CHARACTER ORIENTED COMMUNICATIONS IN THE XEROX CP-V TIMESHARING SYSTEM

This paper summarizes the functions and implementation of the terminal I/O system for the Xerox CP-V Timesharing System. The information contained within was gleaned entirely from the released publications of Xerox Corporation listed in the bibliography and from examination of the released software - version B00 of the Xerox CP-V System.

John R. Catozzi
E225-F W.W. Chu
Spring 1974

Common to all timesharing systems is the problem of transferring character streams to and from user programs in the main memory of the computer to the user terminals. In most systems this action is initiated by the user program issuing a monitor service call specifying the desire to input/output characters from/to the terminal and indicating some set of parameters such as number of characters and the address of the character buffer in the users program. When the input or output function has been completed, control returns to the user program immediately following the service call and the program continues unconcerned with the complex mechanisms that have achieved this communication. This terminal oriented communications mechanism which is a combination of both hardware and software is the subject of this paper. Specifically described is the functional capabilities and the implementation of the Character Oriented Communications (COC) of the Xerox CP-V Timesharing System.

GENERAL USER I/O (BIG PICTURE)

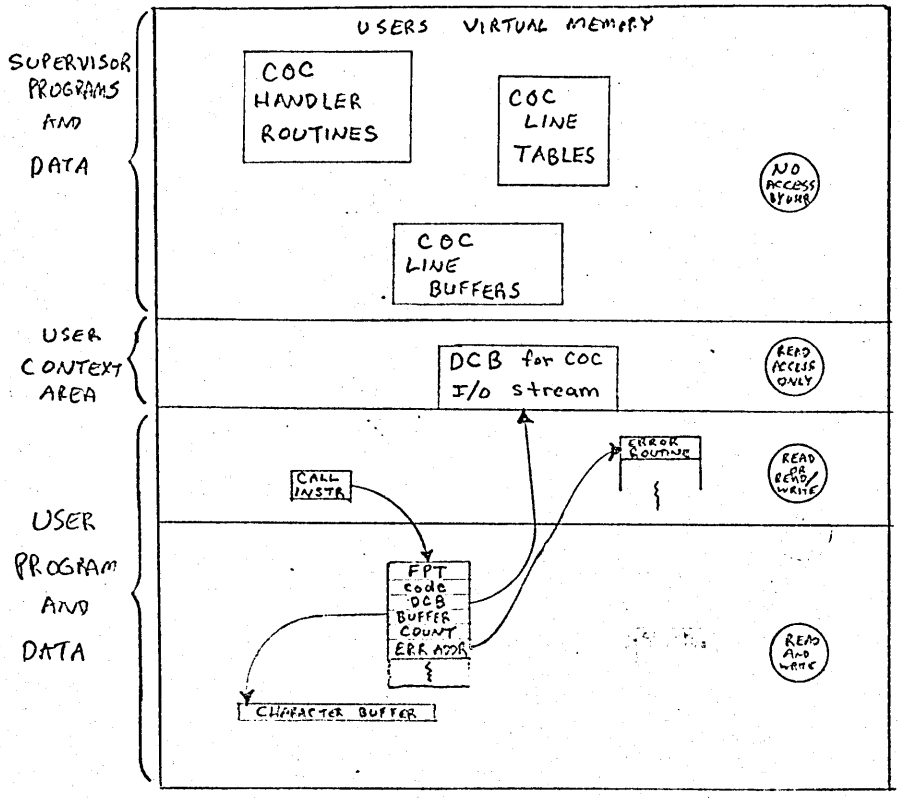
In CP-V, all user input/output requests are initiated by CALL instructions which cause entry to the monitor service routines. The effective address of the CALL points to a Function Parameter List (FPT) which describes the particular service required. Two of these available services are READ and WRITE. The READ and WRITE facilities are generalized in such a way that the user program does not require any knowledge about what kind of medium is being used for input, be it a teletype, a tape drive, a disk file, or any other such device. All the information describing the input or output medium is contained in a Data Control Block or DCB. There is a unique DCB associated with each I/O stream used in a program. DCBs are a part of the users program context and can be either defined by the user or included automatically by the program Load processor. The mapping from I/O stream to a particular device can be done external to the program by setting parameters in the DCB. In this way, a users program could READ data from a teletype or alternately from a tape drive, disk file, card reader or other device without any changes to the program.

