The design of real-time commercial data processing systems includes special computers serving as communications control devices.

These special purpose stored-program "data exchange" computers, used in conjunction with standard data processors, permit significant simplification of the system's programs as well as an increase in the overall system efficiency.

This paper considers the new concepts involved in programming a particular data exchange computer.

Programming considerations for the 7750

by Nicholas Sternad

In the earlier design of real-time commercial data processing systems, efforts were made to use existing electronic computers, essentially without modification. Simple intermediate buffering devices, capable of serializing, deserializing and temporarily storing one or a few characters, were designed to link the communications channels and the central computer. This tended to overtax the computer and decrease its data processing efficiency. Furthermore, it created an extensive set of difficult programming problems.

real-time programming problems

Among the new problems to appear were the assignment of individual input, output and work areas to a large number of communications channels and the allocation of memory space on a flexible and dynamic basis. Other problems involved code conversion, editing, character validity checking and the generation of check characters to establish the validity of input. The operation of the network had to be monitored with communications channels allocated to terminals and the awareness status of the network constantly checked. Detection and indication of environmental error conditions were necessary. In addition, problems occurred relative to continuous operation (to circumvent normally stop-causing errors), response time, memory protection, loop protection (against endless loops) and periodic checking of buffer area capacity.

The analysis of these real-time problems led to the conclusion that the best solution would be to design a "data exchange"

development of data exchange computers

computer—a special purpose, communications-oriented, stored program computer—for external attachment to existing data processors. Thus the central computer could be relieved of most functions having no direct bearing on the actual data processing. Input data would be assembled and edited by the data exchange before entering the central computer and output data transferred from the central computer for further preparation and transmission to destinations in accordance with its directions. The application of such a machine would also facilitate message switching.

The IBM® 7750 Programmed Transmission Control (PTC) was designed with the above problems in mind. As would be expected, some novel programming concepts are involved in using the machine. Since the purpose of this paper is to consider the methods employed in programming the PTC, a brief introduction to its logical organization and operation is given before proceeding.

Logical organization and operation

basic functions

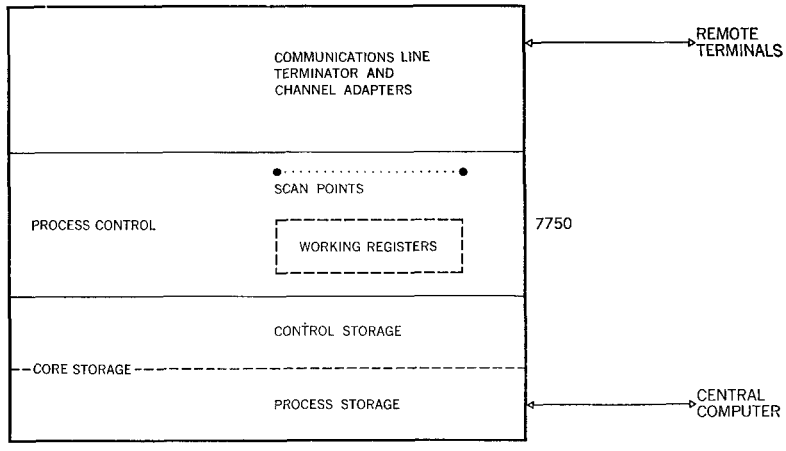
The PTC is a physical and logical link between a communications network and a central processor. It accepts information serially, bit-by-bit from transmission lines, assembles the bits into characters and after performing the processing functions required to complete input messages, transfers the prepared data to the central processor. Output messages, received from the processor, are handled in a reverse manner.

The machine, a binary stored program computer, is shown schematically in Figure 1 together with its functional relation within a system.

control and process storages

The PTC has two core memories—*control storage* (128 words) and *process storage* (4096, 8192, or 16,384 words). Words in both core memories have several formats, each designed for a specific purpose. Regardless of the format, a word consists of 48 bits, 47 bits for information and a parity bit.

Figure 1 Simplified schematic of the 7750 and its relation within a system



Both storages are operated on a 28 microsecond machine cycle. The control storage and process storage cycles overlap each other and the two units operate in synchronism.

Words controlling the input-output operations and environmental error conditions, the execution of the stored program, and data transfer to and from the processor are placed in control storage. Process storage contains instructions, constants, tables and data.

Control storage words are functionally defined words with predetermined addresses. Process storage words are not restricted as to their location, function or format. Hence, the programmer has control of word assignments in process storage.

blocks and chains

Process storage is divided into blocks of eight words to facilitate flexible memory utilization for data storage. Each word contains four 11-bit-character fields; a block consists of 32 character positions. The last (32nd) character of each block is the *block control character* and is used for recording the address of another block which contains the next sequential portion of the data complex. If two or more blocks are linked together by this method, a "chain" is created.

