Presented at the Association for Computing Machinery Conference, Washington, DC, February 18-20, 1975

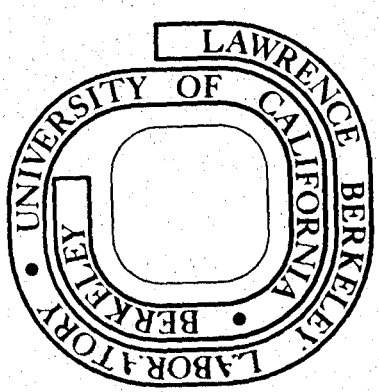
LBL-3655
c.1

DESIGN TOOLS FOR DATA HANDLING SYSTEMS

Gill Ringland

January 31, 1975

Prepared for the U. S. Atomic Energy Commission
under Contract W-7405-ENG-48



LBL-3655
c.1

DISCLAIMER

Design Tools for Data Handling Systems

Gill Ringland

CAP Ltd,
14 - 15 Gt. James St.
London WC1, England

Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

ABSTRACT

Complex data bases are now routinely accessed online. Proper design of access procedures is vital to insuring the security of the data and that it can be recovered if necessary. The design tools discussed in the paper enable operator procedures to be charted in a way that identifies the interrelation between system recovery points and message sequences. The paper also describes the lower level charts (flowcharts to describe interactive processing) used as specification for programs to be executed between recovery points.

Examples are given, from an order entry application. This is representative of applications which are primarily data entry and without the multitube of branching options encountered in some retrieval systems.

1. INTRODUCTION

This paper defines a charting methodology which has been used successfully in the design of online systems. The difficulties in the design of these systems are twofold: ensuring adequate performance, and data security. It has been our experience that using graphic design tools improves the system definition stage by showing the structure clearly. This helps designers to isolate bottlenecks before they give performance problems. Even more important it enables designeers to identify processing points at which data is saved, for instance in a database. When these points have been identified, the data security and system recovery strategy can be planned.

The charting methodology also has been found invaluable in describing a proposed system to users. They, since they have existing practices and procedures, the users may well provide input on which parts of the processing may involve operator training, or which are less convenient than need be. These comments, when they can be accommodated without altering the security of the system, will really help in system commissioning and acceptance.

The charting methods are thus seen, not as an end in themselves, not as providing pretty documentation, but as design tools. It is the author's opinion that computer systems designers have been too reluctant to use graphic design tools. This paper shows one charting methodology which has been proven in use - the aim is to provoke others to define extensions or to refine the scheme or perhaps to implement for use as an interactive design tool on graphic displays.

The author is not familiar with any charting schemes in general use which have the same aims as the one described. James Martin, for instance (2) discusses realtime system design and the man-machine interface but does not

develop a charting scheme to unify the two system aspects. Similarly, while there are many strategies for designing programs to improve this reliability, see for example, (2), the structure charts are related to the static structure (e.g. formal calling parameters, subroutine levels) rather than the execution sequence of the code.

This feature of the charting scheme described below, that of reflecting the execution structure, is particularly important for realtime systems, in which it is usually "socially desirable" to keep the smallest amount of code possible in main store to process a message. By connecting the unit of code stored at any one time to the data required for recovery, the extent of damage from errors may be automatically assessed. This is of particular importance for systems in which data is captured online.

A charting scheme for describing the structure of stored data is described in a companion paper (3).

The rest of this paper is divided into two sections. The first describes the formalism and symbols. The second uses these in the context of a semi-realistic example, online order entry.

2. FORMALISM

The charting methodology is used at three levels in the system: to describe operator procedures, the system structure, and the individual processing modules.

The operator procedure charts show the sequence of messages which the operator expects, and the valid actions (such as data entry, cancel, restart) which can be taken in response to each screen. The program structure charts relate the processing and data security aspects of the