There are three basic items contained in the FPT which are needed by the I/O system in order to perform a particular request. These are the address of the DCB for the I/O stream which is to be used, the address of buffer in the users program, and the number of characters to be read in or written out. Other parameters may be specified for special case action such as error return and Vertical Format Control. The I/O system uses this information to initiate the I/O action. Once the I/O action has been queued up, the system is free to give control to other user programs. For most I/O transfers (with the notable exception of large record tape and disc I/O), the actual transfer of data is buffered into common system memory so that the users program may be outswapped (transferred to external storage) for the duration of the I/O operation, thus making available its memory space for programs which are ready to execute.

TERMINAL I/O (INTERFACE TO THE USER PROGRAM)

All terminal I/O in the CP-V system is handled by the COC controller and its accompanying control routines. The COC routines may be logically divided into two sections. One section operates in a mapped addressing mode whereby it has access to the current users process and the other section operates unmapped (without regard to any particular user). The mapped portion transfers the READ/WRITE parameters from the users context (FPT and DCB) to system line tables and transfers the message characters to and from the users buffer area and its own buffer pool. The COC buffers are four words long and contain up to 14 characters and a halfword link to the next buffer. Free buffers are chained together by a word address link in the first word of each buffer in the free chain. Buffers are allocated to each line for both input and output as needed. When the routines are processing a write operation, enough buffers are obtained to contain the entire message. (Up to 140 characters may be output with a single write request.) Input characters are assembled into similar buffers from the same pool and completed messages are transferred to the users buffer directly from these COC buffers. Figure One shows this relationship. The mapped routines are only invoked at the start and completion of each message transfer request. The unmapped portion of the routines are interrupt driven by each character actually transmitted or received and are used to control the actual transfer of data from the COC buffers to the individual terminals. Both sets of routines interact with the system scheduler in order to change the user state (i.e., indicate that he may be outswapped, that he may be given control or that he is to be blocked from execution while output is in progress). All information pertinent to each line is kept in a set of resident tables indexed by line number. These tables are used and updated by both the mapped and unmapped routines. Figure Two shows the format of the line tables and the kinds of information that they contain.

FIGURE ONE. VIRTUAL MEMORY LAYOUT AND COC LINE BUFFER FORMAT



Line tables contain line information and pointers for input and output buffer insertion and removal point. (See buffer format below.)

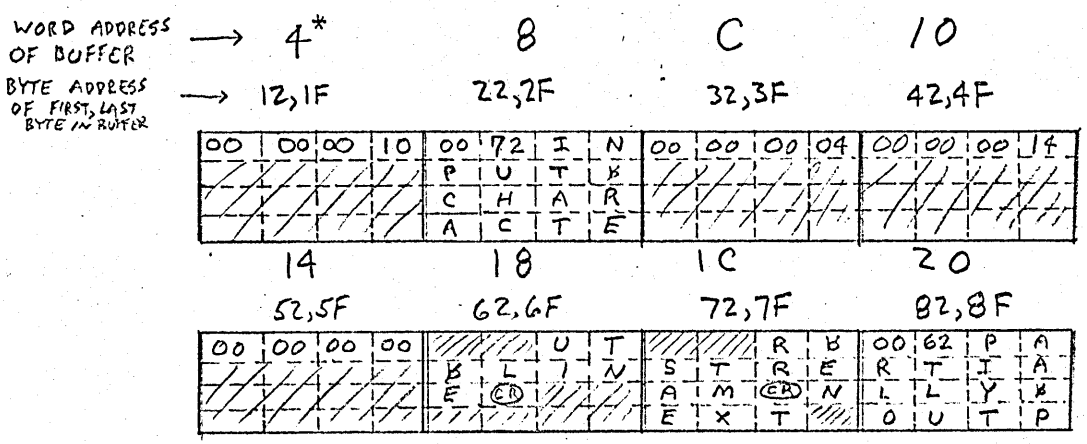
Data Control Block specifies COC output device, tabs in effect, line and page counts, line number and other parameters.

User issues CALL instruction invoking monitor service routine. CALL instruction contains address of Function Parameter Table.

Function Parameter Table specifies operation code (READ/WRITE) and contains addresses of the DCB, address of the data buffer in the users data area, error return addresses and other miscellaneous information.

LINE BUFFER FORMAT

Example shows eight buffers, four free, two being used for input, and two being used for output. Output is in progress, and the user has typed ahead one line and part of another.



