<<Field; 4>>
```

5.9.3 Enter A 6 Digit Number Without Verifying

This number will be put into columns 1 to 6. Since verification is not required an input field should be used.

```
<<Field; 4>>
Field type: I
First character position: 1
Last character position:  6
Field type requirements:  NOR
Prompt message: +Number 1
<<Field; 5>>
```

5.9.4 Add A Fill Field To Blank To The Next Number

The fill field is 5 blanks long, between columns 7 and 11.

```
<<Field; 5>>
Field type: F
First character position: 7
Last character position: 11
Field type requirements:  <sp> !to blank the field
Prompt message: <ret>          !no prompt for fill fields
<<Field; 6>>
```

5.9.5 Transfer The Fill Field To After The Next Number

The next number occupies columns 12 to 17, so the fill field must be transferred to columns 18 to 22.

```
<<Field; 6>>
Field type: T
First character position: 7
Last character position:  11
Field type requirements:  18
Prompt message: <ret>
<<Field; 7>>
```

5.9.6 Set A Constant Word 'TOTAL' Before The Next Number

The next number will be in columns 34 to 39, so the word TOTAL must appear in columns 29 to 33.

```
<<Field: 7>>
Field type: C                   !constant
First character position: 29
Last character position:  33
Field type requirement:   TOTAL
Prompt message: <ret>           !none for constant field
<<Field: 8>>
```

5.10 EXAMPLE JOB DEFINITION - THE MAILING LIST

The mailing list example used throughout this guide has just one record format describing a member. The record is divided into 6 fields, the name, street, suburb, postcode, telephone number and date financial. So there must be one record definition and 6 field definitions, each comprising one record in the job definition batch. I will assume that the name, street and suburb fields are 34 characters long.

```
<<Record: 1>>
Format name: Member
Record length: 122
Number of  fields: 6
Data or check: D
Explicit terminator: I
Next record format: Member      !only one format
Force next format: F            !so force it
```

The first field is the member's name which can contain upper and lower case alphabetics and periods:

```
<<Field: 2>>
Field type: V
First character position: 1
Last character position:  34
Field type requirements:  AOB.L#
Prompt message: +Name:
```

The street is similar to the name except that digits must be allowed too.

```
<<Field: 3>>
Field type: V
First character position: 35
Last character position:  68
Field type requirements:  AOBNL#
Prompt message: +Street:
```

The suburb and the postcode are to be defined so that the operator can use automatic copy if there are several consecutive members living in the same suburb. The postcode must be exactly 4 characters.

```
<<Field: 4>>
Field type: V
First character position: 69
Last character position:  102
Field type requirements:  AOBL#"
Prompt message: +Suburb:
```

and

```
<<Field: 5>>
Field type: V
First character position: 103
Last character position:  106
Field type requirements:  NF"
Prompt message: +P-Code:
```

The telephone number is 7 digits wide and can only contain digits.

```
<<Field: 6>>
Field type: V
First character position: 107
Last character position:  113
Field type requirements:  NR#
Prompt message: +Phone:
```

Finally the date is 9 digits long and is composed of upper and lower case alphabetics, digits and minus signs. Minus signs are included under the N field type requirement. For the date we want to convert lower case alphabetics to upper case.

```
<<Field: 7>>
Field type: V
First character position: 114
Last character position:  122
Field type requirements:  ACNR#
Prompt message: +Date financial:
```

5.11 EXAMPLE JOB DEFINITION - NUMERIC DATA

In this job definition there are two record formats. The first contains one field specifying the number of data items following in the data records. The second record format is the data format. There are a maximum of 10 data items per line, each 10 columns wide. All data items are real numbers. If the record terminator is typed before the last field is entered that is ok and the rest of the fields are filled with zeros.

```
<<Record: 1>>
Format name: DataCount
Record length: 4
Number of fields: 1
Data or check: D
Explicit terminator: I
Next record format: Data
Force next format: F          !count always followed by data
```

```
<<Field; 2>>
Field type: V
First character position: 1
Last character position:  4
Field type requirements:  NRO
Prompt message: <ret>          !record prompt is enough

<<Record; 3>>
Format name: Data
Record length: 100
Number of  fields: 10
Data or check: D
Explicit terminator: I
Next record format: Data       !more may follow
Force next format: N           !can return to DataCount

