design and implement actual systems since ideas for applications will always exceed the supply of those capable of seeing the application through to successful implementation. It will be essential for manufacturers to supply simple and efficient software modules, all useable with each other, and supported with high quality documentation. Also implementation assistance, effective field service, and maintenance support will be needed. System analysis and design may also come from the manufacturers, in some cases, but will be more likely to be provided by consultants. Manufacturers must provide meaningful training courses and effective manuals on the equipment, the software, and its potential applications. Too few people know what to do and those who do must generalize, write it down, and distribute it as widely as possible so that others may learn.

Mini-computers have a great future limited more by our collective ability to understand how they can be used than by any deficiencies or omissions in the hardware.
Teleprocessing systems software for a large corporate information system

by HO-NIEN LIU and DOUGLAS W. HOLMES

Pacific Gas and Electric Company
San Francisco, California

INTRODUCTION

One of the functions of management is to control the organization in such a way that it responds to changes and deviations in the optimum manner. The magnitude of the deviation from the established goal often depends upon the length of the delay in response, any deviation from the best performance objectives must be quickly detected and corrective measures applied promptly.

A fast response corporate information system is designed to accommodate this criterion with the following capabilities:

1. Keeping the Corporate Data Base Freshly Updated

Source data may be transmitted directly into the computer to improve the efficiency of the information flow, thus providing prompt and accurate collection of data from widely dispersed areas. This capability can at least provide the following benefits:

   • Reduction in human waiting time.
   • Reduction in idle resources.

2. Extending the Usage of the Corporate Data Base

New applications could be added to provide benefits not previously available.

   • Direct exchange of information with the corporate data base helps users in diverse locations keep abreast of rapidly changing events. For example:

     • Immediate presentation of operating status aids decision making.
     • Rapid transmission of decisions to the point of execution can be accomplished.
     • Swift distribution of decisions to the associated parties for supplemental decision-making are completed within the time frame.
     • Timely feedback of the results of the decisions allows adjustments to the operating environment in an incremental manner.

A well planned and developed teleprocessing system will provide the backbone of a fast response corporate information system. The remainder of this paper describes the requirements, strategy, facilities, and actual implementation of such teleprocessing system.

SYSTEM REQUIREMENTS

The following requirements are essential for a teleprocessing support of a growth-oriented corporate information system.

1. Support for a Variety of Terminal Types

Each terminal installation must be reviewed to determine the specific terminal type which can best handle the types and volumes of information processing typical of that location. The system must be capable of supporting, in addition to the standard devices, several special devices tailored to satisfy special situations.

   • The standard devices will include:
     • Typewriter terminals
     • CRT terminals
     • Low-price, short-message terminals for data entry
     • Card readers, card punches, and line printers for remote locations.
   
   • The special devices could include:
     • Analog transducers
     • Process control computers

2. Centralized Control of Tele-Communications Network

To assure efficient information flow and optimal utilization of the communications network, control of the teleprocessing system should be centralized so that resources can be allocated dynamically to satisfy changing demands. Conventional systems
have often allowed segmentation of the system resources into disassociated subsystems between which temporarily unused resources cannot be shared.

For a large scale party-line (multidrop) network, provisions should be made to maintain a network discipline that will ensure increased system efficiency as well as dependable service. For example, continuing to poll a malfunctioning terminal which fails to reply degrades service to other terminals on the same line and wastes central processor time. Such a terminal, therefore, should be deleted logically from the network until it is capable of replying.

By contrast, when maintenance personnel are testing a malfunctioning terminal on-line, the system must poll this terminal as usual until testing is completed.

The network control program should provide at least the following functions:

- Polling and addressing network discipline.
- Threshold error counters to allow automatic deletion of malfunctioning lines and terminals.
- Diagnostic terminal mode which bypasses the automatic deletion of malfunctioning lines or terminals to allow for on-line hardware maintenance.
- Manual stop and start for lines and terminals to allow console operator to control network.

3. Support for a Variety of Message Types

Each application project should be able to select the message types best suited for its needs. The system should support a variety of basic message types which may be used independently or as building blocks for more complex activities.

These basic types of messages are:

- **INQUIRY**: An operator may ask predefined questions by specific transaction codes.
- **RETRIEVAL**: A user may select and examine information elements from the data base.
- **DATA ENTRY**: An operator enters new information into the data base, whether update occurs immediately or later.
- **JOURNAL**: An application project reports the status of transactions previously processed.
- **MULTIPLE DESTINATION**: A message processing program responds to a single request with messages directed to two or more locations.

