Implementation of an 8-Channel Statistical Multiplexer

(8-채널 통계적 다중화기의 구현)

李鐘樂*, 趙東浩*, 殷鍾官*, 宋吉鎬**, 鄭元載**

(Jong Rak Lee, Dong Ho Cho, Chong Kwan Un, Kil Ho Song and Won Jae Cheong)

要 約

본 논문에서는 마이크로프로세서를 이용한 8-channel 통계적 다중화기(SMUX)의 구현에 대하여 기술한다. 하드웨어는 S100-bus 비슷한 bus를 통하여 연결되어 있으며 4MHz clock의 Z-80A 중앙처리 장치기판, 프로그램 저장을 위한 16Kbyte LOM기판, data저장을 위한 16Kbyte 동적 RAM 기판 및 세개의 입출력 장치로 구성되어 있다. 이 통계적 다중화기는 50bps에서 9600bps까지의 data를 취급하는 8-channel을 다중화 할 수 있고 한장의 입출력 기판을 제거하고 소프트웨어를 약간 수정하면 4-channel을 수용할 수 있다. 또한 본 장비는 CCITT 권장사항 X.25 link level, V.24, V.28, X.3 및 X.28을 따르고 있다. SMUX 주요특성은 4종류의 입력부호 즉 ASCII, EBCDIC, Baudot, Transcode를 취급할 수 있고 동적 buffer 운영방식과 자체진단 기능을 갖고 있으며, 전체 시스템을 동작시키는데 단지 하나의 CPU를 능률적으로 이용한다는 점이다. 이 시스템의 하드웨어 및 소프트웨어에 관한 자세한 사항은 본론에서 기술한다.

Abstract

In this paper we present development of microprocessor-based 8-channel statistical multiplexer (SMUX). The hardware design includes one Z-80A CPU board with the clock rate of 4 MHz, one 16 Kbyte ROM board for program storage, one 16 Kbyte dynamic RAM board and three I/O boards, all connected through an S-100 compatible tristate bus. The SMUX can presently multiplex 8 channels with data rates ranging 50 bps to 9600 bps, but can be reduced to accommodate 4 channels by having a slight modification of software and removing one terminal I/O board. The system specifications meet CCITT recommendations X.25 link level, V.24, V.28, X.3 and X.28. Significant features of the SMUX are its capability of handling 4 input codes (ASCII, EBCDIC, Baudot, Transcode), the use of a dynamic buffer management algorithm, a diagnostic facility, and the efficient use of a single CPU for all system operation. Throughout the paper, detailed explanations are given as to how the hardware and software of the SMUX system have been designed efficiently.

*正會員, 韓國科學技術院 電氣 및 電子工學科
(Dept. of Elec. Eng., KAIST)

**正會員, 金星電氣技術研究所
(Institute of Technology, Gold Star Electric Co., Ltd.)

接受日字: 1984年 2月 29日

I. Introduction

To increase the information processing capability and the utilization of a computer facility, computational resources of remotely located computers and/or terminals must be shared via an interconnected computer

communication network. In implementing such a network, the communication system design influences the overall system performance and the system costs. To reduce the communication cost, a multiplexing technique is normally used. Frequency division multiplexing (FDM) and synchronous time division multiplexing (STDM) are two commonly used techniques in conventional data communication systems. FDM divides a channel bandwidth into several subchannels such that the bandwidth of each subchannel is at least as great as that required for a single message channel. On the other hand, in STDM each user is assigned a fixed time duration or a time slot on the communication channel. The multiplexing apparatus scans the set of users in a cyclic fashion. After one user's time duration has elapsed, the server is switched to another user. In both FDM and STDM systems, addressing is usually not required, since the user is identified by his frequency portion in FDM, or by his time slot position in STDM.

It is known that FDM does not make efficient use of channel bandwidth because of the need to employ a guard band to prevent a data signal of the channel from interfering adjacent channels and because of the relatively poor data transmission characteristics of the voice band channel near the edges of its band. Although STDM is considered to be more efficient than conventional FDM systems, the STDM technique also has certain disadvantages. Statistical data collected from several typical operating time sharing systems have shown that the system is in an idle state for a large amount of time, and actually sends information only for a small fraction of time. Thus, the conventional STDM technique is inefficient in channel utilization in such an environment, since it allocates a time slot to each user whether or not the user is active.