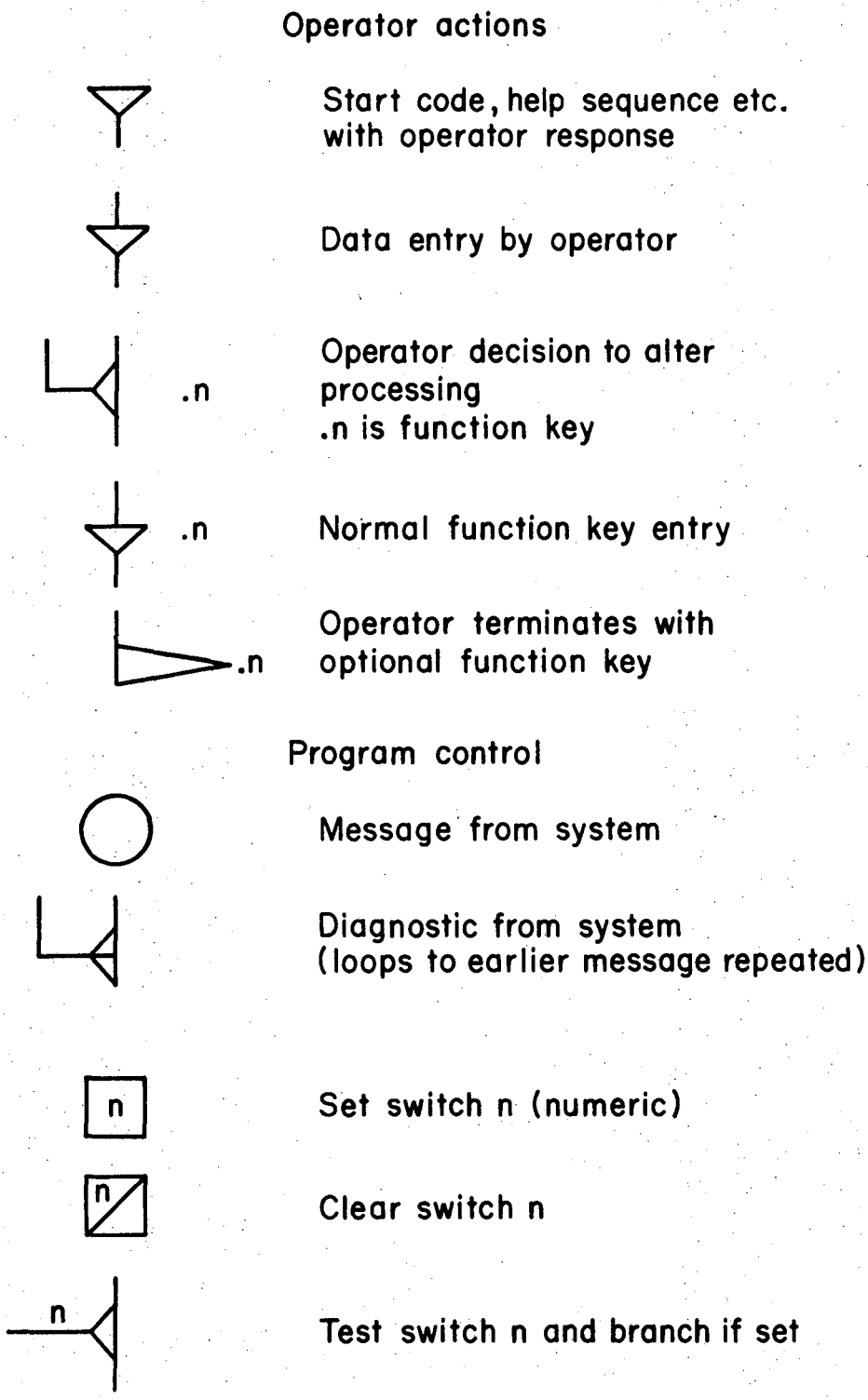


Figure 1. Operator procedure chart symbols.

system during execution to the message written out and entered. At the most detailed level, conventional flowcharts show the processing consequent on a given message at data entry point. The organizational unit at this level is referred to as a message processor.

Operator Procedure Charts

These charts use the symbols given in Figure 1. The operator actions are always taken in response to screen formats displayed, giving a range of options (a menu format) or some data fields to be filled (a questionnaire) or the possibility of entering a function code (.n) to alter the sequence of processing.

Start and help screens, used at the start of processing or after a break, can be initiated either by the system or by the operator after losing her place in the processing.

The data entry symbol on the operator protocol chart is accompanied by a description on what data items are to be entered. The validation of the data items may fail, in which case a diagnostic may appear. The operator then tries again to insert the data items.

Function keys are implemented by hardware on many keyboards. The screenful of data is transmitted and control is passed to a different screen from that normally found in sequence. So, for instance, "end of batch" would often be signalled by a function key depression. (For convenience, in the example we will show the function codes into which the keys are mapped, rather than the function keys e.g. EB instead of .3).

