This paper describes an operational teleprocessing system that allows both low-speed conversational data entry and high-speed remote job entry and output. At the same time, the multiprocessing system provides conventional batch processing.

The total system configuration, which consists of standard equipment, is briefly introduced. Then the teleprocessing control program, the major programming support developed for the system, is described in detail. This program functions as the interface between the teleprocessing lines and the input/output streams of the batch processing system. The final topic is the terminal program, which is provided for the central processing units at the terminals.

A teleprocessing approach using standard equipment

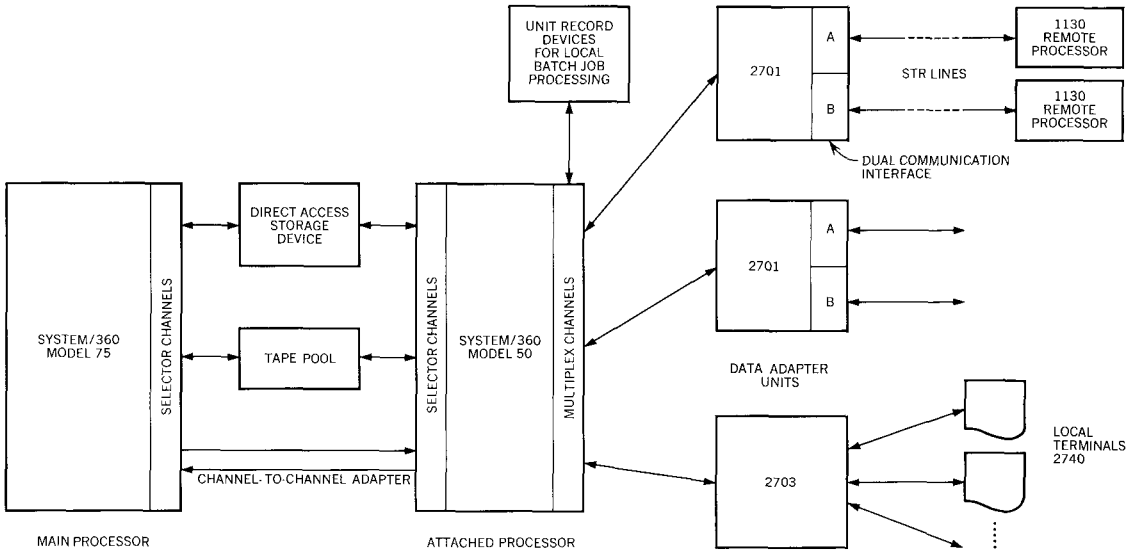
by R. D. Wade, G. P. Cawsey, and R. A. K. Veber

A generalized teleprocessing system has been developed to meet the specific requirements of the Central Electricity Generating Board (CEGB) in London, United Kingdom, which produces and distributes the total electricity supply for England and Wales to local area boards.[1] As the interface to a conventional batch processing system doing advanced applications, this teleprocessing system allows both low-speed conversational data entry and high-speed job entry and output.

The Central Electricity Generating Board has traditionally had a large scientific job-shop installation. The expanded use of computers to assist in the control of the national electricity grid transmission system required a teleprocessing design that would provide rapid results once a control engineer decided on the need for a calculation. The calculations are necessarily complex, requiring a large computer to provide results in a reasonable time (a few minutes). Fast response is also aided by allowing the control engineer to enter data directly into the computer from a remote location and to read directly the output presented to him from a calculation. In addition to performance, efficient use of main storage was a basic aim in the design. The system described is now in operation and has proved successful in all of these respects.

This paper briefly discusses system considerations, including the total configuration. Standard equipment is used, and the programming support provided for the equipment is only extended

Figure 1 General system configuration



to provide the teleprocessing capability. The main emphasis of the paper is on the principal program developed for the system, the teleprocessing control program. This program controls the transmission and receipt of data along both high-speed and low-speed lines to a variety of terminals. The last topic discussed is the program developed for use at the remote termnials.

System considerations

The teleprocessing system is based on standard IBM SYSTEM/360 computers supported by the SYSTEM/360 Operating System. The system configuration is shown in Figure 1. The principal terminal supported is the IBM 1130 computer system, since it allows the combination of a keyboard for data entry and a relatively fast printer for output. Although the ensuing description assumes use of the 1130 to illustrate the concepts, any remote terminal could be used that can communicate with the SYSTEM/360.

The teleprocessing control program (TCP) provides an interface between the teleprocessing lines and the input/output streams of a conventional batch processing installation. The programming conforms with the Synchronous Transmit-Receive (STR) method of data communication, but the system includes local IBM 2740 communication terminals that can be used through the teleprocessing control program for local conversational file manipulation. Although the techniques described in detail assume the STR mode, the basic philosophy of the system is equally applicable to the Binary Synchronous Communication (BSC) method.[2] In fact, the package is now undergoing conversion to BSC. At the same time, remote start/stop terminals, such as the IBM 2740 terminal, will be included, extending the package for additional applications.