To utilize the channel bandwidth more efficiently, the asynchronous time division multiplexing (ATDM) technique, which is also called statistical multiplexing, has been proposed for computer communication. In statistical multiplexing, data are asynchronously multiplexed with respect to the users. When a user has a message to transmit, he is granted access to the channel. If a user has no message to transmit, it becomes disconnected from service. Statistical multiplexing requires an address for each transmitted message and buffering to handle statistical peaks in random message arrivals. Hence, SMUX is more complex as compared with the STDM system, but can yield a large gain in channel utilization.[1]

This paper deals with implementation of an 8-channel statistical multiplexer (SMUX). The system specifications meet completely CCITT recommendations X.25 link level, V.24, V.28, X.3 and X.28. Both hardware design and firmware implementation will be described in detail.

Following this introduction, in Section II we describe the overall system configuration and the system main features. The hardware architecture is explained in detail in Section III. In Section IV software implementation is described. Finally, conclusions will be made in Section V.

II. Description of Overall System Configuration

The system operation of statistical multiplexers is illustrated in Fig. 1. The output data of 4 or 8 asynchronous terminals are multiplexed, and transmitted through the channel with or without modems. Then, received data frames are demultiplexed and distributed to each corresponding terminal. The data in reverse direction are processed and transmitted in the same way.

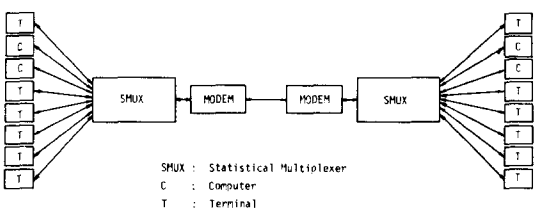

Fig. 1. An example of SMUX operations.

At the transmitter, the input data rate of each channel can be varied from 50 to 9600 bps by appropriate switch settings. Input data are stored in an input line buffer, and echo-back is serviced to the terminal by the SMUX for each character satisfying CCITT-recommended protocols, X.3 and X.28[2]. When the line buffer is full, or the timer runs out, the stored data frame in the line buffer is transferred to the main buffer. A dynamic buffer management algorithm is used to allocate memory to each frame. If the input traffic is high and buffer overflow is to occur, the input data are temporarily stored in the input waiting buffer to protect data loss. Then, each channel is notified that the buffer is almost full. Also, it is checked which channel has been down. If any channel is down, the system is recovered by releasing the corresponding data frames. The data frame stored in the main buffer is serviced on the first-in first-out (FIFO) basis, and transmitted synchronously through the modem. The speed of the modem can be varied from 1200 to 9600 bps according to the input traffic and error condition of the transmission line. In the SMUX system, error-free transmission, sequence control and flow control are supported by X.25 data link protocol. Accordingly, an output waiting buffer is needed for retransmission.

At the receiver, the received frame is checked and processed by the X.25 link routine. If it is a valid data frame, it is stored in the main buffer. In case of transmission errors, retransmission is requested by informing the transmitter that the received frame has been rejected. The stored data frame in the main buffer is serviced on the FIFO basis, and passed to the packet assembly and disassembly (PAD) part to be sent to the corresponding terminal.

The main features of the SMUX described above include its capability of handling various input codes, error-free transmission, down-line programming, front panel diagnostics, channel speed control, channel rate adjustment, dynamic buffer management, aggregate/input interface compatible with CCITT recommendations V.24 and V.28[3], and so forth. Detailed specifications are summarized in Table 1.

Table I. Main features of SMUX system.