0004C Free buffer word pointer

00004 Free buffer Count

* FIRST BUFFER HAS DISPLACEMENT OF 4 TO AVOID ZERO VALUE.
NOTE: ALL NUMBERS ARE HEX (BASE 16)

INPUT BUFFER INSERTION POINTER	007F
INPUT BUFFER REMOVAL POINTER	0022
INPUT CHARACTER COUNT	0018
OUTPUT BUFFER INSERTION POINTER	0069
OUTPUT BUFFER REMOVAL POINTER	0087
REMAINING OUTPUT CHARACTER COUNT	0011

0004 COUNT OF BUFFERS IN USE FOR THIS USER (used to limit type-ahead)

FIGURE TWO COC LINE TABLES

COC TABLES (Indexed by Line Number) Length LNOL *All are Interrupt Altered

COCOC
Count of remaining characters to be output
0 7
0 inactive
1 last character
being output & this COCOI & COCOR are meaningless.

LBUN
User No.
0 7

CPOS
Carriage Position
0 7
Indicates the current column number at which the timing is logically positioned.

CPI
Initial Carriage Position for a read
0 7

RSZ
User requested message size if a read is pending
0 7

ARSZ
Current size of records being read (0 echoed) while read is pending
0 7

COCTERM
Terminal Type
0 7
0 = 33 teletype
1 = 35 teletype
2 = 37 teletype
3 = XDS Model 7015
4-11 = IBM 2741

MODE
0 1 2 3 4 5 6 7
- Break count
- Restricted code set
- Tab simulation
- Read pending
- Transparent Mode
- Escape sequence, (EOA pending for 2741)
- Echoplex

MODE2
0 1 2 3 4 5 6 7
- Activation char. set
- Check parity mode
- Restrict code set to lower case
- 2741 line
- Space insertion mode
- Full Duplex Paper Tape Mode
- Logging off

MODE3
0 1 2 3 4 5 6 7
- Number of lines upspaced during input
- Lost input
- 2741 Keyboard Locked
- Backspace edit mode
- Half Duplex Paper Tape
- Relative Tabbing

BUFCNT
0 7
Current number of buffers in use by the line

MODE4
0 1 2 3 4 5 6 7
- Line Rate (indicates indirectly the chars/second)
- Terminal Type (Timing algorithm number)
- Unused

TL (HW)
0 15
Link to the buffer for input tab simulation while a read is pending
A value of 0 indicated no tabs in effect for the read.
Last byte is reserved for 35 editing.

EOTIME (HW)
0 15
Time of end of message
Also used to time out users having reads pending. In 1.2 second intervals.

COCOI
Byte pointer to current insertion point into the output stream for the line.
0 No buffer

COCOR
Byte pointer to the current removal point from the output stream for the time

COCII
Byte pointer to current insertion point into the input stream for the line
0 no buffer

COCIR
Byte pointer to current removal point from the input stream for the line

The following scenario illustrates the general flow for an input request:

1) The user program executes a CALL instruction which invokes the monitor I/O routines and passes the address of the FPT to these routines.
2) The general I/O routine obtains the DCB address from the FPT and determines from information in the DCB that this is a request for input from the user's terminal. The I/O routine passes control to the COC input routine.
3) The COC input routine moves the necessary parameters from the user's FPT and DCB to the tables for this user's line, sets up to send a prompt character to the user terminal (if specified) and then exits to the scheduler indicating that the current user is waiting for input (and therefore is a likely candidate for outswap).

For a write request, the flow is similar but after transferring the characters from the users buffer to the COC buffers, the user is allowed to continue execution until he exceeds a system limit for number of characters waiting to be output. At such a time, the COC routine reports to the system scheduler that the current user is to be blocked from execution. The user also may be blocked if there are insufficient buffers available to contain his current output request. In this case, he will be unblocked when buffers become available. The user who is waiting for input is rescheduled for execution when the COC routines report that the input on his line is complete. The user who is blocked from output is unblocked as soon as the count of characters waiting to be output falls below a system limit. The output block/unblock mechanism can be controlled by appropriate system limits so that a continuously outputting program will not overrun the COC buffers but at the same time there is always enough characters waiting to be typed so that the terminal is driven at or near full speed.