When a block control character is accessed, either during automatic input-output operations or during data manipulation, the machine logic automatically indicates an *end of block* condition. The stored program must allow for the occurrence of this condition.

In process storage areas used for other than data manipulation, the block control character is used for holding information because block organization is unnecessary.

automatic functions

The 7750 performs automatic and programmed functions. The *automatic functions* are:

Bit-by-bit character assembly and distribution in control storage • Character by character message assembly in, and message distribution from, process storage • Detection of end of block condition during character assembly, character distribution and character by character data processing operations • Transfer of data to and from the processor • Detection and indication of communications channel errors and internal machine parity errors • Recognition of special characters in the character assembly area of the input-output controlling words and execution of automatic functions associated with the detected special characters • Priority request when certain conditions are detected (in some cases, priority request is originated by the program) • Constant checking of priority request status • Change of the operational mode of the machine in accordance with the result of the priority status check.

programmed functions

The *programmed functions* performed by the PTC are as follows:

Preparation of input data for the CPU[1] and output data for terminals with the most common functions being code conversion,

editing (i.e., deleting and inserting functional characters, in some cases changing format), character validity checking, generating check characters and comparing them with input check characters, character count keeping, sequence checking, and message numbering • Dynamic memory allocation for input messages and in all cases where memory area is needed throughout the operation • Output area identification for automatic output operations • Monitoring the input-output operation of the network • Checking the awareness status of the network • Preparation and initiation of data transfer to and from the processor • Checking the validity and completeness of a terminated data transfer to and from the processor and signalling the processor according to the result of the checking • Identification of automatically detected and indicated communications channel and internal parity errors • Testing and isolating faulty network components (terminals, channels) • Temporarily re-routing traffic for isolated faulty terminals and communications channels • Error routines for all phases of the system operation.

parallel operation

The 7750 operates in a parallel manner: while the bit-by-bit character assembly and distribution automatically progress in control storage, the stored program is executed or data transfer is made to or from the processor. A one-cycle "character interrupt" is utilized to store an assembled input character in process storage or to obtain the next output character from process storage.

operational modes

There are six operational modes. The PTC may, at any time, operate in only one mode. A priority scheme exists among the six *operational modes* in the following descending order: SERVICE MODE, CHANNEL SERVICE MODE, COPY MODE, OUT MODE, IN MODE, NORMAL MODE. The NORMAL MODE has no priority, the other five modes are referred to as *priority modes*. Five of the operational modes require programs and only one, the COPY MODE, is a program-independent, fully automatic procedure. The purpose of the COPY MODE is to transfer data to or from the CPU. The purpose of each programmed mode is described later in the section devoted to programming.

An independent control word for each of the six operational modes is located in control storage. The instruction counter and a set of operational data registers for each of the five programmed modes are located in its unique control word. Hence, no programmed means are necessary to save the contents of these registers or the instruction counters when the operation is changed from one mode to another.

mode selection

The mode in which the PTC operates at any time is determined automatically. The PTC operates in the NORMAL MODE when no priority mode is requested, otherwise in the highest requested priority mode. This method is a multi-level priority recognition procedure. All priority requests are recorded in a register either by programming or automatically and terminated in the same register. The current operational mode may be changed either to a higher (but not from the SERVICE MODE which is the highest)

or to a lower mode (but not from the NORMAL MODE which has no priority). Any mode operation is interrupted if a higher mode is requested. However, the request for the current mode is not terminated in the register because the progressing operation was not complete at the time of mode change. Therefore, as soon as no request exists for higher modes, the operation will again return to the just interrupted mode. Change to a lower mode of operation in order of their priority occurs after completing the operation in a higher mode and terminating the associated request.

The programmer may prevent mode change at any place in a program by simply tagging those instructions after which such a change would be undesirable. This method is used to avoid interference by another routine in a higher priority mode before an operation is completed.

Programming considerations

division by function

The functions to be programmed are divided into three separate categories:

1. Functions to be assigned to the CPU because of its superior processing capability or its access to files needed in processing.
2. Functions to be performed by the PTC determined by its unique logical design.
3. Functions that may be programmed in either unit.

Division between the first two categories will be apparent. In selecting the third group, factors to be given prime consideration are memory space utilization, time utilization, internal speed, through-put capability, operation and system efficiency, and maximum flexibility for system growth *rather than* programming convenience. Since in this phase of the system study, allocation decisions are usually based on estimates, functions may be transferred from one unit to another at a later time. Specifically, this may take place after completion of preliminary system specifications, when analyzed simulation results or other considerations indicate the possibilities for system improvements. These changes, of course, would make it necessary to modify the system specifications.

program design

Programming design for the PTC is determined by the functions allocated to it, the detailed requirements of each, and by giving particular attention to the automatic features of the machine.

The basic task is the distribution of programmable functions among the five programmed modes. The priority mode programs are intended to satisfy special conditions and should perform corresponding functions. Nevertheless, the programmer is not completely restricted by binding rules. If he can devise a safe and efficient means to do so, he may take advantage of the higher priority rank of some modes and allocate important functions to them other than the ones for which they were originally intended.