Number of channels to be multiplexed.	4 or 8
Synchronization of data	Asynchronous
Input data code	ASCII, Baudot, EBCDIC, Transcode
Transmission speed of input data	50, 75, 110, 134.5, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600 bps
Number of data bits	5-8 bits
Aggregate speed	1200, 1800, 2400, 3600, 4800, 7200, 9600 bps
Aggregate/input interface	EIA RS232C (CCITT V.24/V.28)
Protocol	X.25 (link level) X.3, X.28
Error detection	16 bit CRC
Error correction	ARQ
Operating mode	Point to point (full duplex)
Buffer size	16 Kbytes
Program size	16 Kbytes
Buffer management	Dynamic management

The system can handle a total input/output data rate up to 20 kbits/s when the CPU operates at the clock rate of 4 MHz. It can support eight users simultaneously that transmit data at an average rate of 1200 bits/s each. The system can have greater data processing capability if the CPU clock of a higher rate is used. However, the transmission speed of each user will be limited without regard to the system performance capability when the error rate on the synchronous aggregate lines becomes high.

III. Hardware Design and Implementation

The SMUX system hardware consists of a central processing unit (CUP) board, a read-only program memory board, a read/write

random access memory (RAM) board, and three (or two when reduced) input-output (I/O) boards. Since the hardware constitutes the configuration of a general microcomputer system, our description will be concentrated on the specially designed I/O boards. Only a brief explanation will be given for other parts of the system.

The CPU board uses a Z-80A microprocessor chip that operates under a 4 MHz system clock. During the development phase we used an S-100 compatible bus, but, at the end of development, we replaced it with a Z-80 system bus. The ROM board uses 1 Kbyte/chip erasable-programmable ROM (EPROM) chips, which add up to 16 Kbyte capacity. Also, a 16 Kbyte dynamic RAM board is used. The memory capacity can be expanded by adding the same RAM boards.

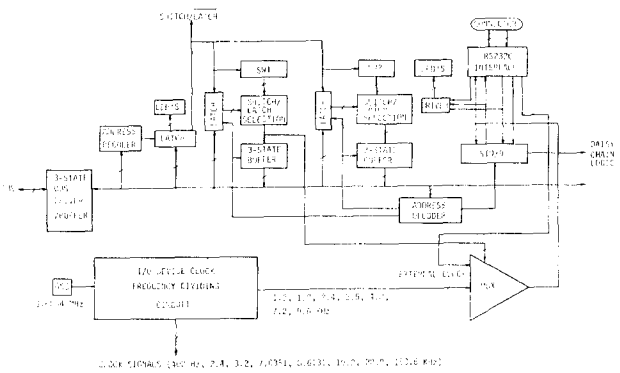

Fig. 2. Hardware block diagram of the main I/O board.

There are two kinds of I/O boards in the SMUX system, referred to as the main I/O board and the terminal I/O board, respectively. Fig. 2 shows the hardware block diagram of the main I/O board. As shown in the diagram, there is a modem port that is used to transmit the multiplexed (or aggregate) synchronous output and to receive the synchronous input data which is to be demultiplexed, and two dual-in-line package (DIP) switches. One of these switches, SW1, specifies the transmission speed of the synchronous aggregated line with an optional external clock received from the remote equipment. The operator of the statistical multiplexer can raise or lower the line speed according to the condition of the current aggregate channel.

The other switch, SW2, defines some other functions. The most important one is the function of master/slave selection, which determines the multiplexer to act as a master equipment or as a slave. The master and the slave act differently at their initial handshaking phase. Initially, both run modem control routines. After physical interconnection, they open a full-duplex data link by transmitting supervisory command frames. If the link is successfully set up, and down-line loading has been enabled by SW2 at the master, then down-line-loading is done. And the master reads its switch settings and various internal parameters, and sends them through the link. When the slave receives them, it writes the information on its latches prepared for each switch. At the slave, the switch settings are completely ignored. But, a part of the switch settings of its own may also be saved in the latches. For such purposes, the CPU must be able to read the switch settings and the contents of the latches, and load the latches with hardware control information. Either the switch settings or the latch contents are loaded on the data bus. The choice between the switch and the latch is determined by a signal called SWITCH/LATCH, which is also stored in another latch. Some signals thus selected are used to control the other parts of the hardware directly. The master/slave function and the down-line loading are necessary for diagnostics that is initiated by the master. On the main I/O board are displays that inform the operator of the overall system status. There are six light-emitting diodes (LED's) that display the latch information, and three LED's that display the status of the aggregate lines, driven by the hardware signals.