The switches, which are set and tested by program, are introduced to explain why different screen sequences may be seen by the operators,

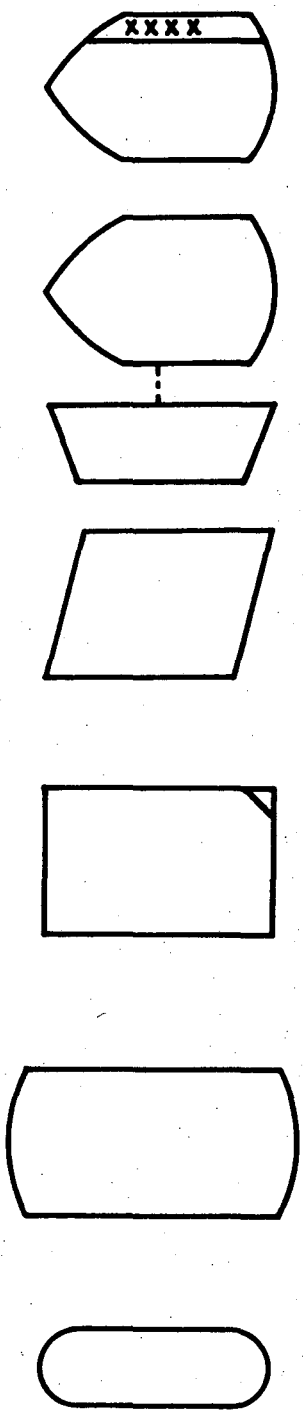


Figure 2. Block chart symbols.

depending on circumstances.

Block Charts

These charts use the symbols defined in Figure 2. They are used to produce an overview of the system actions, in particular the interrelation between operator actions and the sotrage of data on backing store.

The operator input may be validated and corrected by a loop which requires her to retry data entry. This may be shown explicitly by a dotted line between the screen and input symbols, to represent operator think time.

It is important at this level of design to define at what stages of processing recovery procedures may use saved information and when data must be reentered. This is the purpose of the "access to backing store symbol" on the block chart. Further, the status area is defined as the area which is saved when the mapping system is entered, to output a message or request more data.

The use of diagnostic messages indicates that the item has not been accepted, and data entry must be repeated. The symbol therefore may appear in conjunction with the display + input symbol, giving further details of validation procedure.

The entry or exit point appears as for normal flowcharting, to indicate that control is passed either ot the monitor or operating system, or to another message processor, or to an entry point of this message processor.