For example, output messages received from the processor

might be prepared for terminals in the IN MODE, instead of the NORMAL MODE. This would simplify the preparation of data receptions from the processor by assigning the same physical storage area for each such transfer. It would further check the received data for validity before the transferred message is released by the CPU program, providing for an immediate retransfer if error is found.

On the other hand, in order to shorten the execution time of some priority mode programs, a limited amount of preparatory work, or the completion of certain operations, may be incorporated in the NORMAL MODE program. To illustrate, consider the preferred procedure for environmental errors: such errors are identified by the SERVICE MODE program, but the error routines are programmed and executed in NORMAL MODE upon the initiation of the SERVICE MODE program, in order not to keep the operation unnecessarily in the highest priority mode and block the execution of other requested priority modes.

priority mode functions

The first phase of the PTC program design must be concentrated around those functions that imperatively call for assignment to priority modes because the machine logic automatically demands the execution of the priority mode programs when the associated conditions are recognized. These functions are by modes:

Service mode. Identification of automatically detected and indicated communications channel errors and internal parity errors by the program, and initiation of the execution of error routines when necessary. The error routines are generally programmed in NORMAL MODE.

Channel service mode. Dynamic memory allocation for input data and identification of the next character address of output data, when crossing block boundaries in a chain.

Out mode. Preceding COPY MODE operation: preparation of the copy word[2] for data transfer to the processor by setting a start address and a stop address to determine a chain of any number of random blocks containing output data for the processor; provision for time-out termination of the COPY MODE operation in case a machine failure stops the transfer before one of the means of legitimate termination; and the request COPY MODE operation. Following COPY MODE operation: check whether or not the terminated COPY MODE operation produced a complete and valid transfer and signal the processor according to the result.

In mode. Preceding COPY MODE operation: preparation of the copy word for data transfer from the processor by setting a start address and a stop address to determine a chain of any number of blocks designated as input area for data expected from the processor; provision for time-out termination; and the request COPY MODE operation. Following COPY MODE operation: check the validity and completion of the terminated transfer and signal the computer according to the result.

The next step is to allocate further functions to priority modes in order to increase system efficiency, to utilize automatic features provided by the circuity, or to satisfy special conditions requested by the user.

An example of satisfying special conditions might be the completion of pre-arranged, inquiry type messages in the OUT MODE upon request from certain predetermined terminals, interrupting the operation of the CPU, and upon response, transferring the inquiry message to the CPU. When the answer message is received, it is prepared for the terminal in the IN MODE and placed in front of the regular output queue for that channel. By this method, messages can be exchanged on a priority basis between terminals and the CPU.

All required functions not allocated to priority mode programs are programmed in NORMAL MODE. Although it has no priority, it may be called the main program of the PTC because the vast majority of programmed functions are implemented in this program. Its execution is the least dependent on environmental circumstances.

normal mode functions

The functions to be programmed in the NORMAL MODE may vary from application to application. The most common functions are:

Message header processing • Message accounting; numbering of output messages, sequence checking of input message numbers • Code conversion • Character validity checking • Generation and comparison of redundancy check characters • Editing; adding functional characters to output, deleting such characters from input messages; message format change if required • Queueing messages in output areas assigned to communications channels and to the processor • Transmitting acknowledgment messages or characters • Polling; allocating communications channels to terminals for incoming transmission • Initiating incoming and outgoing transmission • Count-keeping for required purposes • Removing blocks from and returning them to the available buffer pool • Checking the available memory area for a predetermined minimum number of blocks • Checking the awareness of the network • Keeping status tables for channels and terminals • On-line diagnostic testing of malfunctioning network components • Isolating faulty terminals and channels, re-routing traffic for such components • Error routines for both data and component errors • Communications with operating personnel through "on the installation" terminals • Initiating communication with the CPU by generating an attention signal.

program organization

After the allocation of functions is completed, the programs must be organized and separately written for each of the five programmed modes. They must be completely separated because the linkage among them is provided by the machine-facilitated automatic priority processing method. Although the program has access to the mode request register[3] and may initiate

a request for any priority mode, only a few special applications would justify the requesting of a programmed priority mode by the program (the COPY MODE is not a programmed mode). The "automatic priority recognition" feature of the PTC is designed to comply with the requirements of the real-time environment in which the system is operating and the associated conditions are detected by the machine logic. Consequently, all four programmed priority modes are normally requested by the machine logic upon detection of associated predetermined conditions.