Other Functions of the COC Routines

Besides input and output of messages and directing the execution scheduling of interactive users, the COC routines perform many other functions. Some routines perform tab simulation, allowing the simulation of up to 16 tab stops for both input and output operations. There are also routines which generate and output page headings and control the pagenation of output, allowing the user to specify both the width and length of the page size. There are special routines to control input of paper tape in both full and half duplex modes and there are routines which allow system communication. These functions, while part of the COC routines, are not directly involved in the basic terminal I/O and will not be described in this paper. The other portion of the COC system which will be described is the actual device handler, that portion of the COC routines which respond to the per character interrupts by the hardware and control the input and output of characters from the COC buffers to the user terminals.

TERMINAL I/O (INTERFACE TO THE TERMINALS)

The moving of characters from the COC buffers to the individual line transmitters and from the line receivers to the COC buffers is the primary task of the unmapped portion of the COC routines. Besides moving the characters these routines do character translation from the internal EBCDIC form to the various terminal character codes, perform intrarecord editing on input and check for special function characters and activation (input complete) characters. Table One is a complete list of functions initiated by control codes and special input characters and gives a good general idea of the type of operations performed. The specific functions of each command and how they are carried out are beyond the scope of this paper except where they alter the general flow of control in the character handling routines. In order to understand how these routines function, one must first understand the capabilities and operation of the hardware used.

The COC Controller and Terminal Devices

The device used for character oriented communications on the CP-V system is the 7611 COC controller. The COC is a hybrid I/O device that logically connects to the system both through a Multiplexer I/O Channel (MIOP) and through the Direct I/O (DIO) interface to the CPU. Connections are also made for two external interrupts to the CPU from the COC. The MIOP connection is used for input of characters from the line scanner to main memory and once initiated, operates independently of the CPU. All output is done a character at a time under direct program control of the CPU by Write Direct (WD) instructions. WD and Read Direct (RD) instructions are also provided for sensing line status, data set status, and for controlling the state of the individual send and receive modules. Table Two lists the complete set of COC commands. The device consists of a controller interfaced to the MIOP and DIO systems and a line scanner and line interface units. Each COC can have up to eight line interface units each with eight send and

TABLE ONE. SUMMARY OF COC COMMANDS

FUNCTION PERFORMED	SPECIAL FUNCTION CHARACTER SEQUENCE OR CONTROL CODE	
	TTY	2741
Report Break Event	BREAK	ATTN, (or B ATTN on input)
Erase Only Current Input Line	ESC X	None
Tab Relative	ESC C	C ATTN
Suppress Lower Case	ESC U	U ATTN
Upper Case Shift	ESC (	(ATTN
Lower Case Shift	ESC)	) ATTN
Erase Last Character	RUBOUT	Backspace ATTN
Tab	ESC I, I^c	TAB
End of Input	FS,RS,US,GS	SPACE ATTN
Line Continuation	ESC CR, ESC LF	N ATTN
Retype	ESC R	R ATTN
Toggle Tabe Simulation Mode	ESC T	T ATTN
Toggle Space Insertion Mode	ESC S	S ATTN
End of File	ESC F	F ATTN
Pre-emptive call to System	ESC ESC,Y^c, ESC Y, four breaks	Four ATTNs, (or Y ATTN on input)

TABLE TWO. COC CONTROLLER FUNCTION CODES

COC CONTROLLER FUNCTION CODES

Code (Hexadecimal)	WRITE DIRECT Control Functions
0	Sense Receiver L Status
1	Turn Receiver L On
2	Turn Receiver L Off
3	Turn Receiver L Data Set Off
4	Sense Transmitter L Status
5	Transmit on L
7	Turn Transmitter L Data Set Off
D	Send Long Space on L
E	Stop Transmit on L
	READ DIRECT Control Function
0	Output Response

where L is line number 0 to $3F_{16}$

eight receive modules providing for a total of 64 lines. The COC routines have no intrinsic limit to the number of COC devices. The individual send and receive modules each have a pre-selected transmission speed (baud rate) and character format (number of bits per character and start and stop bits), and thus each port is dedicated to a particular class of terminals. The character translation selection is not so restricted, but there is no simple means of altering the default translation table (as specified during system generation) for a particular line if the command cannot be received intelligibly. Table Three lists the terminal types currently supported. Each type listed represents a class of terminal devices and the range covers the vast majority of asynchronous terminals. New device types and translation tables may be added without modifying the COC routines. In addition to character format and translation table, each line has a specified timing algorithm. The timing algorithm is used to determine how many "idle" characters are to be sent following tab and new line characters in order to avoid losing output during carriage movement. 2741-type terminals are sent IDLE (X'16') characters and all other types of terminals are sent NUL (0) characters. Table Four lists the five algorithms currently defined.