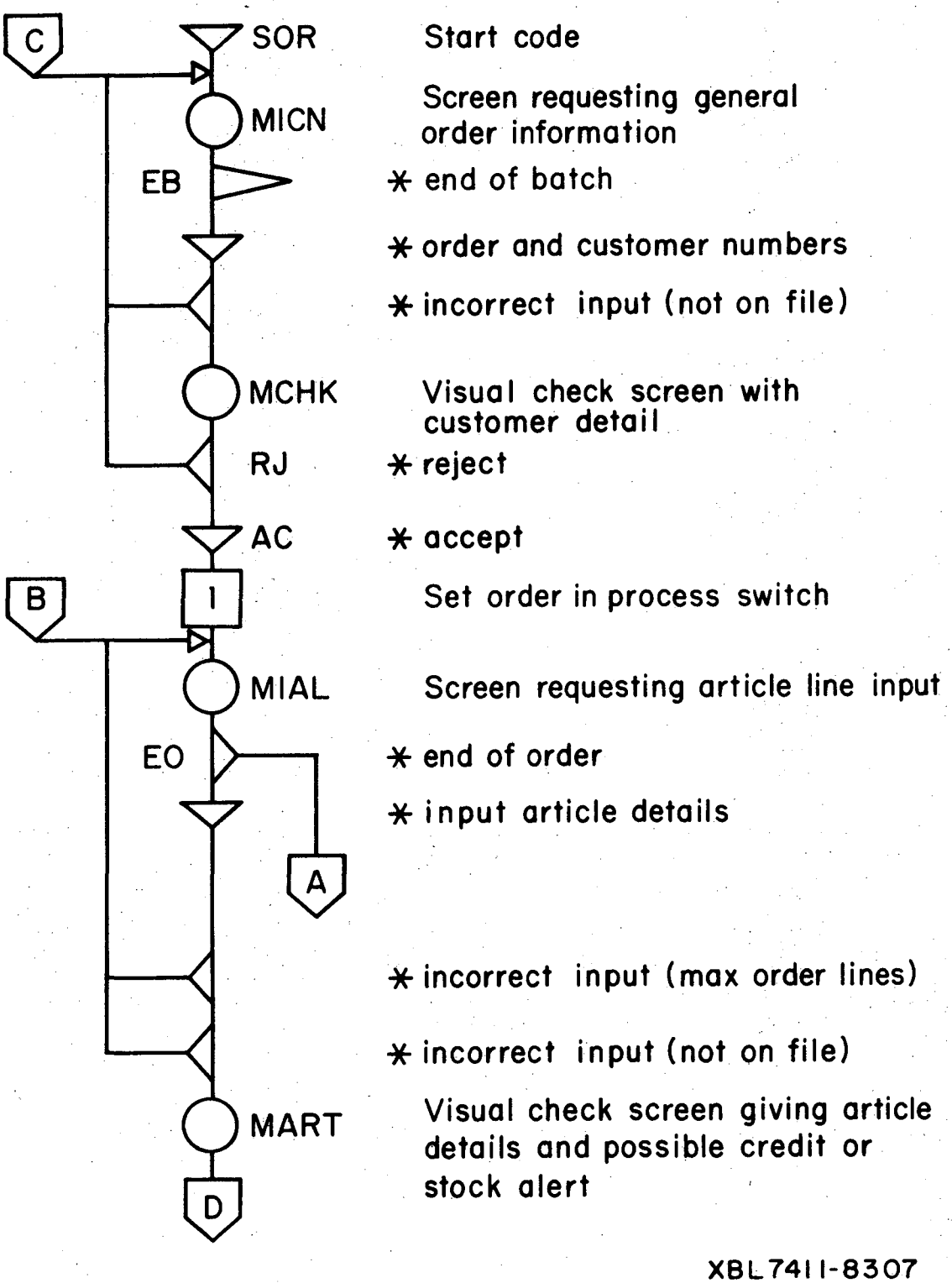


Figure 3.1. Operator procedure for order entry.

Flowcharts

The conventional flowcharting symbols are used, to indicate the processing between data entry, and exit back to the monitor a operating system, within a message processor.

3. EXAMPLE - ONLINE ORDER ENTRY

To follow the example, the reader will probably find the following list of assumptions helpful, even though they are not central to the point of this paper:

* all orders once keyed in and accepted are stored as entered on the order file.
* those articles which cannot be immediately supplied are written to the pending order file in the form of a back order.
* an initial warehouse note is made for all those articles of an order which can be supplied.
* sequential files for later batch processing are made for those articles of an order which cannot be supplied at all and those for which insufficient stock is on hand.
* the article file is updated when the warehouse note is issued.
* warehouse notes for back orders are issued in batch, i.e. outside this system.

Naming Conventions

For convenience, some naming conventions have been used on the charts to indicate the activity represented. The format is: type code, three character identifiers, where type code refers to the type of activity, and the three characters identifier represents a section of the processing.

