E. STĂICUȚ
Central Institute for Management
and Informatics—Bucharest, Romania

J. BARCHANSKI
Institute of Computer Control Systems,
Silesian Technical University, —Gliwice, Poland

Considerations on the Design of an Interface for Computer Networks

The purpose of this paper is to present some technical problems which are encountered in the design of a computer to-computer interface. The hardware and software interface necessary to connect a computer to a node of a communication network is presented. The description is computer-independent, that is, without regard to any specific computer system. The interface is mainly divided into two parts: one part is standard (the same for any type of computer), the other part depends on the particular computer which is to be connected to the communication network. It is presented the functions performed by the interface and how the main problems (co-ordination of asynchronous processes in the two computers, difference in word length, speed, etc.) are solved.

1. INTRODUCTION

The purpose of this paper is to present a functional description of an interface that seems appropriate for the connection of computers to a communication network. Generally, in a store-and-forward network, each node services several data processing computers (DPC) which communicate among themselves by means provided by the communication network, Figure 1. Connecting a DPC to a node requires a hardware and software interface, some part of which must be built into the node and another part into the DPC.

The node-DPC interface must allow message flow in both directions. Since various types of computers need be connected to a large scale network, a need arises to accomodate a variety of word lengths in different DPCs. To do that the interface allows the transmission of data serially, bit by bit. Serial transmission has also the advantage that it requires fewer lines than parallel transmission.

Typically, computers can handle data at high speed but are not always ready to do so. Furthermore, the DPCs and the node computer run asynchro-

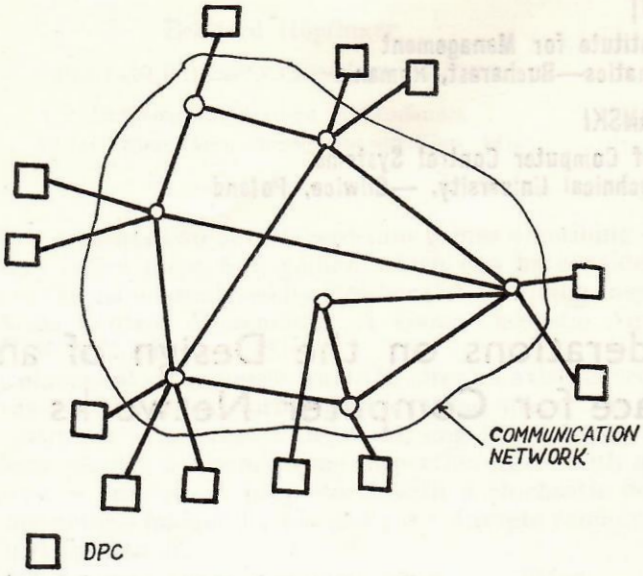


Fig. 1 *Store-and-forward computer network*

nously. To co-ordinate the transmission between DPC and node, a handshake procedure is used [7] — [12]. A transfer of data from sender to receiver takes place only when both sender and receiver have indicated preparesness. This permits either the sender or the receiver to hold up the transmission between any two transfers of data in order to take as much time as necessary to get a new word from memory, to store an assembled word into memory or to activate an interrupt routine that sets up new input or output buffers [17].

In designing the DPC-node interface, an optimal division of functions among its modules must be achieved.

2. HARDWARE INTERFACE

Depending on the level at which the control is performed by the handshake procedure, two types of interfaces are encoutered, the control being performed by message acknowledgment or bit acknowledgment.

2.1. *Interface based on message acknowledgment*

Such an interface is encountered in KUIPNET (Kyoto University Information Processing Network), [2] — [6]. The physical connection is realized by four lines, two lines for each direction, Figure 2. Three control signals (START, READY, END) are used to control the transfer of data in each direction.

Suppose that the DPC wants to send a message to the node. At first the DPC has to send the START signal to the node asking it to get ready to

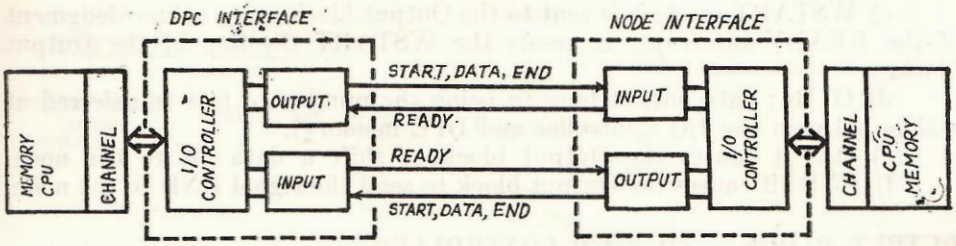


Fig. 2 *DPC-node interfaces*

receive a message. When the node is prepared to receive the data it replies with the signal READY. Only after the DPC has received the READY signal it sends data (bit by bit) to the node. At the end of the transfer the DPC sends the END signal to the node. The transmission from the node to the DPC is done in a similar way.