In addition, the main I/O board has a circuit that generates the clock signals of various frequencies to be used by its own or by other I/O boards. Of these, seven signals

of 1.2, 1.8, 2.4, 3.6, 4.8, 7.2, and 9.6 kHz are used as the transmit clock signals, while the other seven in other I/O boards are used to control asynchronous timing for user terminals. Also, a 400 Hz clock signal is made available for use in other I/O boards as a basic timing unit for counter/timer circuits (CTC's).

The SIO chip plays the most important role in the main I/O board. It sends to and receives from the modem the serial data, converts them into 8-bit parallel data to be used by CPU, performs cyclic redundancy check (CRC), and so forth. Also, between SIO and the modem port is an RS232C interface circuit that converts voltage levels.

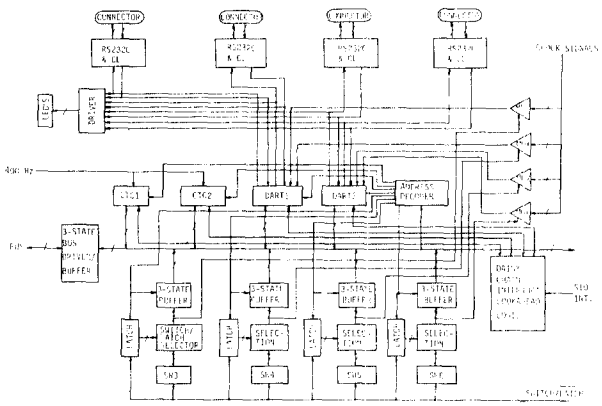

Fig. 3. Hardware block diagram of the terminal I/O board.

The block diagram of the terminal I/O board is shown in Fig. 3. Up to four start-stop mode terminals can be connected to this I/O board as stated above. A DART can handle two terminals with one having higher priority than the other, and two DART's are connected in a Z-80 "daisy chain" interrput circuit. As a result, the four asynchronous channels have relative priorities among them. An RS232C interface circuit that converts voltage levels and also a current loop circuit for teletypes are used between each terminal port and the DART. The DART converts serial asynchronous data into 8-bit parallel ones so that the CPU can handle them. Seven different clock signals supplied by the main I/O board are available for each channel. The frequency of the clock signal used is determined by the binary speed of that channel, and is divided by 16, 32, or 64 for use internally in the chip.

For each channel of the terminal I/O board, there is a DIP switch whose functions are local to that channel. They are the channel transmission rate, number of the data bits used and the number of stop and parity bits for transmission. With these switches there are eight LED's that are driven by hardware signals, which display the input-output activity of each channel. Timing interrupt is initiated by two CTC's, which are used in the counter mode with their basic timing interval of 2.5 ms (400 Hz). Each CTC chip contains four counter circuits, which can, therefore, handle two channels simultaneously. Two counters are cascaded externally for one channel, and can count as long as about 2.7 minutes. Since three kinds of timeouts are used in this multiplexing equipment, a timer has to be managed dynamically.

Z-80A peripheral devices are interconnected in a daisy chain interrupt structure, where the delay due to each device is accumulated with increase in the number of devices and result in malfunctioning if it is more than four. This problem was overcome by using an interrupt lookahead logic circuit on each I/O board. By modifying jumper connections, the lookahead circuits are appropriately interconnected when one adds another terminal I/O board.

IV. Software Design and Implementation

The SMUX firmware which was developed using Z-80 assembly lauguage can be divided into four parts according to their operation; an initialization routine, a polling service routine that includes the input and output process of dynamic buffer management, an idle routine, and interrupt service routines composed of SIO and DART interrupt service routines. A flowchart showing the interrelationship among these routines is given in

Fig. 4. In what follows these routines are discussed.