Character Input Processing

At system initialization, the character input operation is started by issuing a Start I/O (SIO) instruction to the COC device specifying the address of an I/O processor command chain in memory. This command chain is a read of some number of bytes to a specified address followed by a Transfer In Channel (TIC) command back to the read. This effectively causes an endless read operation using the input buffer in a ring fashion. As each character is collected by the scanner, it is placed in memory by the MIOP followed by the line number from which the character was received. After placement of the line number into memory, the COC input interrupt is triggered to inform the computer that it has input to be processed. This interrupt may be honored immediately causing entry

TABLE THREE. TERMINAL CLASSES ALLOWED

Type	Default Timing Algorithm	Translation
33	5	Teletype Model 33
35	5	Teletype Model 35
37	5	Teletype Model 37
7015	5	Xerox 7015 Keyboard Printer
DATA[POINT]	0	Teletype Model 33
EAPL	1	IBM 2741 EBCD APL
ESTD	1	IBM 2741 EBCD Standard
EXEC[UPORT]	2	Teletype Model 33
MEMO[REX]	3	Teletype Model 37
SAPL	1	IBM 2741 Selectric APL
SSTD	1	IBM 2741 Selectric Standard
TI	5	Teletype Model 33 (TI is Texas Instrument's 700 series)

TABLE FOUR. TIMING ALGORITHMS

Timing Algorithm Number 0

Usage: Teletype Models 33, 35, and 37 and alphanumeric displays.

Idles[†]: None

Timing Algorithm Number 1

Usage: IBM 2741 and 2741-compatible equipment.

Idles[†]:

Before carriage return = none

After carriage return = (curpos[††]+15)/10

After tab character = (new position-old position+15)/10

Timing Algorithm Number 2

Usage: Execuport and Dataport terminals.

Idles[†]:

	0-10 cps[†††]	11-15 cps	16-30 cps	31-60 cps	61- cps
Before carriage return	0	0	0	0	0
After carriage return	1	4	8	12	16
After tab character	1	1	2	4	8

Timing Algorithm Number 3

Usage: Memorex terminals

Idles[†]:

	0-10 cps[†††]	11-15 cps	16-30 cps	31-60 cps	61- cps
Before carriage return	7-curpos[††]	10-curpos	20-curpos	40-curpos	40-curpos
After carriage return	0	0	0	0	0
After tab character	1	1	2	4	8

Timing Algorithm Number 4

Usage: This algorithm is a combination of the others and may be used to ensure that an inexperienced user can utilize the system without any character loss. It also supports an experienced user until a change in terminal type can be entered. It is suggested that installations with mixed types of high-speed terminals use this algorithm as the default for high-speed lines.

Idles[†]:

	0-10 cps[†††]	11-15 cps	16-30 cps	31-60 cps	61- cps
Before carriage return	7-curpos[††]	10-curpos	20-curpos	40-curpos	40-curpos
After carriage return	1	4	8	12	16
After tab character	1	1	2	4	8

Timing Algorithm Number 5

Usage: This algorithm is used for terminals that require a number of idles roughly proportional to the carriage movement distance. It may be used for Teletypes and other equipment of similar mechanical design, and is sometimes a better algorithm than number 0 for such equipment.

Idles[†]:

	0-10 cps[†††]	11-15 cps	16-30 cps	31-60 cps	61- cps
X =	60	50	18	15	15

Before carriage return = none

After carriage return = (curpos+15)/X

After tab character = (new position-old position-15)/X

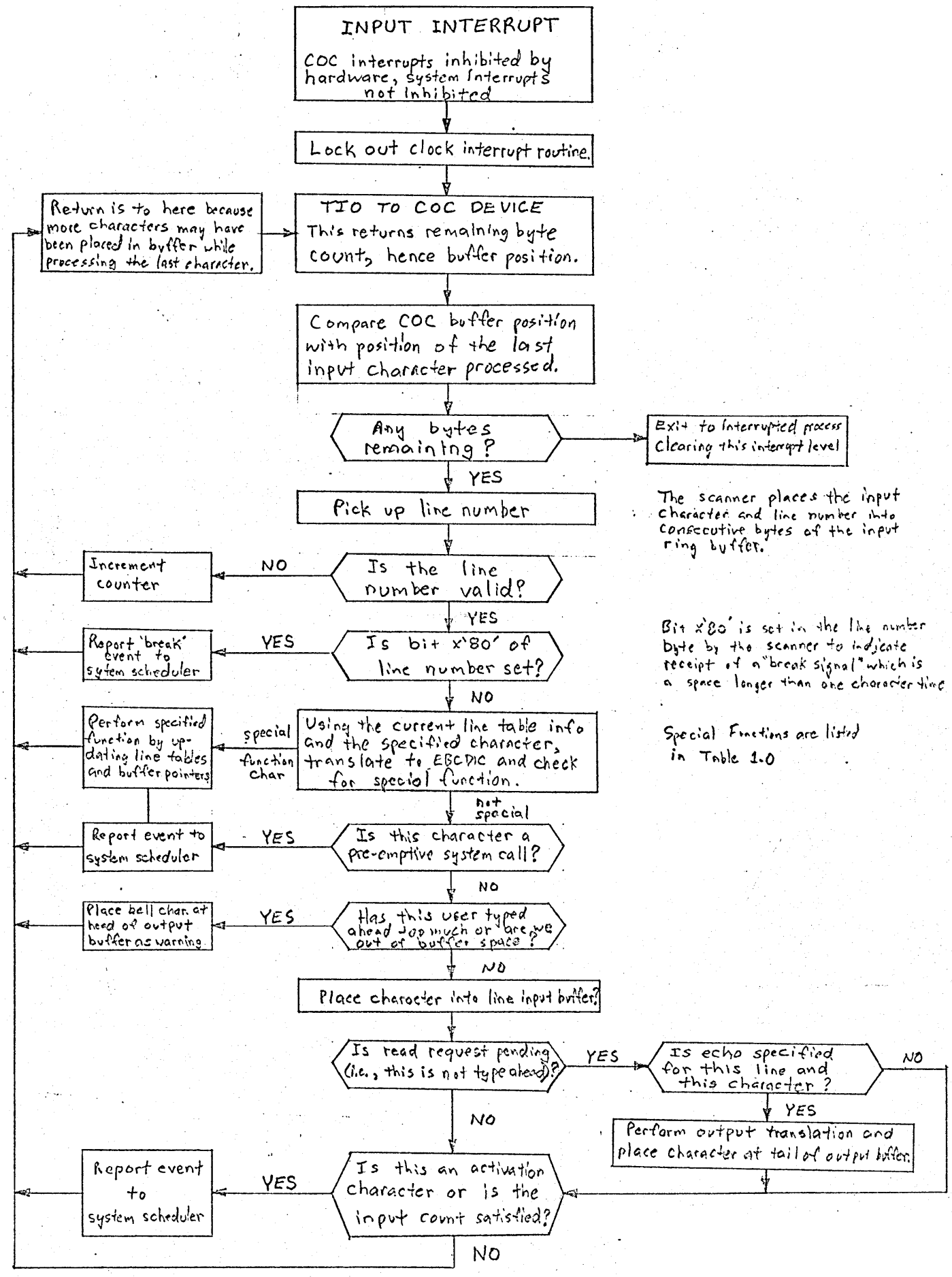
[†] Many high-speed terminals require a delay before sending a carriage return, after sending a carriage return, or after sending a tab character. In such a case, the COC handler must send "idle" characters, the number of which depends upon line speed, carriage position, and characteristics of the particular terminal.

[††] Current carriage position.

[†††] Characters per second.

to the input processing routine or it may be deferred because of a higher priority task. Irregardless of whether or not the interrupts are acknowledged, the MIOP will continue placing the input characters into memory as they are received by the COC. When the input interrupt is honored by the CPU, the input processing routine in the COC handler is entered. By issuing a test I/O (TIO) instruction addressing the COC device, the input routine can determine the current remaining byte count in the I/O processor and therefore deduce the address of the last character it placed in memory. The routine processes all characters until it "catches up" with the COC device. The disparity in speeds between the computer and the terminals is such that one-two bytes per line is an adequate size for the ring buffer for a predominence of 110 and 300 baud lines. A large number of 1200, 1800 and 2400 baud lines would require a larger buffer to prevent overrun. The processing of each character involves checking for special function characters, system call characters, character translation, echoing of the character back to the full duplex terminal and moving the input characters into linked buffers for each line. As a result of certain control characters and input completion characters, relevant events are reported to the system scheduler for the involved users. During the processing of input characters, other system functions of a higher priority are allowed to interrupt. Most other I/O device servicing is a higher priority task as well as the system clock interrupt handling, but not the handling of character output, which is a lower priority task. The detailed flow of the input interrupt handling is outlined in Figure Three.