Since the DPC interface and the node interface have similar structure, only the DPC interface is described below. The interface is divided into three sections: I/O Controller, Output and Input. The Output and Input sections are independent of the type of the computer to be connected.

The functions of the I/O Controller are as follows:

— to generate the signals corresponding to the CPU instructions,

— to exchange the control signals and data between the channel and Input and Output sections, and

— to test the status of the Input and Output sections and to convey the status information to the CPU on its demand.

Four flip-flops are mentioned in the description in the following paragraphs. Thus, the Output block has two flip-flops called WRITE and WSTART. The WRITE flip-flop indicates that the Output block is active. It is set when the Output block sends a START control signal to the node and it is reset when an END control signal is sent to the node. The WSTART flip-flop shows that the READY control signal has arrived from the node.

The Input block has two flip-flops called RSTART and READ. The RSTART flip-flop is set when a START control signal comes from the node and reset after the DPC has answered successfully to the interrupt caused by the START signal. The READ flip-flop indicates that the Input block is active. It is set when the Input block sends a READY signal to the node indicating that it is ready to receive data. It is reset when an END control signal comes from the node.

The signals transferred between the I/O Controller and the Input and Output blocks are as follows:

I/O CONTROLLER — TO — OUTPUT BLOCK

a) INIT: it initializes the Output block; namely it resets all its flip-flops.

b) WRITE set: it causes the Output block to send the START signal to the node and sets the WRITE flip-flop of the Output block.

c) WSTART reset: it is sent to the Output block as an acknowledgment of the READY interrupt. It resets the WSTART flip-flop of the Output block.

d) OTB*n*: data output lines (*n* being the number of bits transferred at a time between the I/O Controller and DPC memory).

e) DA: it causes the Output block to shift a data bit to the node.

f) END: it causes the Output block to send the signal END to the node.

OUTPUT BLOCK — TO — I/O CONTROLLER

a) READY: it is the result of the arrival of the READY signal from the node. It causes an interrupt of the DPC.

b) WREQ: it requests the transfer of a new word from DPC memory.

c) WSTART: it is the output of the WSTART flip-flop which shows that the READY signal has arrived from the node. It is necessary to be reset by WSTART reset as above.

d) WRITE: it is the output of the WRITE flip-flop which indicates that the Output block is active. It is set by the above WSTART. When the END signal is sent to the node, it is automatically reset.

INPUT BLOCK — TO — I/O CONTROLLER

a) RSTART: it is the result of the arrival of the START signal from the node, it causes the interruption of the DPC. It is necessary to be reset by the RSTART reset signal as below.

b) READ: it is the output of the READ flip-flop indicating that the Input block is active. This flip flop is set by the READ set and reset by the REND signal when the END signal is sent by the node.

c) INB*n*: input data lines.

d) RREQ: it indicates that data sent by the node has already arrived.

e) PARITY: it indicates that the data which came from the node is in error.

f) REND: it is caused by the fulfilment of two conditions: the arrival of the END signal from the node and the transfer of all data arrived previously from the node into the DPC memory. It is reset by REND reset signal presented below.

I/O CONTROLLER — TO — INPUT BLOCK

a) INIT: it initializes the Input block.

b) RSTART reset: it resets RSTART flip-flop as an acknowledgment of the interruption caused by the arrival of the START signal from the node.

c) READ set: it sets the READ flip-flop causing the transmittal of the READY signal to the node.

d) REND reset: it resets the REND flip-flop as an acknowledgment of the completion of the interruption caused by the END control signal coming from the node.

e) DR: it indicates that data has already been transferred into DPC memory.