The organization of each program must satisfy the requirements of the real-time environment in which the system will operate. This principle has little effect on the priority mode programs, because generally each has to be divided into logical functions with separate program routines written for them. The entry from one routine to an associated one at logical decision points is normally accomplished either by a straightforward entry or by a branch instruction, depending on the prevailing condition. However, the OUT and IN MODE programs assist, as described below, in programming more than one logical function in a physically consecutive routine without programmed linkage. The preparatory work preceding the data transfer is programmed in a routine ending with an instruction which requests the COPY MODE, and the immediately following instruction is the first instruction of the routine which checks out the results of the terminated transfer. The reason linkage needs not be programmed is that the automatic termination of the COPY MODE causes return of control to the OUT or IN MODE, from whichever the COPY MODE was entered. Since the instruction counter for the OUT or IN MODE was incremented when the COPY MODE requesting instruction was executed, upon return of control, the first instruction of the check-out routine will be executed.

The major part of programming is done in the NORMAL MODE. The organization of this program will largely determine the efficiency of the entire system. The prime function of the PTC is to serve all communications channels simultaneously. In order to maintain a well balanced constant flow of data, the program written for controlling the input-output operations and the associated data preparation must be well coordinated with the machine functions, thereby providing a means for simultaneous input-output operations. The communications channels are automatically scanned on a preset schedule which provides for simultaneous character assembly and distribution operations for all channels. The NORMAL program must likewise work on a "scan basis," examining the status of the existing conditions for each communications channel on a predetermined schedule and executing the instructions necessary to satisfy the detected conditions.

The PTC processes data character by character. In order to follow through the principle introduced above, the program does not delay the processing until a complete message is stored in the

input area assigned to a communications channel. Instead, it proceeds from area to area in a predetermined sequence and processes the input characters in each. Finally, it returns to the first area to process the new characters (if any) stored there since the program has left it. At this point, the program begins its next "scanning round" in order to process the recently stored characters in each of the input areas. In connection with this operation, the program also checks the output areas to initiate output operations whenever output messages are waiting for transmission and the outgoing line of a full-duplex channel is free, or the half-duplex channel becomes available for this purpose.

Another step in line with the principle of maximum coordination between the program and automatic machine input-output functions is to reduce the processing functions to a minimum level that may be accomplished within the time restrictions of input-output operations. Short program routines are then written for each of the "minimized" functions. This method adds flexibility in that the program becomes adjustable to the automatic channel scanning schedule.

The programmer must estimate how much processing can be allowed for servicing one communications channel without unnecessarily delaying service for others. It is also his task to stop the processing at certain logical points, even if it could be continued, in order to provide service for the channel next in the programmed scanning schedule. A good example is the detection of the end of message (end of transmission) character(s) in the input area of a half-duplex channel. Upon detection by the program, a few characters may have been processed already; therefore, certain time was spent for servicing the area. The program now has several functions to perform. The message must be moved to the output queue for the CPU, a block or blocks may have to be returned to the buffer pool, the output area of the channel has to be examined, and if an outgoing message is waiting, an outgoing transmission is to be initiated on the channel. If all these functions are performed consecutively on a single channel, an undesirable delay is caused in servicing the next and consequently the following channels. Therefore, the programmer may plan to execute only one of these routines—or even none. In the latter case, he may simply want to record in the related entry word of his processing control table the address of the end of message routine, proceed to service the next area and so on until his scan program completes a "round" and returns to service the referenced area. At this time, one or two of the due routines will be executed and the rest of the functions may be deferred until the next access of the same area.

It is obvious that separate routines should not be written within the same mode for identically processing data in areas assigned to different communications channels. The same processing functions must be covered by a single program routine for all areas. Only certain initializing instructions must be executed

whenever the routine is executed for a different area (i.e., the operational addresses of a few instructions must be modified to access data in the currently desired area).

When correctly organized, the NORMAL MODE program in its final format consists of numerous short routines linked together by some logical control. The method of organization is described in more detail below.

The program has no direct control over the changes of the operational modes, because the recognition of the highest requested priority (or no priority request) and the consequent change of the operational mode (if any) is automatic. Even if the program requests a priority mode, it is unpredictable whether or not the requested mode will be the next operational mode of the machine (SERVICE MODE, which has the highest priority, is not likely requested by the program). Similarly, program routines within a mode program may be called for at any time, depending on certain conditions. For these reasons, all mode programs and their routines must be retained in process storage throughout the operation of the system. Only programs of a special nature (e.g., periodically executed on-line diagnostic programs) can be stored outside the core memory and called for when required.

Two other important general organization objectives must be considered in organizing all programs:

1. The amount of core memory area occupied by programs, tables, and constants must be kept as low as possible, and at the same time the buffer storage area should be increased as much as possible.
2. The throughput of the PTC should be increased to the highest possible level under the existing environmental circumstances.