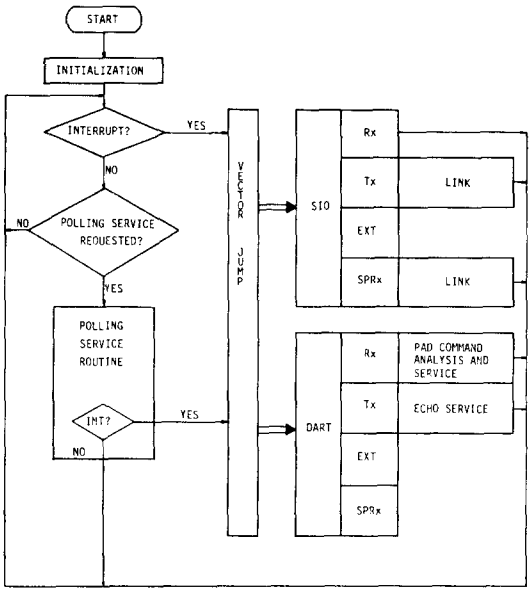

Fig. 4. Block diagram of SMUX firmware.

1. Initialization

Before starting actual operation, the SMUX must be initialized. The system memory is cleared and initial values of the parameters are loaded. Also, the SIO function of link level is initialized, which includes handshaking with a modem that is connected to the SMUX. Also, initialization of DART's for servicing terminal equipments and CTC's for timer operation follows. In this initialization process the information regarding SMUX system operation is read from the front panel switches, and CPU initialization such as interrupt mode selection, stack pointer setting and so on is made.

2. Polling Service Routines

These routines consist of the input line buffer (ILB), output line buffer (OLB) service routines, and the routines of the buffer management input process and output process.

When ILB services are requested, the requesting channels are processed in turn. The data frame is transferred from the input line buffer to the main buffer. That is, the input process of dynamic buffer management is executed. If the size of the input line buffer content is longer than 80 bytes, the extra bytes are rearranged. After the ILB is forwarded, the corresponding ILB request is reset.

The input process of dynamic buffer management is performed according to the modified Chu's algorithm[4]. The flowchart of the implemented input process is shown in Fig. 5. The input process collects addressed input data frames and allocates them to the main buffer efficiently.

If an input process is requested, a block available list (BAL) which c contains the addresses of available space is investigated. If not available, the received frame is temporarily stored in an input waiting buffer, or all the data frames of the corresponding channel are released according to the situation. When there exists any available buffer space, it is examined whether buffer overflow has been informed of. In the case that a data frame is stored in the input waiting buffer, it will be serviced now. Requested data frames are serviced according to the frame addresses, the address translation table (ATT) which has the first block address (FBA), the last block address (LBA), and a buffer status bit (BSB) for each channel. The buffer status bit indicates that there is at least one stored data frame for that channel in the main buffer. When the main buffer contains a frame for a specified channel, the linkage pointer is updated to point to the physical address of the first byte of the newly arrived data frame, and the block containuation bit is updated. And then, the last block address of the ATT is then, the last block address of the ATT is updated to the available memory address obtained from the (BAL).

In the case that there is no data frame for that channel, the FBA of the ATT is updated to the value of available memory address

obtained from the BAL, and the block status bit is set to 1. Next, the data frame is processed to be stored into the main buffer from the ILB. The control now returns to the monitor.

On the other hand, in the case that output line buffer (OLB) services are requested, the corresponding channels will be serviced in turn. When the data frame of the channel is in the main buffer, it is moved to the output line buffer. That is, the output process of dynamic buffer management is done. Then, the corresponding OLB service request is reset.

The output process of dynamic buffer management is performed according to the modified Chu's algorithm[4]. The detailed implmentation of output process is shown in Fig. 6. The output process efficiently distributes the data in the main buffer to the corresponding destinations.

When the process is requested, the data frame is transferred from the main buffer to the destination buffer, and the BAL is updated by the FBA of the ATT. Following this, if the data frame continues the next frame, the FBA is updated by the linkage pointer which indicates the first byte address of the next frame. Then, the control returns to the monitor as before. If the serviced data frame is the last one, the values of the block status bit, LBA and FBA are reset, and the control returns.

In the case that the input process of dynamic buffer management is desired by the SIO special receive interrupt service routine, the task is done in almost the same way as shown in Fig. 5. When the output process of buffer management is requested by the SIO transmitter interrupt service routine, the function is processed as shown in Fig. 6.

3. Idle Routine

The idle routine shown in the overall flowchart of Fig. 4 does nothing but checking and servicing interrupt and service routines.