FIGURE THREE. INPUT INTERRUPT PROCESSING

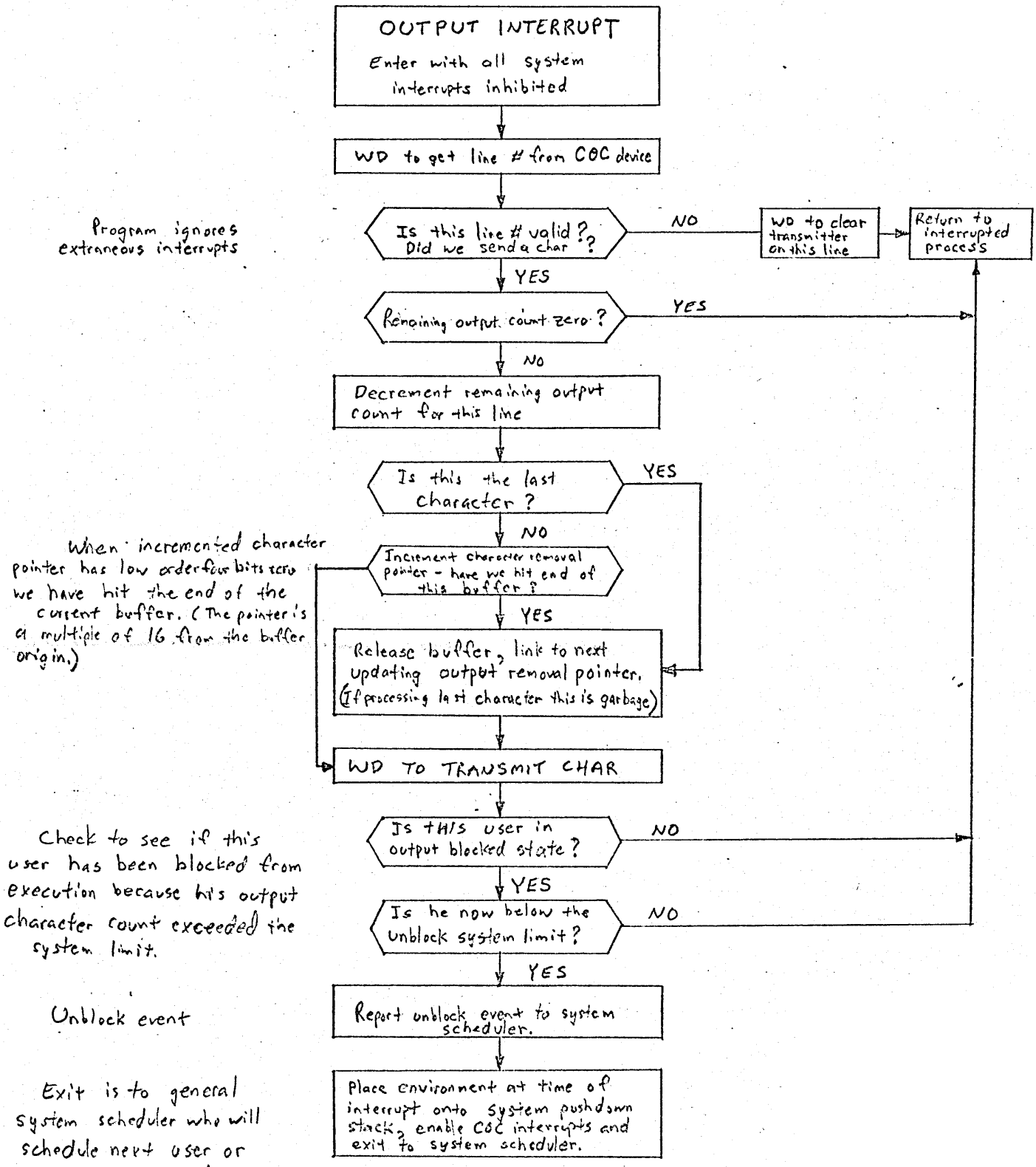


Character Output Processing

Character output on each line is initiated by the WRITE routine whenever it places characters into output buffers for a previously idle line, idle meaning that there is no output in progress irregardless of any input that may or may not be taking place. (Although the COC routines are capable of handling half duplex terminals, the input and output functions are independent of each other.) To initiate output, the routine loads a CPU register with the line number and character to be sent and issues a WD instruction to the COC. Once started, output for a line is driven by the output interrupts (which signal completion of transmission of a character) until no more output exists for this line. The output routine is very simple and takes only a very short time to process a character (all translation has all ready been done, the output routine only picks up characters and outputs them to the terminals). Although the output routine is a lower priority task than the input routine, it cannot be allowed to be interrupted by that routine which might alter the line tables for the line being processed, therefore, the output routine is entered with all system interrupts inhibited and usually remains that way until it exits. This is acceptable to the general system because of the short duration necessary for procesing output interrupts.