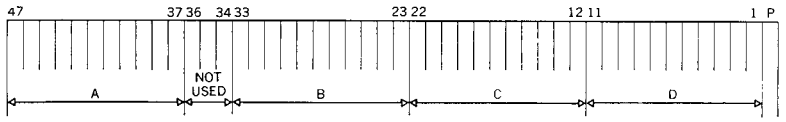
programming techniques

The efficiency of a PTC program depends on the combined utilization of the word formats, instruction set and automatic machine features. In order to illustrate the resultant programming techniques it will be necessary to examine certain word formats and the nature of the instruction set.

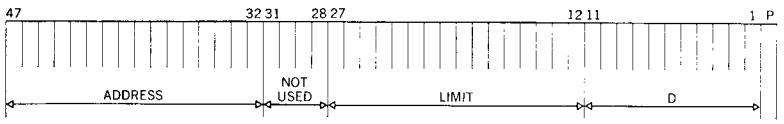
word formats

The formats of the data, limit and channel words are illustrated in Figure 2. *Data words*, which are located in process storage have four character fields which are identified as the *A*, *B*, *C* and *D* fields. *Limit words*, also located in process storage, have three fields; address, limit and *D* field. Normally, the content of the address field is the address of a process storage location; the limit and *D* fields may be used in different ways depending upon the application. *Channel words*, located in control storage, are the controlling words for input-output operations and certain environmental error conditions. These words contain three distinct areas; character assembly area, control area, and address. Characters are automatically assembled in, or distributed from the character assembly area bit-by-bit. The control information necessary for the machine logic, for the proper interpretation and for the execution of the automatic functions associated with

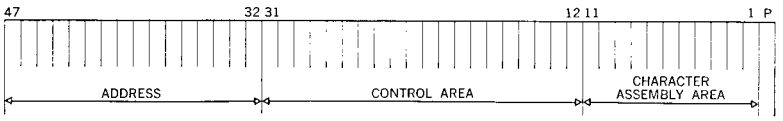
Figure 2 **Data word format**



Limit word format



Channel word format



input-output operations (excluding the data transfer to and from the CPU) are stored in various fields in the control area by the program. The address field contains the address of a character location in process storage where the assembled character is stored during input, or the next character is obtained during output operations. It is automatically incremented by one after a character is stored or obtained and then checked for end of block conditions. Upon detection of an end of block condition, CHANNEL SERVICE MODE is requested.

instruction set

The instruction set for the PTC includes basic instructions and their micro-code modified versions, allowing character, address and limit manipulations, storage to storage data transfers, unconditional and conditional branches, and address-limit comparison. The execution time of an instruction may be one machine cycle or two machine cycles, 28 or 56 microseconds, respectively. Data can be manipulated in original format or in "one's complement" version; the contents of address fields can be incremented; indirect addressing can be used with or without being followed by address increment. INCLUSIVE OR, EXCLUSIVE OR and AND logical operations can be performed.

Programs for the 7750 are written in symbolic language. The mnemonic operation codes have been constructed in such a manner that the first symbol of any mnemonic operation code has the same unmistakable functional meaning. The available assembly program[4] runs on the 1401 and produces output suitable for loading into the 7750 through the 1410 or 7000 series data processing systems. Although the format of each process storage word is determined by its function, the programmer may also operate on the contents of any word with instructions other than those designed for operating on the designated word formats. Operation on the content of a control storage word can be ac-

complished by first moving it to process storage. (Some instructions can manipulate control storage words in control storage.) In addition, the programmer can create new word formats by combining existing ones to suit his operations and save memory space.

A few examples will now be given to illustrate the nature of the PTC programming techniques.

application of limit words

All data flowing through the PTC is controlled through limit words in every phase of the operation. Obviously, each separately handled data complex must be controlled by a unique limit word; that is, every communications channel has an input and an output area assigned to it (except for channels servicing one-way traffic at all times), each through a unique limit word. The area composing blocks will be taken from different memory locations from time to time and the size of each area may change (i.e., become larger or smaller) as the operation proceeds, but the limit word is unique and stationary throughout the operation of the system.

In addition to input-output area assignments, the possibilities for the application of limit words are countless, depending to a great extent on the ingenuity of the programmer. Intermediate storage, work areas and queues can be controlled by them; tables can be constructed from limit words or they may be used as two-field counters, when the count in the address field is incremented and then compared with the constant number in the limit field. When processing data, the address determines the location of the next character to be processed, and it is accessed by an indirect addressing method, providing the location of the limit word—for the machine logic—in which the character address is located. The limit field may be used for recording the location of the first character of the processed data unit or for saving other information, depending on the nature of a particular operation. The *D* field can be used for count-keeping or other references, as needed.

One of the most important utilizations of the limit word is for control of the available message buffer space in process storage.