<<Field; 4>>
Field type: V
First character position: 1
Last character position:  10
Field type requirements:  N.ROT
Prompt message: +Number 1:
```

The field definitions for the other 9 fields of the record are identical to the first field except that the character positions are different and the prompt message would be different.

5.12 EXAMPLE JOB DEFINITION - THE FORTRAN PROGRAM

This job definition is used for keying in FORTRAN programs. Each FORTRAN statement is broken into 5 fields:

1. The comment field in column 1 containing C or a blank
2. The statement number in columns 2 to 5, containing an integer
3. The continuation field in column 6, containing any character or blank
4. The statement field in columns 7 to 72, containing any characters
5. The sequence number field in columns 73 to 80, containing alphabetics and numerics usually

There are three different record definitions for the 3 different statements or parts of statements that make up a FORTRAN program. Most statements are simply lines of the program, called FORTRAN program statements. Statements may be followed by one or more continuation lines. Statements may be separated by comment lines.

All the records will contain 80 characters. The fields may be used rather differently for the different records.

5.12.1 The FORTRAN Program Statement Definition

There are 5 fields. The comment field is always left blank, and can be specified by a fill field. The statement number field is often filled, always by an integer, and the integer is required to be right-justified within the field. The continuation field is left blank and can be specified by a fill field. The statement field may contain any characters and should be left justified. The sequence number field contains alphabetics and digits only and should always be completely filled.

All program statements are to be transferred to the output file on release. An explicit terminator is not required for the record. The next record after a program statement is another statement, but this cannot be enforced because the typist may need to enter a continuation or a comment.

<<Record: 1>>
Format name: Statement
Record length: 80
Number of fields: 5
Data or check: D
Explicit terminator: I
Next record format: Statement
Force next format: N

The first field is a fill field to ensure that column 1 always contains a blank character.

<<Field: 2>>
Field type: F
First character position: 1
Last character position: 1
Field type requirements: <ret>
Prompt message: <ret>

The second field occupies columns 2 to 5, is an integer and must be right justified and filled with spaces. The field may not be copied, because each statement number is unique.

<<Field: 3>>
Field type: V
First character position: 2
Last character position: 5
Field type requirements: N#RU
Prompt message: +Statement number:

The third field is a copy of the first, since statements always leave the continuation field empty. The only differences are that this is record number 4, and the first and last character positions are column 6.

The fourth field occupies columns 7 to 72 and can contain any characters, to be left justified.

<<Field; 5>>
Field type: V
First character position: 7
Last character position: 72
Field type requirements: ABCN.S#L
Prompt message: +Statement:

The fifth field occupies columns 73 to 80, may only contain alphabetics and numerics and must be completely filled.

<<Field; 6>>
Field type: I
First character position: 73
Last character position: 80
Field type requirements: ACNF
Prompt message: +Sequence number:

5.12.2 The Continuation Line

First the typist must escape from entering fields back to entering records by using the <esc>FA (format alter) instruction.

<<Field; 7>>
Field type: <esc>Format Alter RECORD<ret>
<<Record; 7>>

The continuation line contains 4 fields. Columns 1 to 5 are always blank. Column 6 may contain any character except a blank, and must always be filled. Columns 7 to 80 are the same as for a statement number.

<<Record; 7>>
Format name: Continue
Record length: 80
Number of fields: 4
Data or check: D
Explicit terminator: I
Next record format: Continue
Force next format: N

The first field fills columns 1 to 5 with blanks.

<<Field; 8>>
Field type: F
Firstc character position: 1
Last character position: 5
Field type requirements: <ret>
Prompt message: <ret>

The second field is the continuation mark in column 6. It must be filled with any non blank character.

<<Field: 9>>
Field type: I
First character position: 6
Last character position: 6
Field type requirements: ACN.SF
Prompt message: +Continuation character

The third and fourth fields are identical to those in the FORTRAN statement record. They are records 10 and 11 in the job definition. Afterwards the typist escapes back to record level using <esc>FA.

5.12.3 The Comment Line

Comments have only 3 fields. Column 1 always contains the character C, so we can use a constant field to specify it. Columns 2 to 72 can contain any characters at all, and columns 73 to 80 contain the same sort of sequence number as for the previous two record types.

<<Record: 12>>
Format name: Comment
Record length: 80
Number of fields: 3
Data or check: D
Explicit terminator: I
Next record format: Comment
Force next format: N

<<Field: 13>>
Field type: C
First character position: 1
Last character position: 1
Field type requirements: C
Prompt message: <ret>

<<Field: 14>>