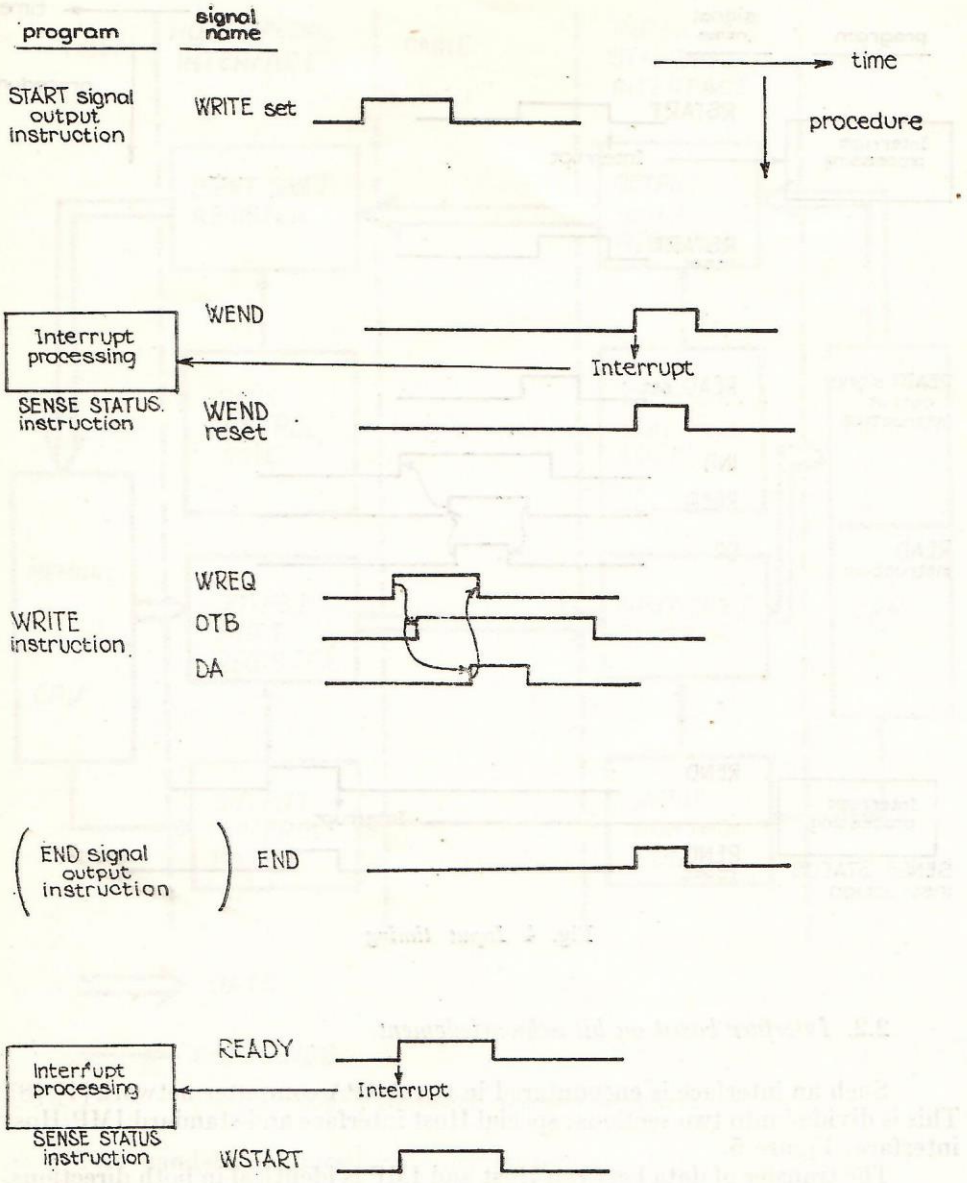


Fig. 3 *Output timing*

Since the DPC and node computer are running asynchronously a situation may arise when both the DPC and the node transmit a START signal simultaneously. In this case the DPC has priority in transmitting its message.

In order to make it clearer how the above described signals operate on each other, some of them are represented in Figures 3 and 4.