It is possible to control the available memory space by one single limit word by storing the first character address of the first available block in the address, the last character address of the last available block in the limit, and the block count in the *D* field of the designated limit word. The blocks are linked together in one long chain. When a new block is needed, it is removed by the program from the beginning and when a block becomes available, it is returned to the end of the chain. The address or limit and the block count in the *D* field are updated each time a change occurs.

example with multi-level limit words

Operations with limit words may be organized on a two or more level basis. Queuing is a good example of a two-level operation, where one or more blocks of limit words (8 limit words in each block) are controlled by another limit word (which may be called a *queue limit* or *master limit* word) and each individual limit word controls a message. Considerable memory space and time saving can be achieved by efficient manipulation of limit

words: for example, multi-address messages can be transmitted to each destination from the same output area. Only the header portion of the message must be stored in the output queue of each channel on which the message will be transmitted. The header is contained in the last character positions of a block, and connects to the text part by the block control character. Additional levels may be used in other operations.

example related to polling

When the incoming line of a full-duplex channel becomes idle and it is about to be allocated to a terminal and the outgoing line is in use, the polling message is inserted between two data characters of the outgoing message, instead of waiting until the transmission is completed. One method to program this application is to position the polling characters in the last few data character fields of a block, as constants. When the polling operation becomes necessary, the program replaces the address of the next output character in the channel word with the address of the first character of the polling message. When the last bit of the currently transmitted output character is sent out, the first polling character is loaded in the assembly area of the channel word and from here on the characters of the polling message are obtained from process storage during "character interrupt" cycles and then transmitted via the channel. When the last polling character is loaded into the assembly area and the address is incremented to the block control character address, the end of block detecting circuitry automatically requests CHANNEL SERVICE MODE. The CHANNEL SERVICE MODE program restores the removed output character address in the channel word, instead of supplying the first character address of the next output block in the chain as it normally does, providing for the continuation of the interrupted transmission. In this operation, the program omits the return of the output block to the available buffer pool because the same polling message will be used each time for the allocation of the incoming line for the same terminal.

example using table lookup

Table lookup is one of the programmer's most effective tools for several reasons. Numerous programmed functions "call" for the application of tables, either by necessity or by logical choice, as the most feasible method. Functions like code conversion, editing, and character validity checking involve table lookup by their nature. Other applications such as message sequence checking, message numbering, and status checking, require table lookup operations due to the multiplicity of the components (terminals, lines) to be serviced or controlled. Simple arithmetic operations, if justified, and shifting must be performed by table lookup, since the PTC has no arithmetic unit, and no shift register available for the program. For the same reason, address modification is accomplished by replacement of the desired low order portion of the processs storage addresses with the contents of a designated register.

The method for the application of table lookup operations is the use of a bit pattern as an indexing factor and then accessing

and obtaining the desired entry from a table. The bit pattern may be a binary number assigned to a specific function by the program or the code bit configuration of an input or processor-provided output character. The entries in a table may be characters, addresses (in address and limit fields) and full words. Data words, limit words and combined words may serve these purposes. A character table must be located in process storage so that the first character address of the table, which is the process storage address of the first entry, ends in a number of low-order zeros equal to the number of bits in the bit pattern used to index the table. The location of a word table is determined by the same principle, but the number of low-order zeros of the first character address is $N + 2$, where N = the number of bits in the indexing bit pattern. This is because process storage operates with a 16-bit character address system and the two low-order bits, representing one of the four character addresses within a word, are ignored when a word is to be accessed. Tables composed of combined words count as word tables when their locations are determined.

A table entry is accessed by the program through address

Figure 3 Utilization of multiple table look up

Table A — **Code conversion and character validity check**

	A	B	C	D
XXXXXXXXXX000000				
XXXXXXXXXX000111				00000000000
XXXXXXXXXX001010			00001010110	
XXXXXXXXXX001100	00000001000			

Table B — **Table for action modifiers**

	A	B	C	D
XXXXXXXXX0000000		00000000011	00000000011	00000000011
XXXXXXXXX0001000	00000000011			
XXXXXXXXX1010110			00000011111	

Table C — **Action branch table**

	ADDRESS	LIMIT	D
XXXXXXXXXX000000	ADDRESS OF CHARACTER PROCESSING ROUTINE	REFERENCE ITEM 1	REFERENCE ITEM 2
XXXXXXXXXX011111	ADDRESS OF SPECIAL ROUTINE	REFERENCE ITEM 1	REFERENCE ITEM 2

modification. The indexing pattern is loaded into the low order bit positions of a register, then modifies the operational address of the instruction operating on the table entry. When a word entry is to be accessed in a table, the modifier is derived from a "shift left two" table before it is placed in the register, since words are accessed through the 14 high-order bits of a 16-bit address.