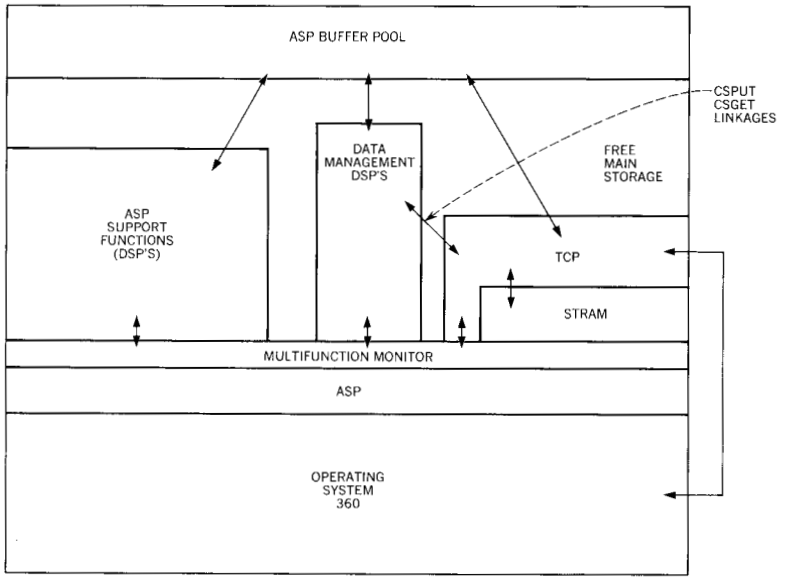
Since the system generally deals with a large number of terminals, special consideration was given to supporting the dual communications interface feature of the IBM 2701 data adapter, which allows two lines to come into one adapter, the switching of lines being program controlled.

The teleprocessing control program functions under the standard Attached Support Processor (ASP),[3] which is a TYPE 2 extension of the SYSTEM/360 Operating System. ASP is a multiprocessor operating system designed to automate the operation of one or more computers interconnected via channel-to-channel adapters. ASP resides either in a separate smaller computer (the attached processor) or in a main storage region of one of the larger computers. Although teleprocessing programming support is now available with ASP, the project described here was developed independently of this support because of the special-purpose application aspects of the project.

ASP provides a flexible base, in association with the operating system, for such teleprocessing work. This system combination helped toward achieving the design aims of minimum main storage and optimum performance; it also allowed, without additional effort, a separate operator's console at the central system to be designated solely for control of the teleprocessing system. This console is currently a local 2740. However, all the techniques described are general enough to be applied to any of the input/output operating system support provided for the SYSTEM/360.

The teleprocessing program provides the on-line, automatic interface to the teleprocessing network. A fundamental design

Figure 2 Program interaction in the attached processor



objective is for it to run with minimal operator intervention. Once contact is established with a terminal, the program then automatically controls that terminal, even in diagnostic mode.

An essential requirement for this application is the ability to diagnose any type of error on a dynamic basis. Therefore, techniques were developed to automatically provide line-error statistics, line-interruption traces, and on-line diagnostic tests of equipment. While the teleprocessing system is working, the central SYSTEM/360 computers, under the combined control of ASP and the operating system, continue with normal batch processing. To support this general teleprocessing function, only minor changes are made to existing IBM SYSTEM/360 programming support.

The teleprocessing system is a step towards providing the facilities of a real-time control system using standard SYSTEM/360 equipment. With the multiprocessing ASP programming support, it is already feasible to design and operate a configuration with little redundancy that can be altered to continue running despite most malfunctions.

Teleprocessing control program

general concepts

The teleprocessing control program controls the transmission and receipt of data along Synchronous Transmit-Receive (STR) lines to remote processors on the lines. It initiates all input and output operations over the lines, and it can control any number of remote processors. The TCP provides the interface between the data management programs of an ASP system and the standard programming support for SYSTEM/360, as shown in Figure 2.

ASP consists of a set of modules, called dynamic support programs (DSP's), multiprogrammed on a time-sharing basis to support such functions as card reading, printing, and volume mounting, and to control one or more large SYSTEM/360 computers running under control of the operating system. To ASP, the teleprocessing control program is just another DSP. It, together with supporting DSP's, is used to provide a data management interface between the TCP and the normal job stream and to print out line-interruption traces. All of these functions are done asynchronously.

The TCP uses a modified version of the TYPE 2 STR access method (STRAM) routine.[4] The modifications allow use of STRAM in the ASP environment and allow the TCP to, in effect, wait for multiple events, which is not normally possible for a DSP. Coding has been added to the STRAM I/O interruption-handling and error-recovery routines so that, when an event is completed, its status is set for any data adapter and an indication is given to the TCP. The error routines have also been modified so that operating system error messages are routed to the TCP and not to the operators' console. Other more specific additions to and exits from the STR access method routine are described in relevant passages.

system requirements

The data management DSP's must allow conversational mode operation, so that the remote user can build and update data files

on the ASP support processor. In addition, the DSP's must provide for remote job entry, so that job output from the batch processing stream can be transmitted at high speed to the printers of the remote processors. These two modes of operation are provided by the TCP for the data management DSP's, the mode used being determined through system macroinstructions.

In addition to these two operating modes, the remote users require 24-hour-per-day access to the central SYSTEM/360 computers. Therefore, it was essential that the whole teleprocessing system be made as simple to use and as reliable as possible. The TCP and associated programs were developed to meet these requirements with minimum disruptive influence on the normal batch processing operations of the central computers.

As well as ease of operation, it was important that the TCP could be expanded to handle more remote terminals of various types, as required. To achieve this, and to allow parallel development of the various parts of the system, the TCP and associated programs are of modular design. The TCP itself controls data transmission, while the reentrant data management DSP's handle the data in the central computer, operating asynchronously with the TCP. This modular approach means that additional remote terminals can be incorporated into the system using high- or low-speed lines without significant increases in the main storage required.