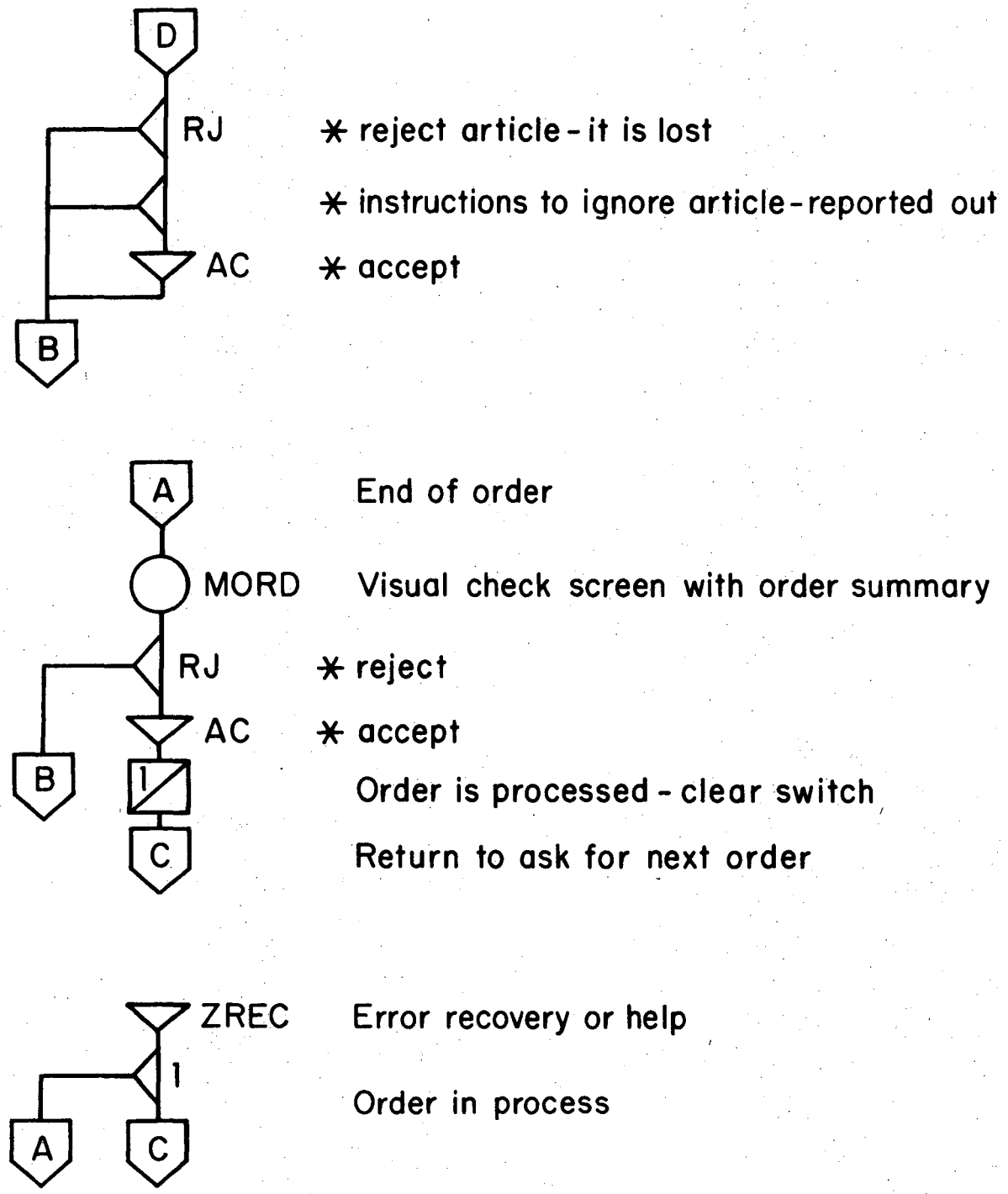


Figure 3.2. Operator procedure for order entry (continued).

Some examples might be helpful:

*ZXXX are exit points to a system process from a message processor, or vice versa.

e.g. ZREC for the recovery procedure,

ZEND to end the application.

*SXXX are entries to the application by providing a start code.

e.g. SORD for start of order entry.

*LXXX are entries to or exits from a message processor, when initialisation is not needed and transfer is via a linkage list.

e.g. LORD for processing of order entry.

LART entry to message processing for each article line.

*MXXX are map definitions used by the message processors and the mapping system, used to format input and output data

*PXXX is the name of a message processor.

e.g. PORD is the name of the order entry message processor.

Operator Procedures

An example of an order entry procedure is shown in Figure 3. It shows that following a start code, to specify that the application required is order entry, a message referred to as MICN is displayed or typed. This requests information on the next order to be processed. The operator may enter order and customer details, which may be incorrect and need the operator to key a replacement. Alternatively, the operator may, on first being presented with the screen for order and customer details, decide that there are no more orders to be processed. Any new application could thereafter be started by keying a new start code.

The second point clearly shown by the operator procedure chart is that some screen sequences follow as a normal part of processing, forming a loop.