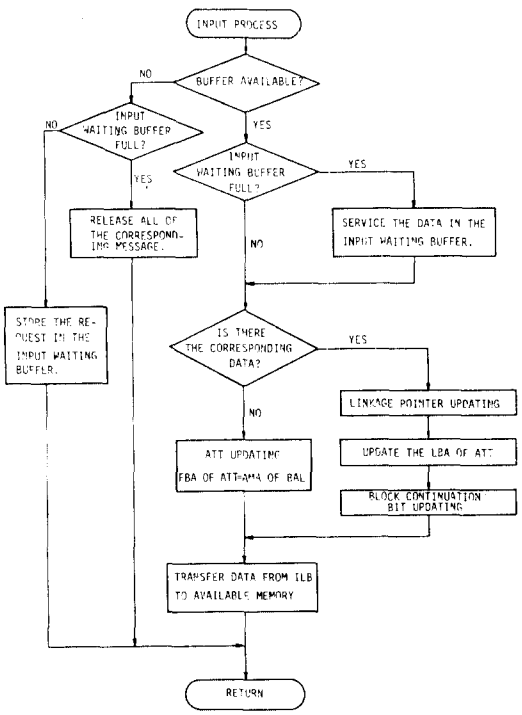

Fig. 5. Flowchart of butter management input process.

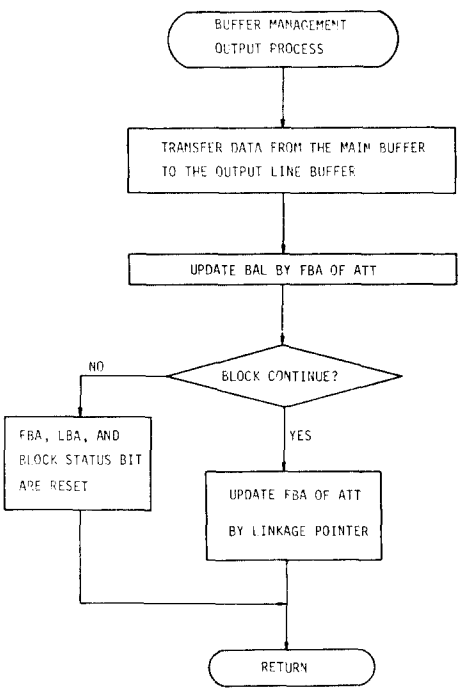

Fig. 6. Flowchart of buffer management output process.

4. Interrupt Service Routines

There are eight interrupt service routines which are related with synchronous and asynchronous transmission. The dybamic buffer management and X.25 link level process are included in the interrupt service routines that handle communication using Z-80A SIO. Also, the packet assembly/disassembly (PAD) function compatible with CCITT recommendations X.3 and X.28 is processed in the interrupt service routines handling Z-80A DART. Each interrupt service routine of synchronous and asynchronous transmission consists of transmitte interrupt, receive interrupt, external status interrupt, and special receive interrupt routines. In what follows, we first describe Z-80A SIO interrupt service routines related with X.25 link level process, and then explain Z-80A DART interrupt handling routines related with PAD functions.

1) X. 25 link level routines

The X-25 link level process can be classified into transmitter and receiver routines as shown in Fig. 7. The receiver routine includes the receiver, the high-level data link control (HDLC) receiver and the secondary routine. Also, the transmitter routine is composed of the sender, the HDLC transmitter, and the primary routine.

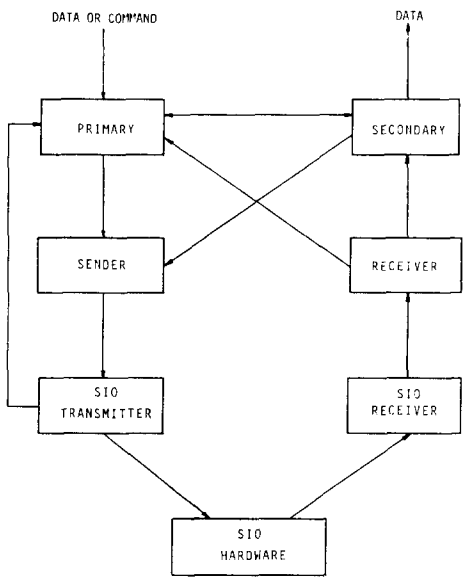

Fig. 7. Block diagram of X.25 link level.