The design of the TCP was also influenced by the need to incorporate support for Binary Synchronous Communication links. The modular approach ensures that the only additions that are required are in the more detailed line control routines within the TCP. All of the remaining modules of the system are independent of the type of transmission and of the type of remote processor or terminal.

The TCP automatically attempts to establish contact with all the remote processors after every central computer failure, and data are not lost because of such breakdowns (i.e., the remote user is not affected by a central computer breakdown except for being unable to gain access to the system during the time of the breakdown). The TCP also ensures that data are not lost due to failures of the teleprocessing links and includes automatic recovery of all detected transient error conditions, such as noise on the lines or short breaks in transmission. In this way, completely unrecoverable errors are almost eliminated and, when they do occur, the TCP informs the central computer operators and allows them to restart data transmission after clearing the cause of the error, without loss of data.

During development and testing of the system, a number of fault detection techniques were incorporated, which are summarized here and described in greater detail later in the paper. The first of these was a line-interruption trace (LIT) program, which stores indicative data in a disk data set after every input/output operation on a teleprocessing line. This data can be printed out

selectively during normal operations and was essential to the initial debugging of the TCP. After the initial testing stages, this facility was retained as an aid to fault analysis and is used in conjunction with a number of on-line diagnostic routines.

The 24-hour-per-day operation of the system implied that time would not be available for running normal diagnostic programs for fault detection (since they would completely interrupt the ASP system). The on-line diagnostic routines are executed asynchronously with the data management DSP's servicing other remote processors. The diagnostic routines are simple to operate and can be initiated by either the remote processor user or by the central computer operator.

As a further aid in fault detection, statistics of data transmitted and transmission errors are kept for every line. These statistics are automatically logged out on a central computer console, giving an indication of any lines that have an error rate higher than a preset expected rate. These topics are discussed in greater detail later in the paper.

operating modes

The telecommunications control program has three modes of operation:

- *Initialization mode*, during which contact is established with particular remote processors.
- *Data communication mode*, during which the conversational and remote job entry facilities for the data management DSP's are provided. This is the normal operating mode.
- *Diagnostic mode*, in which attempts are made to diagnose suspected equipment faults and identify the system components from which errors are originating.

initialization mode

The initialization mode can be entered in either of two ways. Normally, as soon as ASP becomes active, the TCP automatically attempts to set all lines attached to the system into initialization mode. This attempt is made without any console operator action, although the TCP informs the teleprocessing operator that it is carrying out this operation. Also, initialization of one or more particular lines can be requested by the teleprocessing operator entering a command.

To initialize a particular line, the TCP initiates a series of fixed data transmissions on that line in an attempt to identify the remote processor connected to the line. When a line is operating in initialization mode, the second interface of the adapter is idle; meanwhile, other lines connected through active adapters can operate as normal in data communication mode.

Any permanent transmission errors (i.e., repeated data checks or failure to establish synchronization) during initialization are counted by the TCP. If the remote processor has not been identified before that count reaches a preset maximum, the attempt to initialize the line is abandoned and the console operator informed.

If the remote processor is identified successfully, an entry is made in a table called the TCP linetable, which is built into the

TCP program. This table associates each active remote processor location (by its having a unique number) with the adapter and interface to which that remote processor is currently attached. Thus, for example, the data management DSP's can direct data to a particular geographical location, by specifying the unique number assigned to that location, without knowing which adapter is, in fact, connected to the remote processor.

If the initialization has been successful, the updated version of the linetable is printed out for the console operators' reference, and the line is set into data communication mode.

data communication mode

In the data communication mode, the TCP sends data messages continually to the remote processor. Where both interfaces of an adapter are connected to remote processors in data communication mode, messages are sent alternately to the two terminals. This is the normal operating situation. It allows both remote processors to be used concurrently for data input and output. (In a normal remote job entry system, only one remote processor is attached to each adapter at one time.)

The TCP establishes synchronization with the remote processor attached to one interface and transmits a data block to that terminal. On receiving an acknowledge sequence in reply, indicating that the data has been received successfully, the TCP transmits a second data block. After receipt of the second block has been acknowledged, the TCP establishes synchronization with the remote processor attached to the second interface and transmits two blocks of data to that terminal. The TCP continues sending data blocks alternately to the two remote processors either until one of the terminals has some data to transmit to the central SYSTEM/360 or until one of the lines leaves data communication mode. While synchronization is established on one interface, the TCP temporarily ignores the other interface, leaving the remote terminal attached to it continuously trying to establish synchronization.

When the user of a remote processor wishes to transmit data to the central system, the remote processor replies to a data block with an inquiry sequence instead of an acknowledge sequence. This abnormal sequence is detected by STRAM, and an error indication is passed to the TCP. The TCP checks the type of error and, in this case, indicates that no real error has occurred and then attempts to read data from the remote processor. The remote processor can send one or two data blocks followed by an end-of-transmission (EOT) sequence. The TCP receives the data and replies each time with an acknowledge sequence until it receives the EOT sequence. It then replies with its own EOT and returns to the normal alternate sending of data blocks to the two remote processors.