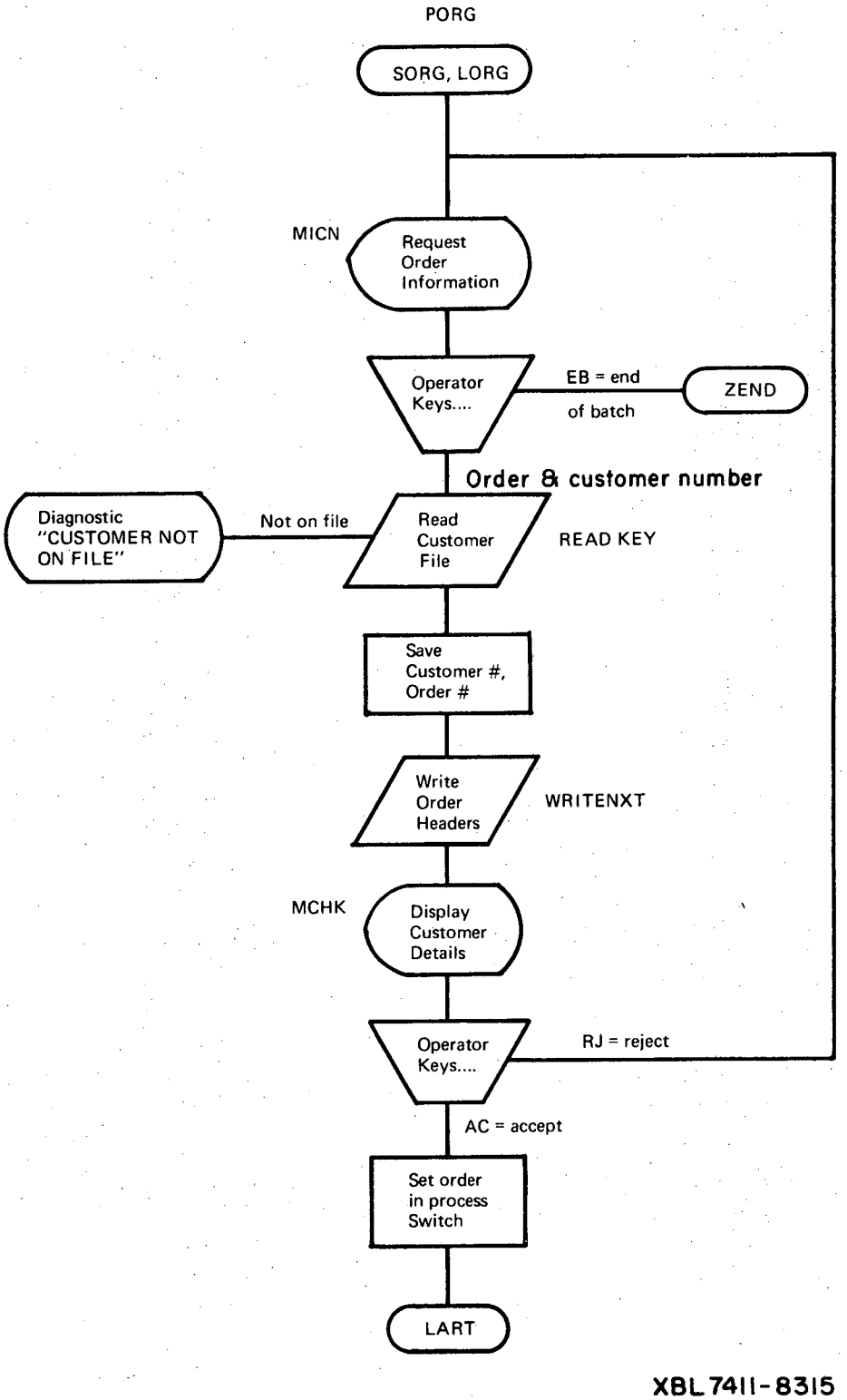


Figure 4. Order header block chart.