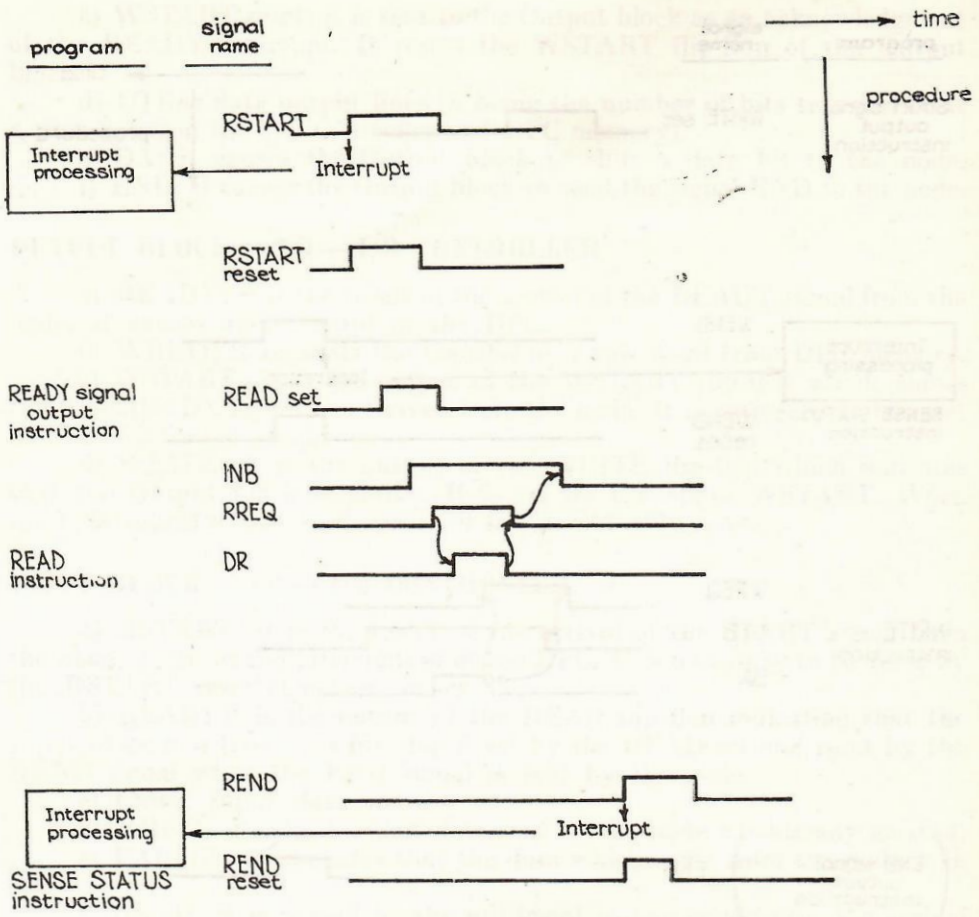


Fig. 4 *Input timing*

2.2. *Interface based on bit acknowledgment*

Such an interface is encountered in the ARPA computer network [1], [8]. This is divided into two sections: special Host interface and standard IMP/Host interface, Figure 5.

The transfer of data between Host and IMP is identical in both directions. For example, the transfer of data from Host to IMP is done as follows. The data are taken from Host memory word by word and stored in the Output register of the special Host interface from where they are transmitted to the standard IMP/Host interface serially, bit by bit. The bit transmission is asynchronous, the transfer of each bit is controlled by two signals "Ready for next bit" and "There's your bit". The end of message is indicated by the signal "Last bit".