If one of the remote processors connected to an adapter fails to be initialized (i.e., remains idle), the TCP sends data to the other remote processor only. The failing adapter is left for operator intervention at the central computer. The operators determine corrective action from the error information displayed, and can subsequently request the TCP to attempt another initialization of

that line. Thus, the TCP can control any number of adapters with any combination of terminals that are idle or in data communication mode, or in fact, in either of the other two modes—initialization or diagnostic.

The data blocks transmitted by the TCP can be of three types, identified to the remote terminal by a format character at the beginning of each block. The first type is dummy data, consisting of two blocks of four characters. Dummy data are ignored by the remote processor but give it the opportunity of causing a line turnaround (a reversal in the direction of transmission). The second type is data from the data management DSP's to be printed on the remote processor line printer. The third type is data from the data management DSP's to be typed by the remote processor console typewriter. Data of the first type, dummy data, are sent whenever there is no data of the other types.

Table 1 TCP operating commands

Name	*Command*
OPEN	Set a line in initialization mode.
CLOSE	Set a line to idle, and reset all status switches for that line.
RETRY	Retry line indicated in command, which has had a permanent TCP error. This command is used after operator has corrected likely cause of original error.
SEND	Send a data message from the teleprocessing operators' console to a remote processor. This function is also provided by the message-switching facilities of the data-preparation DSP's.
TAB	Initiate preparation of a list of all lines currently in data communications mode. This list is the TCP linetable set up during initialization mode, which indicates each line and its equipment address that has successfully entered initialization mode and has not been reset to idle.
CANCEL	Delete the TCP from the ASP system. This command is accepted only if all lines are currently idle.
EXEC	Set a line to the diagnostic mode, and run the test indicated in the command. After the test has been completed, the line is returned to the mode it was in before the command was executed.
STATS	List the data transmission statistics for each line. These statistics are accumulated by the TCP.
ALTER	Perform indicated one of several fault analyses and diagnostic functions. For example, alter operating speed of an adapter (e.g., from 2,000 baud to 1,200 baud) and indicate to the data management DSP's to send a text message to one or more remote terminals using the message-switching facilities of the data management DSP's.

The TCP operating commands available to the teleprocessing console operator and the functions of those commands are listed in Table 1. For all TCP console operating purposes, the remote processor or lines are referred to by their data adapter unit number and interface.

diagnostic mode

The diagnostic mode can be entered either following an ASP teleprocessing operator's command to the TCP (an EXEC command) or when a line is in data communication mode and the remote user enters a request for a diagnostic function. This remote request is intercepted by the TCP before it reaches a data management DSP.

Following a request to enter the diagnostic mode, the TCP sets flags to prevent further data transmission. After waiting for any current activity to be finished, the TCP initiates the test by linking to the diagnostic driver module. While the test is being conducted, the other interface on the adapter being tested is set to an idle state. The diagnostic driver analyzes a test parameter block passed to it by the TCP and loads in the required diagnostic routine. The routine initiates some input/output activity on the line and then returns control via the driver to the TCP. When the activity ends, the TCP finds that the line is in diagnostic mode, so instead of itself analyzing the ending status, it again passes control to the diagnostic driver. In this way, the diagnostic routines are executed in the ASP environment using the multiple wait facilities built into the TCP and STRAM.

The diagnostic driver analyzes each channel-end interruption and records the results in a line control block (LCB), keeping account of successes and failures as the test progresses. (The LCB is an appendage to the data control block, DCB,[5] for the STR adapters. All of these LCB's and DCB's are in the TCP.)

The STRAM error recovery procedures are inhibited on the line being tested during diagnostic mode, so that every fault is counted directly by the driver module. After analyzing the channel-end status, the driver transfers control to the test routine again to initiate a new input or output activity.

The diagnostic driver can control any number of diagnostic routines on different adapters, and the test modules are reentrant so that the same test can be carried out on two adapters simultaneously.

The diagnostic tests themselves are designed to investigate the various parts and functions of the data transmission system as follows:

- Data adapter unit, by executing commands in the adapter's test mode.
- Synchronization of the line, by repeatedly sending single data messages and re-establishing synchronization between each.
- Data transmission ability of the line, by continuously transmitting data between the SYSTEM/360 and the remote processor in either direction. The data transmitted can be chosen by the operator, so that suspected data bit patterns can be tested.

- Local Modem (or Data Set) attached to the SYSTEM/360 data adapter unit. This is done by manually looping the line from the Modem back via a second Modem and a second data adapter into the SYSTEM/360 and running communication diagnostic tests on the two adapters.

The diagnostic driver keeps the results of the test as it progresses, and, on its completion, sends a diagnostic results message to the operator who originally requested the test (the ASP teleprocessing console operator or the remote user). The driver then indicates to the TCP that the test is finished, and the TCP resets its flags and resets both interfaces of the adapter just tested back to their original modes.

remote processor usage

Because the TCP can send data continuously to the remote processor, which itself can at any time request to send data to the SYSTEM/360, use of the remote processor console keyboard and typewriter must be carefully controlled. To avoid conflict between the remote user (keying in data on his console) and the data management DSP's (sending data to be typed on the console), the user must request to use the keyboard before he can begin keying. This request (achieved by pressing the interruption request key) is transmitted to the TCP as a particular data message — TX. In response to this message, the TCP sets flags so that the data management DSP's are not able to send data to the remote processor console, and it indicates to the remote user to begin keying in data by sending him a reply of six successive question marks. This reply is typed on the remote console typewriter. Any data waiting to be sent to the remote processor console precede the question marks.