A table may serve more than one purpose. For example, in some applications, the same table may be used for validity checking and code conversion. Only one access per table entry is needed to perform the double function. In some cases, the multiple table lookup operation constitutes the most efficient means of programming. Figure 3 illustrates the organization of tables as an example for a multiple table lookup operation, including a double function table. The illustration refers to the processing of input characters. Table A is indexed by a hypothetical 6-bit code. Each entry in the table contains a 7-bit character (i.e., another code representation of the same character) for valid, and zeros for invalid input bit configurations. Table B is a "shift left two" table indexed by the 7-bit valid characters obtained from Table A. Each entry is a 6-bit modifier used to access the proper entry in Table C. Table C is an action branch table. Each entry determines the routine to be executed depending on the currently processed valid characters. The program uses the original 6-bit input character as a modifier to access an entry in Table A. It uses the entry obtained from Table A to access an entry in Table B, and the entry obtained from Table B to modifiy the operand address of an indirect branch instruction with the purpose of accessing the first instruction of a desired program routine. If an all zero entry is obtained from Table A, an error routine will be executed. If a data character is acquired from Table A (in the example, 0001000), the modifier obtained from Table B (000011) leads to the regular character processing routine (e.g., storing the converted character in a process area). If a special character is acquired (in the example, 1010110), the modifier (011111) leads to the special routine to be executed upon finding this specific special character. Reference items pertaining to each routine may be kept in the limit and *D* fields of the words in the action branch table if necessary. By using modifiers with two low-order 1-bits, the same modifier is used to access a word (address, limit) and the *D* field in the same word in Table *C*. Only seven instructions are required for the entire routine.

example of programmed area scanning

The programmer must determine a sequence in which the process storage areas, assigned to communications channels through unique limit words, will be examined to perform the necessary functions, as described under program organization. The first step is the assignment of successive identification numbers to the areas to be scanned. The channel words provide a good basis for the assignment. Areas to be serviced more often than others may each have two or more numbers assigned. The number will be updated each time when leaving an area in order to derive the

identifier of the next area to be examined. The number must be reset to the original starting value each time a scanning round is completed. The next task for the programmer is the creation of a program branch table (Figure 4). It is a word table in which each area is represented by one word. Program linkage is provided by recording the location of the program routine (i.e., the process storage address of the first instruction) which is to be executed, upon entering the area, in the address field of the table word. The table is indexed by the "shift left two" versions of the identifiers. If more than one identifier should be assigned to certain areas, a special purpose "shift left two" table is used to access the same table word, regardless of which of the several identifiers is used to initiate the servicing of the same area. When leaving an area, the initial address of the routine to be executed upon return is stored in the table entry by the program unless it is unchanged.

example of priority routine in normal mode

In case the SERVICE MODE program identifies an error which calls for the execution of an error routine, a suggested method is to execute the error routine in the NORMAL MODE. This will avoid keeping the machine in the highest priority mode and thereby delaying either the execution of other requested priority mode programs or another request for SERVICE MODE. However, the error routine may require a priority over the regular program routines in the NORMAL MODE program. The programmer, therefore, may generate his own "priority error routine" in the NORMAL MODE program by saving the contents of the instruction counter and operational data registers located in the word controlling the execution of the NORMAL MODE program, then placing the location address of the first instruction of the selected corrective routine (still in SERVICE MODE program) in the instruction counter immediately after the identification of the specific error. At the end of the selected error routine, the NORMAL MODE process word must be so restored that the execution of the interrupted routine will then continue. Such priority schemes in the NORMAL MODE may also be created for other purposes. Furthermore, by stacking the requests, a number of preferred routines—of a different or similar nature—may be executed continuously.

data flow

Figure 5 is a simplified chart illustrating data flow in the system during the input operation. The chart differentiates between actions in the PTC control storage, the PTC process storage and the CPU.

Figure 4 Program branch table

	ADDRESS		LIMIT	D
XXXXXXXXX0000000	ADDRESS OF AREA SCAN INITIALIZATION ROUTINE		REFERENCE ITEM	REFERENCE ITEM
XXXXXXXXX0000100	ADDRESS OF PROGRAM ROUTINE		REFERENCE ITEM	CHANNEL WORD ADDRESS
XXXXXXXXX0001000	ADDRESS OF PROGRAM ROUTINE		REFERENCE ITEM	CHANNEL WORD ADDRESS