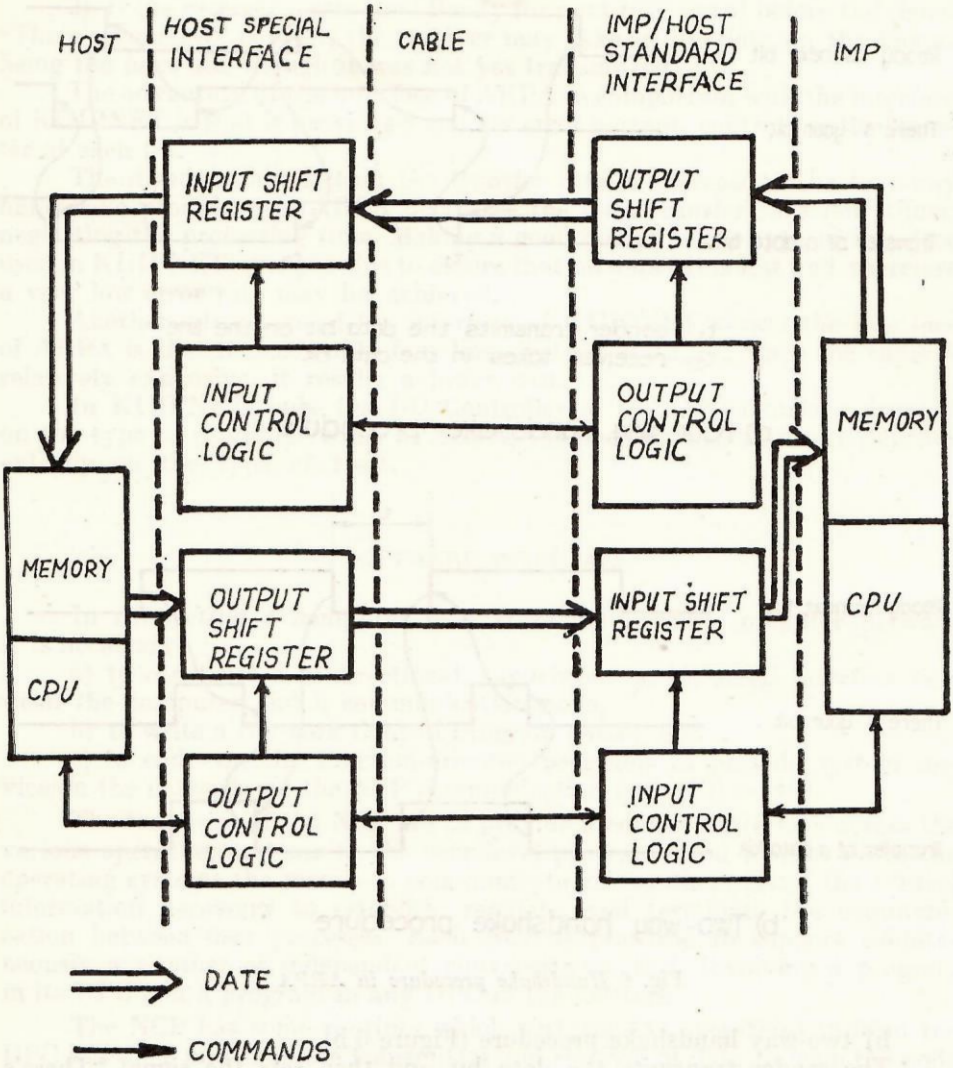
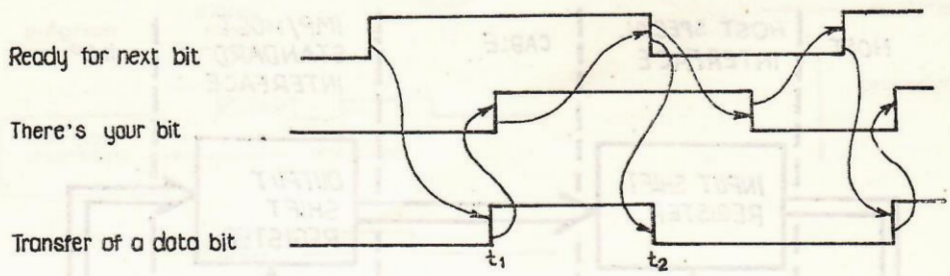


Fig. 5 *Host-IMP interface in ARPA*

Two handshake procedures may be used:

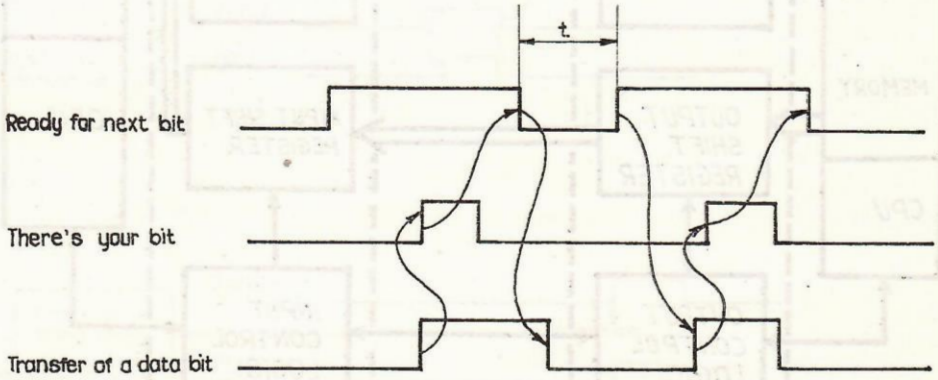
a) four-way handshake procedure (Figure 6 a).

The sender transmits the data bit, then sets the signal "There's your bit" and sends it to the receiver (first transit on the line). The receiver takes in the data bit and acknowledges the sender by resetting the signal "Ready for next bit" (the second line transit). The sender answers by resetting the signal "There's your bit" (the third line transit) and then the receiver sets the signal "Ready for next bit" which informs the sender (fourth line transit) that it can transmit the next bit.



t_1 - sender transmits the data bit on the line
t_2 - receiver takes in the data bit

a) Four-way handshake procedure



b) Two-way handshake procedure

Fig. 6 *Handshake procedure in ARPA*

b) two-way handshake procedure (Figure 6 b).