While the remote user is keying in data, the TCP continues to send data (either dummies or data to be printed on the remote line printer). When the user has either typed 63 characters or hit the end-of-field (EOF) key, a line turnaround occurs, and the TCP receives a data block from the remote processor. If EOF is pressed, an indication is given in the data block to the TCP, which causes the TCP to reset flags to allow data to be sent to the remote processor console again and to prevent the remote user from keying in more data until a reply is sent to the messages just received by the TCP. The remote user may also press the interruption request key after EOF if he desires to key in a new data message immediately. In this case, a second data block is sent to the SYSTEM/360—again the characters TX. Thus, the TCP receives two data blocks. On receipt of the first, the TCP sets flags to allow output to the console and passes the data to a data management DSP; the second is a request to use the remote keyboard again. The TCP remembers this request but does not immediately reply with the question mark sequence; this message is, in fact, attached to the last reply message from the data management DSP that has just received the remote user's data. In this way, the remote user can continuously communicate with the data management DSP's without unnecessary delay due to turnaround.

During the operations described above, the TCP continues sending data to the second remote processor attached to the same adapter (assuming that terminal is in data communication mode).

input/output interface

We have described the basic TCP operations and mentioned data being passed to and from data management DSP's. This communication is provided through macroinstructions. Two macroinstructions, CSPUT and CSGET, pass the data and indicate in parameters both the line number of the remote processor to which the data is to be sent or from which it is to be received and the address of the data input or output area.

The output data to the remote processor consists of line records, each being a two-character count followed by a format character and then by the data for that line of print (thus, each line record can be from 4 to 123 characters long). The format character determines which device the data is printed on at the remote processor (i.e., console typewriter or line printer) and also specifies the carriage control for the line printer, if applicable. These line records are blocked into ASP buffers by the data management DSP's, and each CSPUT macroinstruction causes the data of one such buffer to be sent to the TCP. The TCP, in turn, splits the buffer contents into a maximum of four data blocks (each a maximum of 203 characters) for actual data transmission. The input/output interface module determines these data block lengths and translates the data to the STR four-of-eight transmission code. The exact length of each data block is determined by the form of the line records in the buffer, since certain overflow conventions between message data blocks must be followed. The format character also indicates for console printer data whether or not this is the last data (i.e., may the user key in data after this data has been sent to the remote processor?).

The input data received from the remote processor can be the special message TX, which is dealt with by the TCP, or it can be a data block. If it is a data block, it consists of a format character, which indicates whether the user has pressed the end-of-field key or not, followed by 62 data characters. The TCP input/output interface translates the data to extended binary coded decimal interchange code (EBCDIC) from the transmission code and then passes it to the data management DSP for the line on which the data has been received. The format character is tested and flags set accordingly, to allow output to the remote console (EOF pressed) or to prevent it.

data management DSP's

There are two types of data management DSP's. The first is for data preparation (conversational mode) and the second for remote printing (remote job entry mode). A data preparation DSP is automatically scheduled for a particular remote processor when the TCP receives the first data block from that terminal. Execution of a remote printing DSP can be scheduled either by a request from a data preparation DSP or by ASP itself when job output from the main processor is to be transmitted to one or more remote processors.

The data preparation DSP's accept input data from the remote processor and operate according to the control verbs in the data. The main functions of these DSP's are to create, edit, and update application (job control) and data files on the support processor prior to job execution on the main processor. The files are created as ASP files with control information (such as location and status) kept on reserved disk storage areas. Checkpoints are regularly made of these reserved areas, and they are protected against all but the most serious ASP breakdowns. In the control area, there is a file table for each remote processor's data, which enables a particular terminal to prepare its own data and also to have access to data from other terminals. Thus, the jobs initiated from a terminal can include multiple input data files and historical data files without the normal remote job entry requirement of transmitting all that data from the terminal itself.

The data preparation DSP's also have extensive facilities for checking the format of input data before initiating a main processor job and for editing the data to correct any errors found. The remote user is thus ensured that when he initiates his job by a control message, it will have a better chance of running successfully on the main processor—saving expensive and unnecessary delay in job turnaround due to data errors.

The data preparation DSP's are also used for message switching between the remote processors and the ASP console operators. The sending of messages to single or multiple terminals and/or consoles can be initiated, and the messages are queued by the data management DSP's until they can be transmitted.

The remote printing DSP's establish the format of and transmit output data from the main processor to the remote terminals. These programs create the line records discussed earlier and block them into ASP buffers for passing to the TCP. Additionally, these DSP's can be used to transmit the contents of data files to the remote processors, so that a user can inspect the contents of any of his files by initiating a request for a remote print DSP to be activated and specifying the required file.

error detection and recovery

The facilities included in the data transmission system for automatic error recovery and for fault detection can be summarized as follows:

- Error recovery extensions to the STRAM error routines to attempt to recover from apparently permanent errors without operator intervention.
- Statistics gathering to indicate automatically when the error rate on a line increases above an installation standard (e.g., this standard is currently three data checks in 10^5 bits transmitted, which is a relatively high error rate in normal circumstances). Statistics also provide a long-term indication of the performance of a line.
- Diagnostic tests, mentioned previously, that can be run to attempt to isolate a particular device as the cause of faults

in the system. These tests can also be used as a confidence check when faults have been corrected.