Upon entry, the output processing routine issues a RD instruction to the COC which returns the number of the line which causes the interrupt. If there is no more output in the buffers for this line, the transmitter is turned off and the routine exits. If more output is available, the next character is output with a WD instruction. The routine releases buffers to the free pool when they become empty, and chains to the next buffer in the output queue for the line being processed, when necessary. The only other functions performed are optional parity bit generation and reporting to the system scheduler whenever a blocked user's output count falls below the system limit, so that he can again be scheduled for execution. Figure Four details the flow for the output interrupt processing.

FIGURE FOUR. OUTPUT INTERRUPT PROCESSING



Line Initialization and Dropped Line Detection

A third interrupt driven routine is entered every 1.2 seconds from the system clock routine. This routine polls all the lines via WD instructions to see if any previously dormant lines have connected and to see if any connected users have been dropped, either by line disconnect or deliberate hang-up. New lines are reported to the scheduler so that a new user may be added to the system, and line disconnects are reported so that the user may be gracefully deleted from the system. One additional task of this routine is to time pending input requests so that a user may be logged off the system if he does not respond within a system specified time limit. This facility helps prevent users from tying up valuable communications equipment and preventing access to other users. Although system clock interrupt servicing is allowed during the processing of input interrupts, this routine is prevented from interferring by a flag, set by the input processing routine, which indicates to the clock service module that this routine is to be skipped. For proper timing, a count is kept of the number of successive times the routine is skipped because of this flag.

IMPACT ON SYSTEM

Approximately 2000 words of memory are required for COC handling routines, and are allocated as follows:

1. Input and output interrupt routines take up 500 instructions.
2. Read/write routines are comprised of 500 instructions.
3. Activation detection and echoing routines contain 400 instructions.
4. Get/put buffering routines have 200 instructions.
5. Line detection and initialization routines have 200 instructions.
6. The teletype translation table requires 65 words of memory.
7. Miscellaneous tables and constants comprise the remaining 135 words.

Additional storage is required for each communication line in the system; 23 bytes for control information and eight words (average) for buffering input and output messages.

IBM 2741-type terminal translation tables are available via SYSGEN parameters for EBCD and standard code sets.

Four translation tables are available for 2741-like terminals, allowing translation of EBCD and Selectric(r) code sets with either standard or APL keyboards. Each translation table adds 96 words to storage requirements if incorporated in a system.

This results in about 1800 words of tables and buffers for 100 lines (with 2741s).

Assembly parameters have been defined to allow conditional assembling of the procedure concerning 2741 terminal logic, page headings, performance monitoring, and buffer security checking. Assembling out all of these will reduce core requirements by 760 words.

Approximate execution times in microseconds are:

Write processing - per write	250
additional per character	140
Read processing - per read	580
additional per character	220
Input interrupt processing - per character	110
Output interrupt processing - per character	80
Buffering routines - per 14 characters buffered	110

Assuming an average write size of 40 characters and an average read size of ten characters, the per character execution time will be approximately 235 µsec on output and 399 µs on input. Average terminal I/O rates of one character input and four characters output per second per user result in an overhead burden of 13.4% of a SIGMA 7CPU per one hundred users.

REFERENCES

Xerox Publications

Xerox Control Program - Five (CP-V). Data Bases Technical Manual, 90-19-95, Section VG.05.

Xerox Control Program - Five (CP-V). Systems Management Guide, 90-16-74F.

Xerox Control Program - Five (CP-V). Time Sharing Reference Manual, 90-09-07F.

Xerox Universal Time Sharing System, Basic Control and Basic I/O Technical Manual, 90-19-85A, Section DC.

Xerox Released Software

Xerox Control Program - Five (CP-V) Module listings for version B00: CALPROC, ALTCP, COC, COCD, IOQ, SCHED