The sender transmits the data bit and then sets the signal "There's your bit" which is sent to the receiver (the first line transit). The receiver takes in the data and acknowledges the sender by resetting the signal "Ready for next bit" (the second line transit). Then, instead to wait the arrival of this to the sender and the propagation of "There's your bit" signal resetting to the receiver, the receiver maintains the signal "Ready for next bit" off for a short time t and then turns it on.

This procedure is simpler and easier to implement than the first one. However, it has two desadvantages:

1) If the signal "Ready for next bit" is maintained off for a time too short so that the sender does not notice it, it will continue to wait for the receiver to take in the data bit.

2) If the receiver resets the "Ready for next bit" signal before the signal "There's your bit" is reset, the receiver may take in the data on the line as being the next bit, though it was not yet transmitted.

The advantage of the interface of ARPA in comparison with the interface of KUIPNET is that it provides a smaller error percent, controlling the transfer of each bit.

The disadvantage is that the transfer rate is decreased. The four-way handshake procedure of ARPA decreases the data transfer rate four times, neglecting the processing time. Making a good design of the type of interface used in KUIPNET, it is possible to assure that no data bit is lost and therefore a very low error rate may be achieved.

Another advantage of the interface of KUIPNET versus the interface of ARPA is that it needs only four lines instead of 12 [8]. Since the cable is relatively expensive, it results a lower cost.

In KUIPNET, only the I/O Controller of the DPC interface depends on the type of computer while in ARPA, the special Host interface depends entirely on the type of Host.

3. SOFTWARE INTERFACE

In connecting a computer to a store-and-forward computer network it is necessary:

a) to construct a bi-directional, asynchronous, bit serial interface between the computer and a communication node,

b) to write a Network Control Program (NCP), and

c) to code specific function-oriented programs to provide system service on the network via the NCP communication path [13] — [16].

The functions of the NCP are to provide a common interface across the various operating systems to the user level processes, and to provide to the operating systems the means to communicate among themselves, the control information necessary to establish, regulate, and terminate the communication between user processes. Each NCP is prepared to support simultaneously a number of independent conversations, each involving a program in its DPC and a program in any DPC in the network.

The NCP has some routines which perform I/O operations to/from the DPC interface and thus the transmittal and receival of data to/from the node is achieved [16].

Various approaches can be taken in the implementation of the NCP. It may be a part of the operating system or it may run as an user job. In either case the NCP is the only task within the system with access to the DPC interface. Because the NCP does not compete with other tasks for use of these devices, the NCP need not relegate the responsibility of scheduling input/output operations to the operating system. Therefore, the NCP executes I/O operations to the DPC interface directly, Figure 7.

When the DPC interface receives a START, READY or END control signal from the node, it interrupts the DPC system. These interrupts are processed by the NCP which instates itself as the system's first-level interrupt