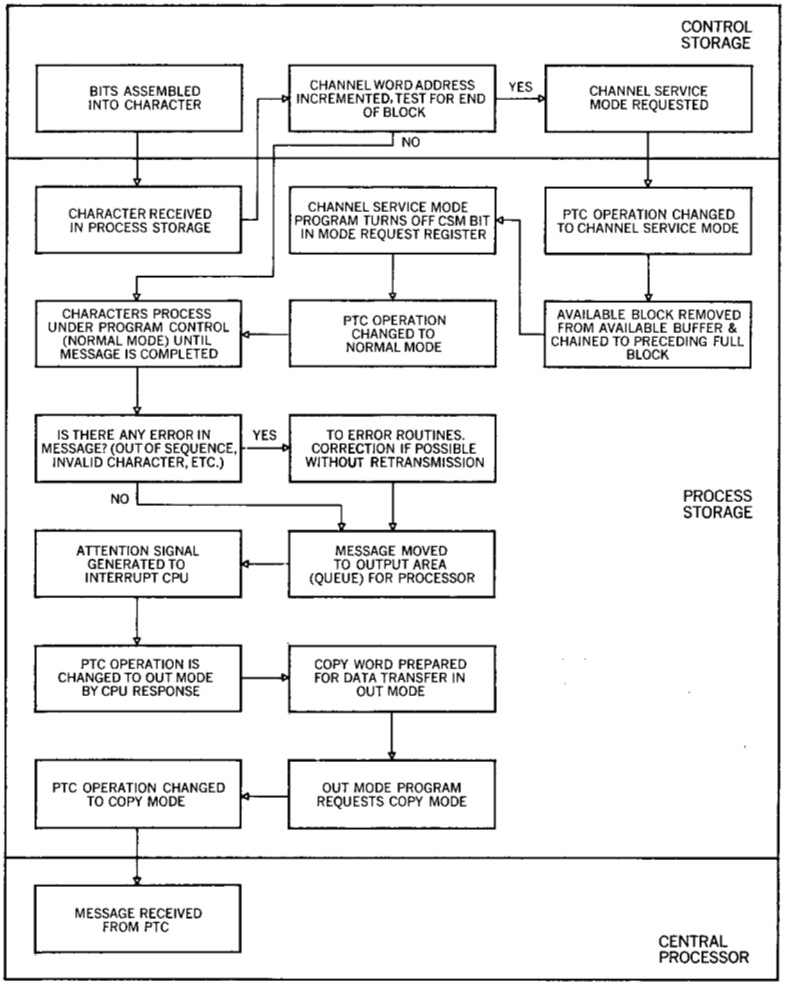
program maintenance

A well organized program will not only increase throughput but, equally important, facilitate program maintenance.

It is a much easier task to add, delete or change short routines than alter complicated long programs. The priority mode programs are comparatively short and each is divided into functional routines determined by its purpose. In the NORMAL MODE, where there would be room for long program routines, the environmental circumstances lead to numerous short routines.

A real-time system is exposed to changes to a broader extent than a conventional data processing system. However, most of the changes will affect the tables, constants and limit words rather than the programs. If the distribution of the traffic volume changes, then the schedule of programmed area scanning may have to be changed. If the system is expanded by additional terminals and channels, the number of entries in various tables, as well as the numbers of limit words and constants must be increased. This expansion has little or no bearing on programs if specifications

Figure 5 Simplified data flow chart for input



remain the same. If the system is contracted, the only difference will be a decrease instead of an increase. In all above cases, maintenance is comparatively simple. The only case in which the programmer faces a more difficult problem is when terminals and channels of completely different types are encountered in the system.

multi-programming

Multi-programming is necessary for the 7750 but it is provided essentially by the logical circuitry. Consequently it is a simplified problem. The programmer has to write five separate programs, only one of which, the NORMAL MODE program, requires special attention. Although the interaction among programmed and program-independent functions is intricate, the programmer need not consider the frequent occurrence of automatic actions because the only effect they may have on the five mode programs is the automatic random distribution of execution time among them through the automatic interrupt method. The integrity and continuity of the programs is protected by the logical circuitry.

continuous operation

Continuous operation is achieved to a great extent by circuitry designed to detect environmental errors and machine parity errors without stopping the machine, and to indicate them to the program in such a way that each error can easily and quickly be identified by nature and by component. When such errors occur, information is displayed by the machine logic to the program through a register and control words. At the same time, the operational mode is changed to the highest level priority mode, the SERVICE MODE, to allow quick access for the program to check the information.

system reliability

These automatic features accommodating multi-programming and continuous operation can be coupled with programming techniques for the isolation of faulty components and immediate communications with operating personnel. The program supplies a detailed diagnosis and thereby facilitates placing the faulty component back into operative status. For a system of this complexity, a high level of availability and reliability can be attained only through programming design which makes effective use of the automatic features of the components.

ACKNOWLEDGMENT

The programming techniques reported in this paper represent the work of a large number of individuals (from the IBM Data Processing and Data Systems divisions) who were responsible for the design of the 7750 and the design of the first particular systems to use the machine.

CITED REFERENCES AND FOOTNOTES

1. "CPU" denoting "central processing unit" refers to the system's central computer throughout the paper.
2. "Copy Word" is the control word for data transfer to and from the CPU.
3. "Mode Request Register" is the register in which requests for priority modes are recorded and terminated.

4. *IBM 7750 Assembly Program Using the IBM 1401*, Reference Manual C28-6259, International Business Machines Corporation, 1962.

BIBLIOGRAPHY

- *IBM 7750 Data Control Package*, Preliminary Bulletin J28-8096, International Business Machines Corporation, 1962.
- *IBM 7750 Programmed Transmission Control Programming Logic and Organization*, Reference Manual C22-6695, International Business Machines Corporation, 1962.