- Line interrupt traces, which are always available, so that a fault can be analyzed each time it occurs. These traces are also available for analysis following running of diagnostic tests if the test results themselves are not conclusive.

These facilities are all available under the ASP system and do not interrupt the normal operations of any part of the data transmission system except when a particular adapter is in diagnostic mode. When one interface is being tested, the other interface is automatically set into an idle state until the test has been completed, at which time it is reset to its previous operating mode.

errors in data communication mode

Errors due to equipment or line faults are detected by the STRAM error routines. These routines automatically initiate error recovery procedures in an attempt to continue normal operations. The recovery procedure used depends on the type of error that has occurred. For example, if the remote processor indicates that it has not received a data block correctly (error reply), that data block is retransmitted; if the remote processor fails to reply to a data block after a predetermined interval, the line is tested for synchronization and an inquiry sequence is sent to interrogate the remote processor again.

When the STRAM error recovery procedures have completely failed to clear the error condition after the specified number of attempts, an indication is passed to the TCP that a permanent error has occurred. The TCP error routines then analyze the type of error and determine what procedure is to be followed.

In data communication mode, certain types of unusual conditions and errors are expected. An inquiry sequence in reply to a data block is an indication that the remote processor is waiting to transmit data. Thus, the TCP ignores this unusual condition and attempts to receive the data.

If the error is due to the remote processor sending error sequences in reply to a data block, the TCP extends the STRAM error recovery procedures. It has been found that about ninety percent of these errors are due to transient line faults. By delaying for a short time before retransmitting the data, the error can often be cleared. To do this, the TCP error routines send a sequence to the remote processor that causes a delay of several seconds and also indicates to the terminal that extended error recovery is being attempted. The TCP then breaks the line, re-establishes synchronization, and repeats the data block in which the error occurred. If another error occurs before the current ASP buffer of data (a maximum of four data blocks) has been cleared, a TCP permanent error procedure is entered. This procedure is also used for errors not of the error reply type. A message is sent to the ASP console operator indicating the line and type of error that has occurred, and the TCP sets an entry in a table, which is checked

if the operator attempts a retry by entering a TCP command (this procedure is discussed later in the paper).

In addition to informing the console operator, the TCP tries to indicate to the remote processor that an error has occurred by sending an EOT sequence. When the remote processor receives this sequence, this processor knows that any retry will result in the transmission of all of the data blocks that were in the ASP buffer when the error occurred, and it therefore replies with its own EOT sequence. If the remote processor does respond with an EOT sequence, STRAM again indicates this condition as a permanent error to the TCP, but the TCP ignores this.

These procedures to inform the remote processor of error conditions and to indicate the type of recovery to follow are nonstandard STR operations that have been added to the system to reduce operator responsibility to a minimum. Retrying a TCP permanent error results in repeating the operations that caused the original error. If a further permanent error occurs before any successful data transmission, the TCP closes the line and again informs the operator, who can only restart data communication on that line by first setting it to the initialization mode. If the attempt to retry data transmission after an error is successful, the data communication mode is resumed as normal.

statistics and line interrupt traces

The appendage to the STRAM routine for handling channel-end and/or abnormal-end conditions and the STRAM first-level error routine have been modified for the TCP. An exit has been added to each so that a TCP module, which passes information to the line-interruption trace and statistics-gathering routines, is entered every time a channel-end interruption occurs from an STR terminal adapter unit (i.e., every time an operation is completed on any line). This module is thus entered during execution of the operating system input/output supervisor and has access to the status information for the unit on which the interruption occurred. The module updates line statistics in a line control block (LCB) associated with the adapter. These statistics are the number of data characters transmitted over the line and the number of errors requiring retransmission.

Each time an interruption occurs, the format of a line interruption trace (LIT) is established. The LIT, which has an interruption number to identify it, contains the following information:

- *Event completion block*, indicating status of I/O event.
- *Input/output block*, the OS/360 I/O control block for this adapter.
- *Line control block*, the STRAM I/O control block for this adapter.
- *Channel command word chain*, the channel program being executed.
- *Line control block appendage*, diagnostic flags, counts, and statistics.
- *Line control block user area*, flags that indicate the current operating mode and pointers to data to be transmitted or just received.

This information is passed to another DSP that creates the format of, stores, and retrieves LIT's. The LIT is processed according to flags in the LCB appendage. These flags initially indicate that only LIT's corresponding to error conditions are stored on disk storage. However, the flags can be changed for any adapter by using the ALTER command and can be set so that all LIT's are stored on disk or that LIT's are printed on a line printer as they occur. This DSP also retains the most recently stored LIT's for each adapter (about 120 for each) in ASP files on disk storage. These can be printed by the teleprocessing console operator by entering an ASP start command indicating which adapter's LIT's are required. The LIT's are printed asynchronously (SPOOLed)[6] by a local printer.

The statistics mentioned above are updated in the LCB appendage in main storage. They are then periodically (every few minutes) added to a statistics record on disk storage. This record is at a fixed disk location and is protected from all but the most serious ASP breakdowns. The record on disk has three levels of statistics. The first level, updated every few minutes, is checked automatically every hour. Any lines with a high error rate are indicated to the ASP teleprocessing operator by messages, and the statistics are then added to the second-level figures. The console operator can have these second-level figures listed by entering the TCP command STATS. They are also available to the installation accounting routines, giving an indication of the remote processor usage of the system. The third level of statistics is updated each time the second level is cleared and again can be listed by the operator.