Field type: I
First character position: 2
Last character position: 72
Field type requirement: ABON.S#L
Prompt message: +Comment text:

And the final record 15 is identical to the final fields of the first two formats.

APPENDIX A

COMMAND LEVEL INSTRUCTIONS

Verb	Argument	Action
APPEND		add to the end of an existing batch
BATCH	number	set the number of the next batch to be used. Without the number, QDATA types the number of the current batch.
DELETE		delete the record just found by searching
DELIMITER		not yet used
ENTER	job name	begin keying a new batch under the job definition named
ERROR		find the next record containing a known error
EXIT		close all files and return to monitor mode
HELP	keyword	be helpful, regarding keyword
INSERT		insert a new record after the one just found by searching
NEXT		go to the next record
OPERATOR	name	record the operator's name as the last person to manipulate this batch. Without the name, QDATA types the name of the current operator.
PROMPT	keyword	change the prompt characteristics to keyword. Without the keyword, QDATA types the current prompt setting.
RECORD	number	search for the record number given. Without the number, QDATA types out the current record.
SEARCH	field /string/	search for a record containing the given string in the given field. Without the field or string, QDATA searches for a record containing

		the last string given in a SEARCH command.
SUPERVISOR		open up a new range of commands to a supervisor
TERMINAL		not used yet
TOP		find the first record
UPDATE		update the current record
VERIFY		start verifying the batch

Additional Supervisor commands:

ASSEMBLE	job name	assemble a batch as a job definition
CLEARVERIFY		clear verify so that the batch can be verified again
OUTPUT	filename	release the batch to the file system with the given name
PRINT		print the batch onto a lineprinter
REMOVE	batch or job	delete the given batch or job from the QDATA system

APPENDIX B

INPUT LEVEL INSTRUCTIONS

Command	Action
<ret>	field terminator in enter, append or verify; retain field in update mode
<lf>	record terminator in enter, append or verify; retain record in update mode
<esc>A(uto)	turn automatic copy/skip on and off
<esc>B(ackspace) F(ield)	delete the current field (if any), and return to the beginning of the previous field for reentering
<esc>B(ackspace) R(ecord)	delete the current record (if any) and return to the start of the previous record to update the record already entered.
<esc>C(opy) F(ield)	copy the contents of the matching field in the previous record into this field in this record
<esc>C(opy) R(ecord)	copy the contents of the previous record into this record for all fields remaining in the record
<esc>D(elete) F(ield)	delete the current field for reentering
<esc>D(elete) R(ecord)	delete the whole of the current record
<esc>F(ormat) A(lter) format	change the record format to that given for one record. If no subsequent record is set, return to the default record afterwards.
<esc>F(ormat) D(efault) format	set the given format as the default format
<esc>F(ormat) K(eep) format	as Format Alter, but continue to use the new record format for subsequent records.
<esc>H(elp)	;type the friendly helpful text
<esc>N(ull) F(ield)	fill the rest of the field with fillers, left justifying the data in the field

<esc>N(ull) R(ecord)	fill the rest of the record with the fillers specified for the individual records.
<esc>R(ecord)	type the current record or the previous record if the current one has not yet been entered.
<esc>S(kip) F(ield)	fill the rest of the field with blanks, left justifying the data
<esc>S(kip) R(ecord)	fill the rest of the record with blanks
<esc>T(erminate)	return to command level
<esc>W(hat)	type details of the current status of the data entry job

APPENDIX C

STANDARD JOB DEFINITIONS

These job definitions are held on the system library TED: and can be used by any operator:

CARD80 enter an 80 column card image with no restrictions whatsoever

FORTRA enter a FORTRAN source language program. The job definition contains 3 types of records defined as in chapter 5 except that the card sequence number is not compulsory, and a record terminator is allowed to skip the sequence number. When a completed batch is OUTPUT, each record will be filled to 80 columns with spaces, so to truncate trailing spaces and make the minimum sized file, the user will need to use

"_.COPY filename=filename/T".

JOBDEF enter a job definition for a QDATA job. JOBDEF is described fully in Chapter 5.

APPENDIX D

FIELD TYPE REQUIREMENT CODES

A	alphabetic characters, A-Z, upper case only
B	blanks
C	convert lower case alphabetics to upper case
F	field must be completely filled by operator
I	generate field terminator automatically if field is completely filled
K	forbid <esc>SF to skip the field
L	left justify the data within the field
N	digits 0-9 and +,-
O	lower case alphabetics
P	print the field during verification even if it is not a verify field
R	right justify the dat within the field
S	all special characters not covered by other codes
T	allow a record terminator to finish the record before all the fields have been entered
U	forbid <esc>CF to copy a field
Z	forbid <esc>NF to null a field

FIELD TYPE REQUIREMENT CODES

0	fill field with zeros
.	allow periods
*	fill field with asterisks
#	fill field with blanks
$	fill field with dollars
"	allow automatic copying
/	allow automatic skipping