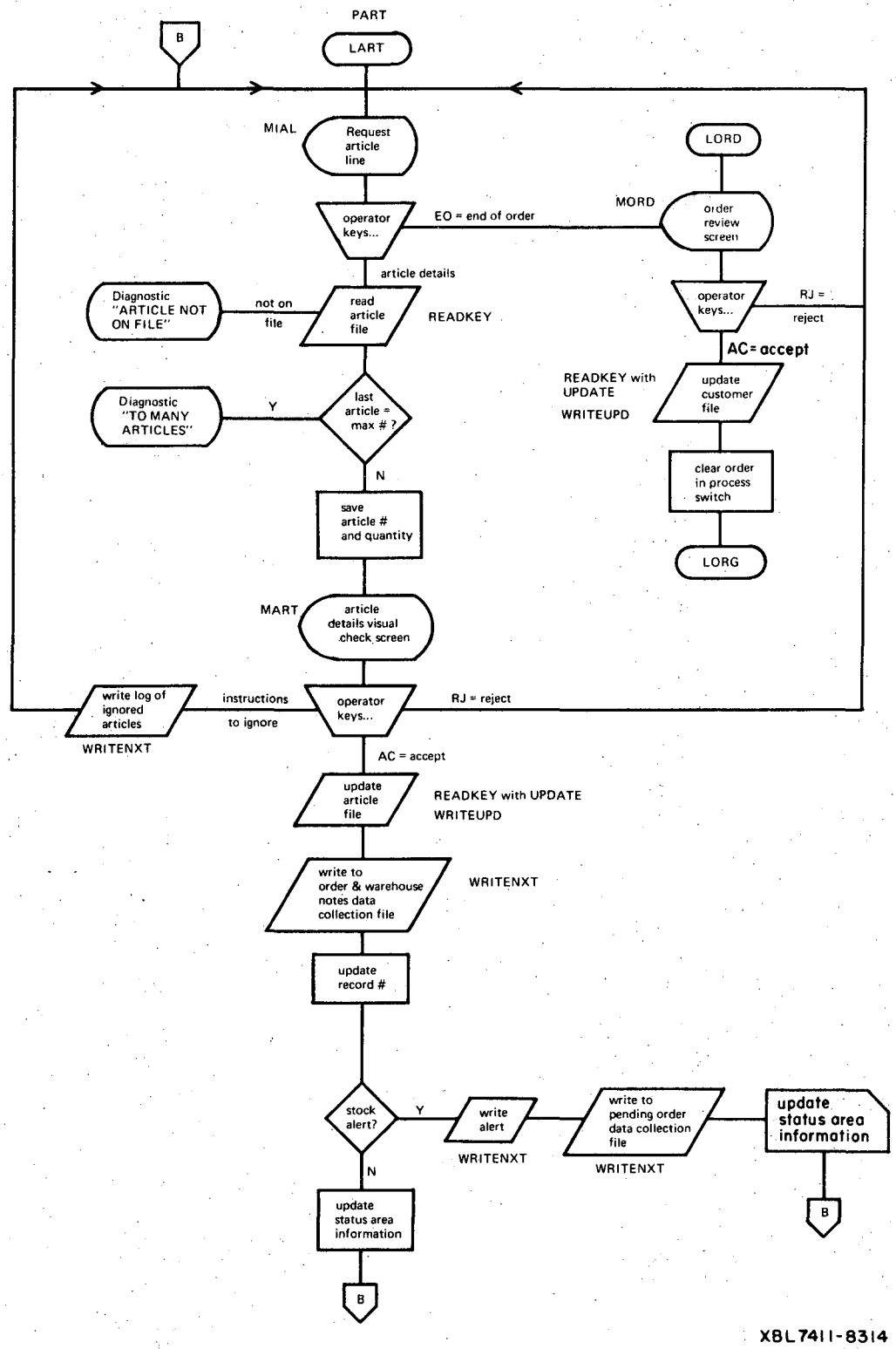


Figure 5. Order article block chart.

Within the loop, error paths under program control or operator decisions may cause the sequence of messages to change. For instance, in the article line input section, the normal path is from the article line questionnaire, through computer validation and visual check screen, to acceptance of the order.

The loops are used to define coherent sections of processing which will be grouped together into a message processor.

System Design and Block Charts

When the operator procedure for an application has been established, the inter-relation with the file architecture must be established, and the recovery points identified. This is done by means of block charts.

The block chart of order entry (Figures 4 and 5) shows the information which the systems analyst provides for the programmer. He defines the application-oriented diagnostics (such as 'article not on file' and the stage at which this validation should be performed. He defines the data in the status areas, which is saved each time a new message is output, and used for recovery. He defines the points in the processing at which files are read and written.

It can be seen that the loops identified on the operator procedure chart are reflected in the block chart.

Message Processor Structure

The programmer uses the block charts to organize his work into message processors. The order entry application divides into three message processors, PORD, PART and PERR in a logical manner. It is anticipated