TCP support of local terminals

When the system using remote processors over STR lines was working satisfactorily, the TCP was extended to support local typewriter terminals (the IBM 2740). Obviously, since these terminals are basically typewriters, the remote print capability is not available, but, in all other respects, they provide all of the same facilities to the user at a terminal such as the IBM 2740 as though it were the keyboard of the IBM 1130 console. In addition, the local user can overcome his lack of a remote print-type facility by directing his job output of file listings onto one of the ASP support line printers at the central computer location or to any of the attached 1130 computers.

As was mentioned earlier, the TCP has been developed on modular lines, so that incorporation of the new type of terminals was simple. Additional routines to control the input/output of the 2740's were incorporated into the TCP using the same basic modules that already existed in the ASP system for supporting multiple operator consoles. Thus, for data transmission to a local 2740 terminal, the TCP interface to the 2740 support routines of ASP is similar to the TCP interface to STRAM for data transmission to a remote 1130 over an STR line.

The local terminal user has exactly the same status as the remote users and can communicate with them via the TCP. The

only operational difference is that the local terminals are always in data communication mode and are available to the local user all the time that the TCP and ASP systems are active.

The local terminal support, in addition to giving new users access to the system, also provides an alternative means of entering work from a remote user in the event of an extended breakdown of his data link or remote processor. (All the files created by a remote user can be made available to the local user.) This alternative access to the system becomes more important as the use of the system increases, and the ease with which it was provided is an indication of the importance of the initial design philosophy of the teleprocessing system, to allow new and various types of access terminals.

Terminal program

The design objectives for the terminal program are to:

- Allow a small amount of data input from a keyboard.
- Allow a larger amount of printed results, to be completed as soon as possible after the data has been keyed in.
- Provide inquiry facilities from the terminals, with as fast of a response as possible, but in any case within the limits of operator frustration.

From these objectives, the need for a central processing unit at the terminal is not obvious. However, the output printing rate required use of a line printer, and the 1130 is a successful competitor with other equipment. The 1130 configuration, with 4K bytes of main storage, includes an IBM 1132 line printer and an IBM 1134 paper-tape reader (for the program loader only).

A line speed of 2,000 baud, combined with 2701's at the SYSTEM/360 end, and use of the dual communication interface seemed to be the most economical method of servicing the terminals. This decision took into account the idea that the high line speed should be sufficient to keep one 1130 in operation for the time it took to switch the interface, send data to the second 1130, and switch back. However, this brought to the forefront an immediate problem at the 1130—the limited main storage size of 4K bytes.

The required programming support to control the communications adapter, printer, and console/keyboard amounts to about 3K bytes, leaving only 1K bytes for the control program and input buffers and for the print buffers. The paper-tape reader is only used for control program input, and the control program resides in main storage until power is turned off on the 1130.

The second problem concerned the conversational requirement. The STR line control method does not provide for this except by using the end-of-transmission sequence followed by an agreed turnaround in data transmission direction. However, this was ruled out by the fact that the 1130 programming support

requires about 10 seconds to process an end-of-transmission sequence, which is out of the question.

Therefore, it was decided to implement an unusual line control sequence in order to initiate a line turnaround in the conversational mode. An inquiry sequence (TL-INQ) is sent in reply to receipt of an error-free data message from the SYSTEM/360, instead of an odd or even acknowledge sequence. The 1130 then transmits a second inquiry sequence, the data, and an end-of-transmission sequence. After this, transmission reverts to normal procedure. The complete line control procedure is shown in Table 2 for two 1130's connected to one 2701 with the dual communications interface feature. Note that no changes have been made to 1130 basic programming support.

The programming support for the 1130 (SCAT1) need not be altered to send the inquiry sequence in response to data instead of the conventional acknowledge sequence. An indicator bit in SCAT1, which would normally prevent sending the inquiry sequence, is turned off by the terminal program. Thus, there is no loss of time in regaining synchronization.

To print output at an 1130 terminal, 400 characters are sent in one cycle. These 400 characters are organized into any number of lines, each line preceded by a two-hexadecimal count and by a format character, which may indicate skip to channel 1, print on console printer, etc.

The organization of the program is such that the contents of one buffer, 200 characters, is received. As the first 200 characters are being processed and printed, a second 200 characters are received. As soon as all of the second 200 characters have been received, if no message is waiting to be sent from the 1130, a SCAT1 acknowledge and receive function is initiated, followed by a 1/2-second loop, to allow the acknowledge sequence to be sent. Then a SCAT1 CLOSE function is performed, followed by a SCAT1 "open data-in" function to synchronize and accept an inquiry on the next cycle.

The 1130 operator is not allowed to key in unless he has first inquired from the SYSTEM/360, by pressing the interrupt request key, whether there are any console messages queued. The interruption request key sets up the program to send the message TX to the SYSTEM/360. If there are no console messages to be sent to the terminal, the SYSTEM/360 sends a line of data, possibly blanks, with a format character K. On receipt of this line, the terminal program unlocks the keyboard to allow the typing of data. When the EOF key is pressed, the data is sent to the SYSTEM/360.

terminal language

The terminal language is designed for comprehensive conversational file manipulation from the terminal keyboard. Any character can be specified, through one of the commands, to be a backspace character. Information is typed as a series of messages of one or more lines, terminated by the special end-of-message key (EOM). Blank characters or commas are used as separators.