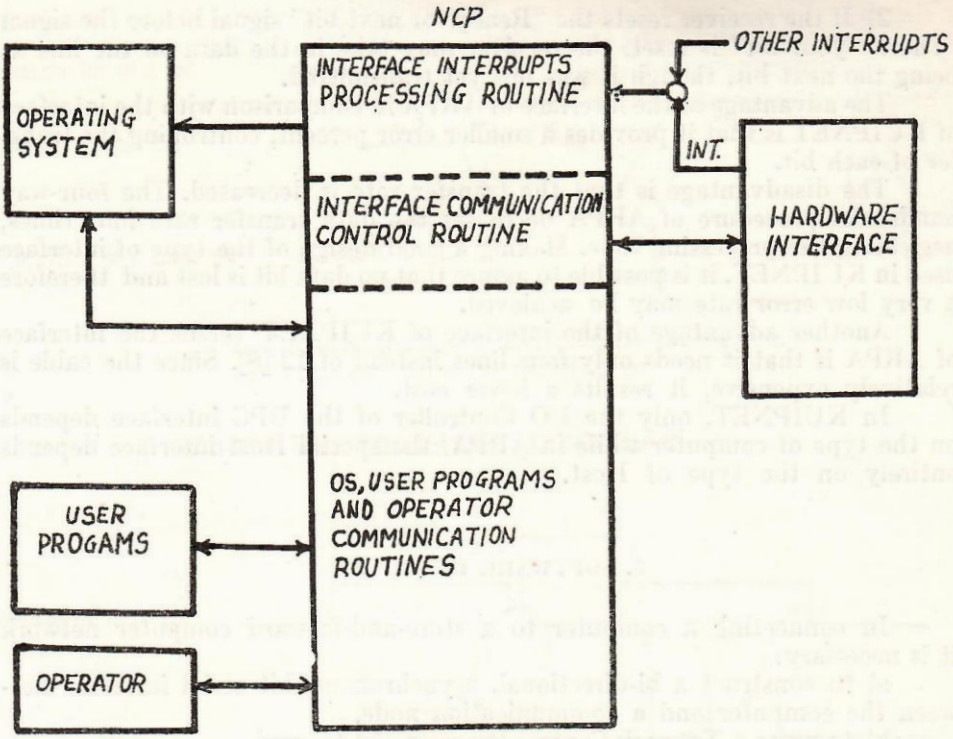


Fig. 7 *Communication between NCP and other modules*

handler. In this capacity, it examines each I/O interrupt which occurs, trapping and processing those of concern to it, and allowing the operating system to process the others. Handling I/O in this fashion has the following attributes:

a) it requires no support software in the operating system,

b) it is efficient because, driven directly by interrupts and not requiring the dispatching of a task at the conclusion of each operation, input can be processed quickly and output initiated as soon as the interrupt for the previous output operation has been received.

4. DPC-to-NODE MESSAGE TRANSFER

Messages flowing from DPC to the node and from the node to the DPC may be categorized as: user messages, NCP control commands or some irregular messages [16].

The NCP has a flag which indicates whether the NCP buffer is busy or empty. First operation is to set this flag to indicate that the NCP buffer is busy. Then the NCP initiates message transfer by informing the DPC hardware interface of the initial address of the message in the DPC memory and about

its length (in words). This is done by executing I/O instructions. NCP instructs the hardware interface to send START control signal to the node, then it stops running and the control is given to the operating system. After that a new task may start running or the monitor may run in an idle loop. When the READY signal arrives, an interruption, occurs, the NCP starts running, it examines the status word to see what interruption has occured, and after finding READY interrupt it instructs the hardware interface to start message transmission to the node. After the message is transmitted, the DPC is interrupted, the NCP starts running and instructs the hardware interface to send the END control signal to the node. Then the NCP buffer flag is set to indicate empty buffer so that the NCP may store another message in its buffer and initiate the message transfer procedure when it is requested by user programs. Thereafter, the control is returned to monitor.

5. NODE-to-DPC MESSAGE TRANSFER

In this section we present only the operations executed by the DPC's operating system, NCP and hardware interface when a message comes from the node. The node performs operations similar to those discussed in section 4 to transmit a message to a DPC directly connected to it.

When the DPC hardware interface receives the START control signal coming from the node an interrupt occurs, the NCP starts running and performs a test for simultaneous START. If there is simultaneous START, the NCP returns the control to the operating system because the DPC has priority on the node in transmitting its message. Thereafter, the DPC does not return READY signal and waits for it to come from the node in order to start the transmission of its message. If there is not simultaneous START and the NCP buffer is busy the DPC cannot receive the message from the node and the READY control signal is not sent to the node. If the NCP buffer is empty, the NCP prepares to receive the message from the node. Thus the NCP buffer flag is set to indicate that the NCP buffer is busy and it makes known the initial address and the length of the buffer to DPC hardware interface. Then, the NCP stops running and the control is given to the operating system.

After receiving the READY control signal the node transmits the message which is automatically stored in the DPC memory at the address the DPC hardware interface was previously instructed.

When the transmission is over the END control signal comes and interrupts the DPC. The NCP starts running and examines the message arrived.

6. CONCLUDING REMARKS

In connecting a computer to a store-and-forward computer network, it is necessary to build a hardware and software interface with the communication network, and also to write a software module to co-ordinate the communication between that computer and other computers in the network.

This paper presents the hardware and software interface. Several interface structures are presented together with their advantages and desadvantages. The interface structure is described without referring to a certain type of computer and also the possibility to connect various types of computers to the network is provided. To connect computers with various word lengths, the data are transfered serially and to co-ordinate the asynchronous process performed by the receiver and sender, a handshake procedure is used.

ACKNOWLEDGMENT

The authors wish to thank Professor Toshiyuki Sakai of the Department of Information Science of Kyoto University, Japan, for the possibility to study problems of computer communications in his laboratory. Due to the help of the other laboratory members it was possible to get acquainted with the KUIPNET and to implement the high level protocols mentioned in the references.