A complex activity example:

- **DATA CHANGE (INQUIRY + DATA ENTRY)**: An operator inquires into the data base and then enters changes based on the inquiry response.

4. Versatile and Balanced Message Control Facility

In support of these message types, the message control program should also provide the following services:

- **HEADER BUILDING**: Identification, time stamping, routing, and classification of messages to permit off-line analysis of message flow in addition to on-line control.
- **QUEUE MANAGEMENT**: For a large real-time system, the interval between message arrivals is often less than the service time so that messages cannot be processed serially nor can the system keep up with the demand for its resources. The resulting backlog of messages must be managed to smooth out peak loads and provide a tolerable response time.
- **PRIORITY MANAGEMENT**: Certain activities are of such importance that they demand immediate attention regardless of the backlog of other messages. A priority scheduling mechanism would permit such activity to avoid long waits in queues by providing express routes throughout the system.

5. Efficient and Easy Applications Programming

Economic considerations require an approach be taken which reduces programming, testing, and maintenance costs of message processing programs. The teleprocessing system should present an interface which permits such cost reduction.
6. Testing Provision for Message Processing Programs

To facilitate testing new or modified message processing programs in an actual operational environment without endangering the on-going operations. The system should protect at least the following resources:

- **DATA BASE**: Retrieval of data elements from the data base should function normally for the testing program, but any attempts to update the data base, directly or indirectly, must be intercepted.
- **OPERATIONAL PROGRAM**: A different storage protection key should be assigned to the testing programs.

The system should also provide the following functions:

- **MESSAGE TRACE**: Print or log every work area associated with a testing program to show the message in different stages during processing as a diagnostic aid. This function should be available by request for operational programs also.
- **REFRESHING**: After trying a specific condition to which the testing program fails to respond normally, it is desirable to refresh the copy of the testing program from the library so that different conditions can be tested to speed up the debugging cycle.
- **TASK INDEPENDENCE**: If one testing program fails, the system must take action for abnormal termination of this individual program (subtask); however, all the other programs in the same region should continue processing independently of this failure.
- **TIME LIMITING**: A program should be terminated if it does not complete within a specified time limit. This function is of value for operational programs but especially for testing programs to break tight programming loops.

7. Data Base Security

To protect the integrity of the corporate information system data base, security measures must be provided against unauthorized update and retrieval of privileged information.

Security should be a function of the operator's level of authority, the location of the terminal, the transaction code, as compared to the sensitivity of the data element.

8. System Reliability

A real-time information processing system must demonstrate its reliability to its users.

There are three aspects to reliability in any system:

- **ENDURANCE**: Protection against failure of its own programs, and graceful degradation of the system under adverse conditions.
- **RECOVERY**: Provision for restarting the system close to the point of failure after the disturbance has been removed or corrected.
- **AUDIT TRAIL**: Each day's message log should be retained for a period of time to permit reconstruction of a single event or a sequence of events which led to failure of a program module or the system.

After the fact analysis is often the only technique possible for problem identification/solution in a real-time environment. Some provision should be included for a computer search of the message log when specific selection criteria permit.

9. Facility to Evaluate System Performance

Usage statistics should be gathered to detect problem areas of the system worthy of special attention, so that solutions can be implemented to improve:

- Main frame throughput
- Network traffic
- Terminal operation efficiency
- Application program proficiency

**STRATEGY OF THE SYSTEM**

The principal strategy entails the reduction of redundant coding otherwise inherent in the massive application programming effort by shared system modules, wherein the following disciplines should be imposed on the system directly or indirectly:

- Application Program Proficiency
- Network Traffic Efficiency
- Terminal Operating Efficiency
- Main-Frame Throughput Efficiency

Let us define the term "application program" as referring to a message processing program tailored to handle one or more varieties of messages as identified by the transaction codes.

The three stages of application program structure described below will demonstrate the progression of teleprocessing software architecture for a large corporate information system.

**STAGE 1: Centralized Data Management Functions for All Application Programs**

A previous paper (1) has described in detail how to centralize the data management functions to obtain
the following benefits:

- Reduction of Core Memory Requirement
- Reduction of Program Loading Time
- Centralized Control of Shared Data Base
- Optimal Allocation of Resources Associated with Shared Data Base
- Flexibility in File Design and Record Layout

After excluding the data management function from an application program (Figure 2), the following six key functions remained to be performed:

1. **Input Buffer**
   Get message from message queue as it arrived at input buffer; the message string is in original terminal form, containing terminal control characters and varies in length and format.