Table 2 Line control with SYSTEM/360 - 2701 connected to two IBM 1130's

SYSTEM/360		IBM 1130	*Comments*
Obtain synchronization with 1130-A			SYSTEM/360 has data to send to 1130-A. At this point 1130-A has data to send to SYSTEM/360, but must wait for second buffer before sending inquiry sequence.
Inquiry sequence	→		
	←	Acknowledge sequence	
Data record 1	→		
	←	Acknowledge sequence	
Data record 2	→		
	←	Inquiry sequence from 1130-A	1130 has data to send. SYSTEM/360 recognizes non-standard sequence.
	←	Inquiry sequence	
Acknowledge sequence	→		
	←	Data record 1	
Acknowledge sequence	→		
	←	End-of-transmission sequence	1 second only to complete processing this EOT.
End-of-transmission sequence	→		
Obtain synchronization with 1130-B			Due to dual communication interface, 1130 at interface B must now be serviced.
Inquiry sequence	→		
	←	Acknowledge sequence	
Data record 1	→		
	←	Acknowledge sequence	
Data record 2	→		
	←	Acknowledge sequence	
Obtain synchronization with 1130-A, etc.			1130-B has no data to send to SYSTEM/360.

After end-of-message, the user's keyboard is locked until the system has typed a reply acknowledging that the previous message has been received and processed. Control messages begin with the character # and can be abbreviated to a single character (the first character of the command). Anything not beginning with the character # is treated as data for a current file that the user is manipulating. The control words presently used are shown in Table 3.

terminal extension

Although the capabilities of the terminal program for conversational data entry and high-speed printed output are discussed here, the communication techniques with the TCP allow for extension of the terminal program. With additional equipment, such as card readers, high-speed remote job entry could easily be incorporated, while preserving the conversational data-entry techniques. Also, the terminal techniques are not restricted to the 1130; they could be applied to any terminal that can communicate with the SYSTEM/360 using the STR method (and later using BSC). A further extension could allow the remote terminal to act as concentrator for several input/output devices, such as an additional keyboard or card reader and more printers attached at another level.

Nonprogrammed terminals, such as the IBM 2740 and 2780, would communicate directly with the TCP at the central SYSTEM/360.

Table 3 Control words

Control word	*Function*
USER	Identifies user at start of session
NEW	Identifies a new file
HOLD	Preserves a file after use (which otherwise would be automatically deleted)
DELETE	Deletes a file immediately
EDIT	Specifies editing, such as deleting, replacing, or inserting lines
CHECK	Transfers control to a specified routine, such as a compiler prescan routine, to perform application-dependent data checking
GO	Enters file into job stream for normal processing as a job
TYPE	Types all or part of specified file at specified terminal (not necessarily user's terminal)
PRINT	Prints all or part of specified file at a specified terminal
INCLUDE	Incorporates a specified file into current file
MESSAGE	Sends a message to any other terminal
QUERY	Provides information on all files held at central system
REFER	Makes file referred to the current file, for file manipulation
FINISH	Closes file and logs off current user
ARITH	Initiates a conversational desk calculating function

Summary

The techniques described have all been implemented and are working well in an on-line environment assisting control of a vital national facility. The teleprocessing design is not just a technique to combine low-speed and high-speed terminals in one general package; it also provides a basis for development of techniques to provide real-time control facilities on standard general-purpose equipment and support programs. This basis is achieved by the automatic, on-line philosophy of the teleprocessing control program. This program is designed to run 24 hours a day, 7 days a week with minimal operator intervention. No external contact between operators of the central system or the terminal is needed.

The foundation is laid for the next development, that of entering direct monitoring signals into the TCP and initiating

appropriate calculations or responses. The multiprocessing control by ASP fulfills the reliability consideration of such a system, and it is feasible to build on this to provide automatic reconfiguration after failure, which is a requirement of real-time control. Already, the system described here is capable of reinitializing and restarting itself from an intermittent equipment or program failure, and is running in the environment of a normal batch job processing system.

ACKNOWLEDGMENT

Development of these techniques was contributed to by Mr. B. E. Carrell, SDD Teleprocessing Planner for IBM United Kingdom. The data handling routines and terminal language were largely developed by Mr. F. S. G. Richards and Mr. M. Wall of the Central Electricity Generating Board.

CITED REFERENCES AND FOOTNOTES

1. F. S. G. Richards and J. Hewson, "Real time aspects of communications and computing techniques in the security and control of electricity supply systems," *Proceedings of the Second IFIP/IFAC Conference*, Menton, France (June 1967).
2. J. L. Eisenbies, "Conventions for digital data communication link design," *IBM Systems Journal* **6**, No. 4, 267–302 (1967).
3. IBM SYSTEM/360 *Attached Support Processor System* (ASP) *System Description*, H20-0223, International Business Machines Corporation, Data Processing Division, White Plains, New York.
4. *Synchronous Transmit-Receive Access Method: Application Description*, H20-0242, International Business Machines Corporation, Data Processing Division, White Plains, New York.
5. W. A. Clark, "The functional structure of OS/360, Part III, Data management," *IBM Systems Journal* **5**, No. 1, 30–51 (1966).
6. SPOOL signifies Simultaneous Peripheral Operations On Line.