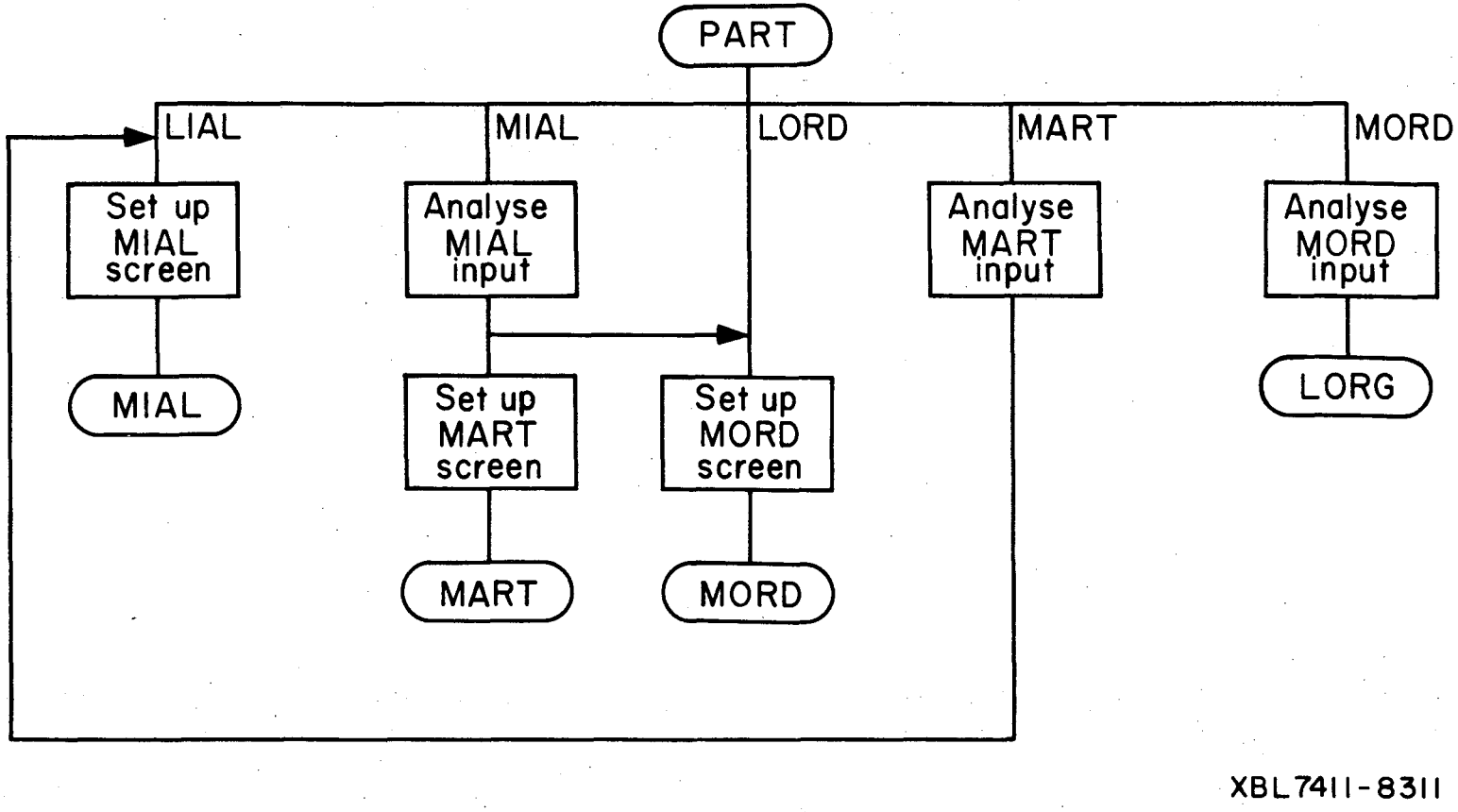


Figure 6. Message processor structure chart.

that in this type of processing, the structure of a message processor would be as typified in Figure 6.

In this message processor, there are several legs. Each leg is similar to a normal section of executable code in that there is a defined point at which user data may be available as input from a keyboard, some processing (including possibly file I/O), an output message is set up and the program exits. The let may be entered as a result of a message (e.g. MART), or as a result of a call from another message processor (e.g. LORD), or as a result of a start code (which is not relevant for this message processor).

Each leg may be conveniently represented by standard flowcharting techniques.

4. CONCLUSION

The author does not claim this to be a perfect scheme -- however, she hopes that it will help analysts in the field to develop a framework that can be used for their applications and to reduce their problems.

The scheme was developed out of a predecessor defined by J. Johnson of CAP. It was jointly formulated by a team consisting of Esmond Hart, Pat Bailey, Bob Wustman, Andrew Patterson and the author.

REFERENCES

1. James Martin "Design of Realtime Computer Systems", Prentice - Hall, 1969

2. J. L. Ogdin "Improving Software Reliability" Datamation, January 1973

3. E. A. Hart and G. Ringland LBL Report 3659 (1975)

TECHNICAL INFORMATION DIVISION
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720