Let us discuss first the functions of the receiver routine. In the HDLC receiver routine, characters are received from the HDLC hardware through SIO receive interrupts. This function is the same as that of the SIO routine interrupt routine, and as a result, it is hardare-dependent. When an SIO special receive interrupt occurs, the valid received frame is serviced by the receiver routine. The routine that will process the received frame is decided according to the frame type by the receiver routine. Command or data frames are serviced in the secondary routine, and response frames are handled in the primary and the sender routine. The secondary routine processes the incoming data or commands. Also, in the case of global changes, the message of the secondary routine is transferred to the primary routine. If the received frame is a valid information frame, the buffer management input process is requested. This process is similar to Fig. 5. When the received information frame is not valid, retransmission is requested. Besides, the response frame corresponding to the command is processed in the secondary routine, and this frame is serviced in turn in the sender routine.

In the primary routine, either the outgoing data or commands to be transmitted or the response frames received from the receiver routine are processed. Next, they are passed to the sender routine. Some of the parameters used in the primary routine, which controls the secondary, are shared with the secondary routine. The control byte in the data, command or response frame from the primary or secondary routine is decided in the sender routine. In addition, the related parameters are updated there. By the HDLC transmitter routine, the completed frame is transmitted to the HDLC hardware character by character through SIO transmitter interrupts. The function of this routine is almost the same as that of the SIO transmit interrupts.service

routine mentioned earlier. One can note that this routine is also hardware-dependent. The transmitted frame is not instantly released. Instead, it is stored in the output waiting buffer until it is acknowledged. The maximum number of the information frames to be acknowledgedged cannot be greater than seven. If the frame to be transmitted at this instant is a new information frame, the output process of buffer management is requested. Then, the services of the primary and the sender routine are required.

In the CCITT recommendation X.25 link level, the state diagram or state transition table is not given. On the other hand, the state transition table is described in the BX.25 link level protocol proposed by Bell Laboratories[5]. This BX.25 link level protocol is compatible with CCITT X.25 link level protocol. For this reason we have followed the BX. 25 link level protocol in implementing the X.25 link level process. In the SMUX system, all the command and data sequences of 16 states except for servicing the poll/final function have been considered.

2) PAD routines

A brief diagram of the overall PAD architecture is shown in Fig. 8. Each channel has six buffers-- an input line buffer (ILB), an output line buffer (OLB), a PAD command buffer, a PAD service buffer and an echo buffer. Although these buffers are assigned one for each channel, respectively, they are used by the routines that are shared by all channels.

There are many routines in the PAD as shown in Fig. 8. ANALYS alalyzes the PAD command signals, and converts the IA5 (International Alphabet No. 5) -- formatted into an internal message code, according to which SERV takes a necessary action, and sends the PAD service signal to the service buffer if necessary. LINK is responsible for physical level connection, and controls asynchronous modem signals. After the physical link has been set up, the PAD can receive characters from the terminal. When a character arrives at the asynchronous receiver/transmitter (ART) hardware, an interrupt is generated, which activates ARTRX. ARTRX fetches the character in the ART hardware, and stores it in the ILB or the command buffer according to the present state. If a character code other than IA5 is being used, the format of that character is converted into IA5 by CONVRX while being stored in the buffer. EX3 then acts on the received character, and checks whether it is one that corresponds to the PAD parameters. The checklist is used when EX3 takes action on the received character stored in the buffer by calling service subroutines such as ECHOIN, EDIT, and others. ECHOIN stores the character in the echo buffer.

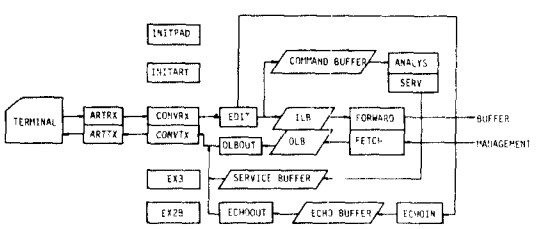

Fig. 8. Butters and routines in the PAD.