REFERENCES

1. * * *: AFIPS Spring Joint Computer Conference Proceedings. vol. 36, pp. 543—597, 1970; vol. 40, 1972, pp. 243—298.
2. BARCHANSKI J.A., STAICUT E.: *General Aspects of Computer Networks*; Podstawy Sterowania, Tom. 5, no. 3, pp. 269—284, 1975.
3. SAKAI T. et al.: *Inhouse Computer Network for Information Processing and some Applications.* Seminar on Computer Assisted Chemical Research Design, Honolulu, U.S.A.
4. SAKAI T., et al.: About KUIPNET—*An Inhouse Computer Network.* Technical Report of the Professional Group on Electronics Computers of the IECEJ. EC 73—53, (in Japanese).
5. OHNISHI H.: *Inhouse Computer Network and its Sofware.* Thesis for the MSc. degree of the Dept. of Information Science, Kyoto University, March 1974, (in Japanese).
6. SAKAI T., et al.: *Inhouse Computer Network KUIPNET for Information Processing.* PACNET Symposium, Sendai, Japan, August 20—22, 1975.
7. DAVIES D.W., BARBER D.L.A.: *Communication Networks for Computers.* John Wiley & Sons Ltd., 1973.
8. * * *: *BBN Report 1822*, U.S.A.
9. * * *: *British Standard 4421.* British Standards Institutions, British Standard House, 2 Park Street, London Wl.
10. DAVIES D.W.: *Communication Networks to Serve Rapid-Response Computers.* Proc. IFIP Congress, Edinburg, pp. 650—658, 1968.
11. FRASER A.G.: *On the Interface Between Computers and Data Communications Systems.* Communications of the ACM, vol. 15, no. 7, pp. 566—573, July 1972.
12. FRASER A.G.: *Spider- A Data Communications Experiment.* Computing Science Technical Report 23, Bell Laboratories, USA.
13. BARCHANSKI J.A., SAKAI T, STAICUT E., HAYASHI T.: *User-Level Protocols for the KUIPENT.* Pacific Area Computer Network (PACNET) Symposium, Sendai, Japan, August 20—22, 1975.
14. STAICUT E., BARCHANSKI J.A.: *Specifications and User's Manuals for MELCOM 70 TELNET and FTP, MACC 7/F TELNET and FTP, MACC 7/F Picture Access and Transfer Protocol.* Internal Reports, Sakai Laboratory, Dept. of Information Science, Kyoto University, Japan.
15. BARCHANSKI J.A., STAICUT E.: *Interfacing and Protocols for a Computer Network.* Podstawy Strrowania, Tom 6, no. 1, pp. 103—116, 1976.

16. BARCHANSKI J.A., STAICUT E.: *On Implementation of the Network Control Program for a Computer Network.* to be published in Podstawy Sterowania, Poland.
17. IWAMA K.: A *Two-Input/Output-Pair Automata and Its Applications to the Design of Computer Connections.* Master Thesis of the Dept. of Information Science, Kyoto Univ. Japan, 1975.

1. 06. 1976

CONSIDERATION SUR LA PROJECTION D'UN INTERFACE POUR LES RESEAUX D'ORDINATEURS

Le but de cet article est celui de présenter quelques problèmes techniques, rencontrés dans la projection d'un interface entre les ordinateurs. On traite tant de l'interface hardware que software, nécessaires à la connexion d'un ordinateur à un noeud du réseau de communications pour ordinateurs. La description ne tient pas compte de l'ordinateur , c'est-à-dire qu'on n'a pas en vue un type exprès d'ordinateur. En principe, l'interface se divise en deux parties, dont l'une est standard (la même, quel que soit le type d'ordinateur) et l'autre qui dépend de l'ordinateur particulier devant être connecté au réseau de communication. Dans cet article on présente les fonctions que l'interface effectue, ainsi que la modalité de résoudre les principaux conflits (coordination des processus asynchrones de deux ordinateurs, longueurs différentes des mots, vitesses différentes, etc.).

СООБРАЖЕНИЯ ПО ПОВОДУ ПРОЕКТИРОВАНИЯ ИНТЕРФЕЙСА ДЛЯ ВЫЧИСЛИТЕЛЬНОЙ СЕТИ

Целью данной работы является представление технической проблемы, встречающейся в проекте инерфейса между счетными машинами.

Представлен интерфейс „hardware and software", необходимый для соединения счетной машины с узлом сети связи. Описание дается независимо от счетной машины, т.е. не имеется в виду определенная специфическая система счетных машин.

Интерфейс в принципе разделен на две части: первая часть является стандартом (для любого типа счетных машин), другая часть зависит от определенной счетной машины, которую предполагается присоединить к сети связи.

Представлены функции, выполненые интерфейсом и способ разрешения основных вопросов (координация асинхронных процессов в двух счетных машинах, разница длины слова, скорость и т.д.).