2. **Input Edit**
   Normalize the input message string to fixed form by interpreting the terminal control characters, replacing absent characters and fields with nulls. Because different types of terminals have unique sets of control characters and logic, an application program that contains this function will always be dependent on the type of terminal and its logic.

3. **Data Validation**
   Validity check the information content of the incoming message string to intercept bad data and send out error messages so the user may correct and reenter the message.

4. **Process and Access Data Base**
   Process message content and exchange information with the corporate data base. Construct the logical response message to the user in fixed form without terminal control characters.

5. **Output Editor**
   Convert the logical response message from internal format to display format inserting terminal control characters for transmission. If there is more than one message to be sent to different types of terminals, construct different message strings to corresponding terminals.

6. **Output Buffer**
   Dispatch message in terminal form.

**STAGE 2: Independence of Application Program From Terminal Hardware Characteristics**

Shared message editors normalize input messages and format output messages in order to isolate the application programs from the tedious function of terminal control character interpretation.

Several advantages are derived from this approach:

1. **Programming Proficiency**
   - One application program can handle similar information from several types of terminals each with a format most suited to its special features.
   - Shared message editors permit optimization of terminal characteristics at low programming cost since they need be programmed only once.
   - New terminal types may be added and input/output display formats redesigned without application reprogramming.
   - High-level languages, such as COBOL and PL/1, can be easily applied to process fixed format message records.

---

**Figure 2—Stage 1**

**Figure 3—Stage 2**
• Applications programs are easier to design, program, and test.

2. **Main-Frame Efficiency**

• Static core requirements for application programs and work areas are reduced.
• Application programs process a message more quickly, reducing the dynamic core requirements, measured in bytes occupied per second.

3. **Network Efficiency**

Optimized use of terminal control characters shortens the message length, conserves message transmission time, reduces line load, and permits an increase in the number of terminals per line; the communications network may then comprise fewer lines at a sizable reduction in installation and maintenance costs for a given number of terminals.

4. **Operator Efficiency**

Optimal use of terminal format control characters increases operator efficiency as much as it relates to display readability and input cursor control. Since the application program is truly independent of the display format, it need not be changed when a display format is redesigned or modified. This feature simplifies making improvements in terminal display design formats.

**STAGE 3: A Single Retrieval Module Replaces Many Application Programs**

Progressive reduction of redundant coding from Stage 1 and Stage 2 application programs have already placed the following functions in the shared system modules.

• Network and Message Control
• File Definition and File Access

• Data Definition and Data Retrieval
• Input and Output Buffer Management
• Input Message Normalization and Editing
• Output Message Formatting

Only three functions remain to be performed by the application program.
• Input Data Validation
• Processing logic and the interface with file management programs for data retrieval from the shared data base
• Pattern editing for output message; i.e., insert decimal point, comma, $, etc.

Expanding the input message editor to perform the function of “input data validation” and the output message editor, “pattern editing,” there remains only one function for the message processing program, and even this very last function can be performed by shared system modules.

• A shared system module can obtain information from the message descriptor to request data element retrieval from the data base via the data management modules.
• For most basic message types, such as INQUIRY, RETRIEVAL, DATA ENTRY, JOURNAL, etc., the processing logic can be easily represented by a simple list which defines the processing path through and within the shared system modules.

Therefore, if we create a simple list for each transaction code, the shared system modules can perform the required processing logic without recourse to an application program except when extraordinary processing logic occurs. The Stage 3 teleprocessing system will add benefits in addition to those previously derived in Stage 2.

1. **Programming Proficiency**

• Shared input data validation and output pattern editing permit optimization in program design and efficiency.
  • Input data validation can be designed, coded, and tested in optimal fashion at low programming cost since they need be programmed only once.
  • Standard error messages can be generated directly.
• Application programming, testing, debugging for most transactions are eliminated.

2. **Main-Frame Efficiency**

A single resident reentrant module replaces many many application programs, eliminating the roll-in, roll-out time.
3. **Network Efficiency**

A well-designed input data validation and error message notification technique can effectively cut down the amount of bad message traffic in the network.