EDIT is used when the terminal user wants to edit one line of data or command before forwarding or execution. Editing is done properly on the ILB or the command buffer. If the ILB becomes full or any other forwarding condition occurs, FORWARD informs the monitor that the content of the buffer should be transferred to the main buffer as a frame.

ARTTX is an ART transmit interrupt service routine, which calls EX3 to service the character to be transmitted from the OLB, the service buffer or the echo buffer. The echo buffer has top priority in transmission. The characters in the OLB are fetched by OLBOUT. When the OLB becomes empty, FETCH informs the monitor that the buffer should be loaded with a new frame.

CONVTX converts the character that is about to be transmitted if a code other than IA5 (ASCII) is being used. ECHOOUT, which is called by EX3, fetches a character from the echo buffer. INITART and INITPAD

initialize the ART hardware and the PAD routines, respectively. EX28, which is a state machine that is invoked at every occurrence of an event, maintains and keeps track of the state between the terminal and the PAD. The event code is usually generated as a result of executing routines such as EX3, SERV or buffer management routines. This event, together with the present state, determines the next state to get into and the sequence of tasks to be done. At the end of the tasks, the present state is updated by a new state. Therefore, EX28 can be considered to be an important part of the PAD in that all actions should be taken in accordance with the sequence indicated by EX28.

Since the SMUX is a point-to-point communication equipment, all functions that are related with networking or virtual calls have been excluded in our implementation. In the SMUX system, all the PAD parameters defined by CCITT X.3 except for 8 (discard output) and 11 (line speed selection) have been implemented. The PAD parameter values implemented in the SMUX are shown in Table II.

Table II. PAD parameter values implemented in SMUX.

Parameter reference number	Function	Selectable values
1	PAD recall character	0, 1
2	Echo	0, 1
3	Data forwarding signal	0, 2
4	Idle timer delay	0, 1-255
5	Ancillary device control	0, 1
6	PAD service signals	0, 1, 4
7	Operation on a break signal	0, 8, 18
9	Padding after a carriage return	0-7
10	Line folding	0, 1-255
12	Flow control of the PAD	0, 1
13	Line feed insertion after a carriage return	0, 1, 2, 4, 3, 7
14	Padding after a line feed	0-7
15	Editing	0, 1
16	Character delete	0-127
17	Line delete	0-127
18	Line display	0-127

V. Conclusions

We have presented implementation of a microprocessor-based statistical multiplexer with particular emphasis on firmware development of the system. With efficient design of hardware and software, it was possible to realize the system that is capable of multiplexing 8 asynchronous channels simultaneously with a single Z-80A microprocessor. The SMUX system is mostly digital and software controlled. Z-80A DART and SIO have been used for the asynchronous and synchronous communications. Also, for efficient storage management, a dynamic buffer management algorithm is used. The developed software has been designed to be compatible with CCITT recommendations X.3, X.28 and X.25 link level. We believe that our system design presented is efficient, economical and unique in a sense.

To improve the performance of the SMUX, the use of a multiprocessor scheme can be considered for system modularity and expandability. Also, the direct memory access (DMA) to support synchronous communication and WD2511 LSI chip for X.25 link level process may be used to improve the SMUX system. Then, the SMUX can be used as a network concentrator in a public packet switching data network.

References

[1] N. Abramson and F.F. Kuo, *Computer Communication Networks*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.

[2] *CCITT Recommendations X.3, X.25 and*

X.28. Volume VIII-Fascicle VIII. 2, VIIth Plenary Assembly, Geneva, Nov., 1980.

[3] *CCITT Recommendations V.24 and V.28.* Volume VIII-Fascicle VIII.1, VIIth Plenary Assembly, Geneva, Nov,1980.

[4] W.W. Chu, *Dynamic Buffer Management for Computer Communications.* Proce - edings of the Third Data Communication Symposium, pp.68-72, Nov., 1973.

[5] *Operations Systems Network Communications Protocol Specification.* BX.25, Issue 3, Director - Purchased Products, June, 1982.