4. **Operator Efficiency**

Standard error messages make it easy for the operator to take corrective action on bad input data; the problem of inferring the same meaning from different error messages coded by programmers in various application programs is avoided.

**SYSTEM LOGIC FLOW**

The system support package comprises the regional resource manager, the two message editors, the retrieval module, and the data control manager. Refer to Figure 5. The following paragraphs describe the general system logic flow.

1. **Initialization**

At the beginning of the day, the regional resource manager performs the following functions:

- Builds program and descriptor directories to expedite the paging activities.
- Opens the message queues and on-line files.
- Prepares the shared buffer pools.
- Loads the resident modules.
- Attaches input and output message editors as operationally independent subtasks.

When initialization is complete, the input editor is activated and begins to scan the input queues for a message to process.

2. **GET Message**

Scanning of the message is accomplished by means of a table of displacements into a list of message queue control blocks. A pointer to a control block may appear up to four times depending upon the priority assigned to the message queue.

If a scan of the entire list fails to find a message, scanning is suspended for a predetermined period of time and then recommenced.

3. **Application Message Header**

After a message is gotten from the input queues, the input message editor constructs a message header for the application program (see Appendix II). This header serves the following functions:

- Passes pertinent information relevant to:
  - Message disposition.
  - Message status.
  - System status.

- Passes information to the application program in a form readily usable by high level languages.
- Permits the application program to pass certain information back to the system.

4. **Process Routing**

The input editor obtains the process routing information from the transaction code table and constructs a control list for the regional resource manager.

For the majority of transactions, this control list will make no reference to an application program. For those special transaction codes requiring complex processing logic, however, a reference to the
associated application program will be included in the list.

5. Message Format

The message may simply be a request for a message format. The input message editor posts control to the output message editor to generate a message format (captions and control characters) based on the information in the message descriptor.

6. Normalize Input Message

The input message editor processes the elements in an order controlled by the input message descriptor (Appendix I). The input message descriptor sequentially defines the attributes of each message field:

- The starting position in vertical and horizontal coordinates.
- The maximum length.
- A field designated as caption will be eliminated from processing.
- A field designated as mandatory information must be present or the entire message will be rejected.
- The retrieval descriptor index points into the retrieval descriptor so that additional editing may be performed.

The retrieval descriptor defines additional attributes for the message field:

- The length and location of the area reserved for the message field.
- The data class expected; i.e., numeric or alphanumeric.

Based on the above information, the input message editor performs the following functions:

- Checks for invalid characters, such as a letter in a numeric field, posts error condition if invalid character found.
- Deletes punctuation, such as commas in a numeric field.
- Aligns to the left or right.
- Truncates if the input message descriptor length exceeds the retrieval descriptor length, such as if the operator included too many decimal positions.

7. Data Validation

The retrieval descriptor serves for both editing and data validation. As many as 256 data validation routines may be programmed to permit the choice of an appropriate validation technique. Some examples of checking:

- RANGE: Test the numerical value of a data field against a predetermined range of values.
- CODE: If the input data field is a code argument in a table, the system will perform a table look-up to determine if it is a valid argument.
- RECORD KEY: If the input data field references a record key in an on-line file, the system can issue a read (key) against that file to determine if it is valid.
- FIELD ASSOCIATION: When one or more input fields depend upon the value of another input field, the system can match them against a predefined associative decision table.
- DATE: The following tests can be applied to a date field:
  - Any valid date.
  - A holiday.
  - A work day.
  - Today's date.
  - Test a range of predefined work days from today's date.
  - Test a range of predefined elapsed days from today's date.

8. Standard Error Message

If any errors have been identified, a standard error message is prepared.

The following considerations are taken to design the standard error messages:

- FIXED LOCATION: Error messages always appear in the same location to attract the terminal operator's attention. For example, all the error messages will appear on the last two lines of the CRT screens.
- MULTIPLE ERRORS: To conserve network efficiency and eliminate unnecessary traffic arising from bad messages, the system will handle up to four errors per message at a time.
- STANDARD PHRASE: The error message will reference the specific input field and indicate the kind of error the system detected for that field.

9. General Retrieval Module

The message may have been a general retrieval request. The retrieval descriptor would then have been a core-resident skeleton sufficient to satisfy the normalization and validation routines. The data identification information (element control numbers) supplied by the operator will be inserted in a copy of the skeleton so that data retrieval may proceed.

The retrieval routine builds a list containing the file name, the record key, and one or more data
element control number and receiving area pairs, and requests the services of the data control manager. Nulls are returned for a data element when the operator’s or terminal’s security clearance is less than that assigned to the element. A status code informs the retrieval routine of any abnormality. The retrieval descriptor defines whether an abnormal status code is to be ignored or considered an error.

10. Output Message Editor

The output message editor pattern edits the data field if it is required, prefaces whatever control characters are necessary to position the message character string in the terminal format. It has the following three modes of operation:

- **Device With Non-Formatted Memory**
  - Blanks between graphics in the same line and blanks at the front of the line are replaced everywhere possible with control characters.
  - Lines are truncated on the right after the last graphic.
- **Device With Formatted Memory:** When the transaction requires a new format at the device.
  - The memory of the device is cleared.
  - Fields marked as variable in the output message descriptor (device dependent option) are inserted in the message string full size without the blanks suppressed, even if they are completely blank.
  - Blanks in caption and format fields are replaced with control characters wherever possible.
- **Device With Pre-Formatted Memory:** When the transaction requires the same format as is at the device.
  - Caption and format fields are omitted.
  - Blanks in variable fields are replaced with control characters wherever possible.
  - The effect of each control character is considered with respect to the existing format.

11. PUT Message

This routine places the prepared response message in the proper destination queue for dispatch to the receiving terminal.

12. Termination

A privileged transaction code allows the console operator to terminate the teleprocessing system.

The system termination module directs the message control region to discontinue polling on all lines as soon as incoming messages have been received.

When polling has been discontinued, the input editors are directed to return control to the regional resource manager whenever they find the process queues empty. Normally, they wait a predetermined period of time and then scan the queues again.

When all message processing is complete, the regional resource manager terminates the teleprocessing system.

**SYSTEM FACILITIES**

This section will briefly describe some system support functions not mentioned in the previous section:

1. **Terminal Start-Up Procedure**

An operator must log-on before attempting any other activity on a terminal. For his own protection, he should log-off when his work is completed or during an interruption in which he leaves the terminal.

When log-on occurs, an employee number is entered in the terminal table. This employee number is used to set up individual restrictions on the terminal and to facilitate error and security violation tracing. Each time a log-off is processed, a corresponding log-on is required before business can be resumed.

A second operator may log-on at a terminal without the previous operator logging off; the second operator’s employee number and restriction code replace the first’s.

2. **Data Base Security**

Six modes of operation are supported for terminals:

- **Business:** Normal mode for business work. Most applications functions are valid in this mode. INQUIRY, RETRIEVAL, DATA ENTRY, DATA CHANGE, MULTIPLE DESTINATION and JOURNAL are available and work as defined.
- **Training:** Operator training mode for practicing business work.
  - INQUIRY and RETRIEVAL work as defined.
  - DATA ENTRY and DATA CHANGE appear to the operator as defined but fail to update the data base.
- **Supervisor:** Extended mode for business work. All applications functions are valid in this mode. At the application’s discretion certain transactions may be reserved for supervisor mode or more information may be passed in this mode.
A terminal in supervisor mode may:

- Put another terminal in the same office in supervisor mode if that terminal is authorized for supervisor mode.
- Display the employee presently logged on a particular terminal.
- Copy a message to another terminal in the same office.
- Diagnostic: Systems aid for on-line engineering maintenance; message directed to special diagnostic programs which display generated status information on the console terminal.
- A terminal in diagnostic mode may return itself to any mode authorized for it.
- It may copy any terminal in training mode and any terminal may copy it.
- Console: Network control terminal located at the computer console; terminal, line, and transaction code status tables may be altered from this terminal.
- Master: Network monitor terminal; permits dynamic observation of system for debugging and audit control. Master mode terminals may change the mode status of other terminals to any mode of operation, including master.

3. Network Monitoring

Hardware errors will be analyzed and may cause the following actions to be taken:

- Send an Error Message
  - Describes the error to the console operator.
  - Describes action being taken by the system.
  - Suggests action which should be followed by the operator.
- Manually start or stop a terminal, line segment, line, or group of lines from operation on request.
- Automatically stop a terminal, line segment, or group of lines, depending on error which occurred.
- Redirect messages to an alternate terminal (which may be a different type) when the original destination terminal has any type of hardware error which renders it unable to transact business.

4. Formats

For data entry, a formatting facility preformats a terminal’s memory with caption material and control characters at the operator’s request. The operator indicates the particular transaction format required by suffixing a letter F to the associated transaction code. The facility responds with the format, and the operator simply fills in the data.

If the information content of the message is acceptable for processing, the system restores the variable fields to blanks, and the operator can proceed to the next transaction if it is the same. If it is not, a new format may be requested.

5. Data Collection

A data collection facility stores audit trail records created by application modules which update the data base and transaction records destined for off-line batch processing programs. These records are sorted and cataloged by off-line programs for easy retrieval at the end of the on-line day.

6. Testing Program Library

A special program module library contains all test status programs. Any module loaded from this library is automatically placed in test status to protect the integrity of the data base from unproved routines.

IMPLEMENTATION OF THE SYSTEM

The teleprocessing system package has been written in IBM-360 Operating System Assembly Language (ALC).

1. It is fully interfaced with the IBM-360 operating system MVT (multiprogramming with variable number of tasks) environment.
   - The application programs and the system programs operate as independent subtasks of the regional resource manager; abnormal termination of a subtask will not stop the remaining subtasks in the region.
   - The package is not tied to any particular release of O/S; hence, if a new version is released, there should be little effect on this package.
2. The teleprocessing system package takes full advantage of the existing operating system facilities.
3. It is intended to interface with all the operating system supported languages (COBOL and ALC interface have been implemented).
4. The entire package has been designed to be dynamic in nature; that is, all programs are load modules. They are not linkage edited into the application program; thus, the package may be redesigned and improved without any appreciable effect on the application programs.
5. The entire package has been programmed in re-entrant code.
6. The system has been coded in a modular fashion. Each routine was individually coded, tested in detail, subgrouped, and finally all routines were combined together.
7. The message control and message processing regions are independent of each other to permit relocation.
of the control program to a front-end communications computer when the network size warrants the change.

8. The hardware anticipated over the next several years includes two large central processors with a million bytes of main memory, supported by smaller satellite computers and a score of multi-drive disk storage units. The system is being designed to support several hundred terminals, most of which are expected to be high speed CRT display units.

ACKNOWLEDGMENT

The authors wish to thank Mr. J. R. Kleespies for his encouragement and support; Mrs. G. L. Kenny, Messrs. R. A. DaCosta and P. A. Terry for their dedicated efforts in design and programming; Mrs. L. J. Fiore for her careful typing; Mr. R. P. Kovach, University of California, for his early research and design of the system; and Messrs. F. J. Thomason and J. W. Nixon of Haskins & Sells for their invaluable advice.

BIBLIOGRAPHY

1 H LILU
   A file management system for a large corporate information system data bank
   FJCC Proceedings 1968
2 J MARTIN
   Design of real-time computer systems
   Prentice Hall 1967
3 J MARTIN
   Programming real-time computer systems
   Prentice Hall 1965
4 Advances in fast response systems; fast response system design and auditing fast response systems
   EDP Analyzer February 1967—March 1967 and June 1967
5 C HAUTH
   Turnaround time for messages of differing priorities
   IBM Systems Journal Volume 7 No 2 1968
6 W P MARGOPOULOS others
   Teleprocessing system design
   IBM Systems Journal Vol 5 No 3 1966
7 M G JINGBERG
   Notes on testing real-time system programs
   IBM Systems Journal Vol 4 No 1 1965
8 J D ARON
   Real-time systems in perspective
   IBM Systems Journal Vol 6 No 1 1967
9 J L EISENBIES
   Conventions for digital data communication link design
   IBM Systems Journal Vol 6 No 4 1967
10 L F WERNER
    Software for terminal-oriented systems
    Datamation June 1968
11 J DIEBOLD
    Thinking ahead—Bad decisions on computer use
12 Heading format for data transmission
    A USAAII Tutorial Communications of the ACM

APPENDIX I

Input message descriptor (fixed form)

| BITS | 1. Retrieval Descriptor Index | 8 |
|      | 2. Message Field Length       | 7 |
|      | 3. Mandatory Field Indicator  | 1 |
|      | 4. Line Number (Vertical Spacing) | 6 |
|      | 5. Spare                      | 1 |
|      | 6. Spare                      | 1 |
|      | 7. Position (Horizontal Spacing) | 7 |
|      | 8. Caption Delete             | 1 |

Input message descriptor (free form)

| 4 BYTES |
| 1. Retrieval Descriptor Index | 8 |
| 2. Message Field Length       | 7 |
| 3. Spare                      | 1 |

Output message descriptor

| 2 BYTES |
| 1. Retrieval Descriptor Index | 8 |
| 2. Device Dependent Options   | 8 |
| 3. Line Number (Vertical Spacing) | 6 |
| 4. Data Scan Override         | 1 |
| 5. Format                     | 1 |
| 6. Position (Horizontal Spacing) | 7 |
| 7. Caption Field              | 1 |

BITS

<table>
<thead>
<tr>
<th>1. Data Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. Arithmetic (Right Numeric, Left Zero Fill, Decimal Alignment (0-7))</td>
</tr>
<tr>
<td>0 0 0 Binary, Display As Decimal</td>
</tr>
<tr>
<td>0 0 1 Binary, Display As Hex</td>
</tr>
<tr>
<td>0 1 0 Packed</td>
</tr>
<tr>
<td>0 1 1 Zoned</td>
</tr>
<tr>
<td>b. Alphameric (Left Alignment, Right Blank Fill)</td>
</tr>
<tr>
<td>1 0 0 Graphics (Terminal's Entire Character Set)</td>
</tr>
<tr>
<td>1 0 1 Alphabetic</td>
</tr>
<tr>
<td>1 1 0 Alphanumeric</td>
</tr>
<tr>
<td>1 1 1 Numeric</td>
</tr>
</tbody>
</table>

| 2. Decimal Alignment (For Arithmetic Class Only) |
| 0 to 7 Places or Date Verification Decision Table (Replaces Decimal Alignment Table) |

From the collection of the Computer History Museum (www.computerhistory.org)
APPENDIX II

Application message header

TPCMSHDR The 40 byte application message header allows communication between the message editors, MSGIN and MSOUT, and the application module. Use of the information is left to the discretion of the application analyst.

TPCINTM Arrival Time of Day

A binary clock maintains the time of day in units of 1/150 second (6.2 milli-seconds). The high order byte contains binary zeros.

The application may insert this field in the generated transaction records' sort keys to post by arrival sequence.

Because the conversion of this field to decimal hours, minutes, and seconds is time consuming, it is not appropriate to do so in the on-line environment.

TPCUSER User Status Flags

MSGIN initializes this field to binary zeros.

MSOUT logs it in the QTAM message header.

QDUMP retrieves it at the end of the day for application analysis.

Each application may define its own coding structure. However, the codes should, in the least, describe the TPCSCODE selected and explain why, so that the application analysis can reconstruct the process condition.
TPCMODE Terminal Mode

A terminal may be placed in one of several operations modes which define how the system will react to messages from it.

1. Training—The application appears normal to the operator, but no transaction records should be generated or posted to the masterfile.

An application may desire to maintain special files of pseudo accounts for training and testing and post these in training mode.

2. BUSINESS—The application reacts normally to all stimuli.

3. Supervisor—This mode is normally for terminal operator supervision but on occasion some business work will arrive from a terminal in ‘supervisor’ mode. The application may handle such work as business or may grant special privileges.

Supervisor mode is allowed only for specific terminals and specific employees in supervisory positions.

4. Operator—This mode is normally for systems operation, but on occasion some business work will arrive from a terminal in ‘operator’ mode.

5. Master—This mode is normally for systems programming, but on occasion some business work will arrive from a terminal in ‘master’ mode.

TPCINNR Message Sequence Number In

QTAM maintains an input message sequence number for each terminal

The application may insert this field in the generated transaction records for journal sequencing or as a debugging trace.

TPCDATE Today’s Julian Date, YYDDD+

This field is supplied for the application’s convenience.

Because the conversion of this field to calendar format, e.g., YYMMDD, is time consuming, it is not appropriate to do so in the on-line environment.

TPCSCODE Transaction Code Modifier

TPCTCODE Transaction Code

A transaction code identifies an entry from the operator and the related response to the operator.

The transaction code modifier X'FO' is assigned to the entry on input and to the standard response for a valid entry on output.

The modifiers X'Fl', X'F2', X'F3', X'F4' designate alternate responses selected by the application module processing the entry. They must, however, be designed to fit the display format of the standard response

X'FO' is the standard response.

X'Fl' is the error description response.

For Data Entry and Data Change

X'Fl' is the standard acceptance response which re-initializes the terminal buffer and screen for the next entry.