REPORT R-739 SEPTEMBER, 1976 UILU-ENG 76-2227

CSL COORDINATED SCIENCE LABORATORY

DESIGN AND OPERATION OF A HIGH LEVEL MULTICOMPUTER COMMUNICATIONS PACKAGE

PAUL ROBERT BODENSTAB

UNIVERSITY OF ILLINOIS – URBANA, ILLINOIS

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* DESIGN AND OPERATION OF A HIGH LEVEL MULTI-COMPUTER COMMUNICATIONS PACKAGE		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER R-739; UILU-ENG 76-2227
7. AUTHOR(s) Paul Robert Bodenstab		8. CONTRACT OR GRANT NUMBER(s) DAAB-07-72-C-0259
9. PERFORMING ORGANIZATION NAME AND ADDRESS Coordinated Science Laboratory University of Illinois at Urbana-Champaign Urbana, Illinois 61801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Joint Services Electronics Program		12. REPORT DATE September, 1976 13. NUMBER OF PAGES 35
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Multicomputer Communications
Concurrent Processes
Mutual Exclusion
Critical Sections

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The need for a means of communication between tasks in either a multiprogramming or time-sharing environment becomes necessary when one wants to coordinate the activities of several real time interdependent tasks. Likewise, when a computing system is composed of several interconnected computers the same need for communication among them arises. In the Coordinated Science Laboratory at the University of Illinois, the computing system is composed of Digital Equipment Corporation's PDP-10 and PDP-11/40 computers connected together via a recently built high speed (approx. one megabyte per second) channel

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. ABSTRACT (continued)

interface. Due to the Advanced Automation Laboratory's current interest in machine vision and industrial robotics which will require the execution of extensive interdependent control tasks on both computers, the necessity for a high level communication package which will allow any task on one computer with any task on the other computer is readily apparent.

UILU-ENG 76-2227

DESIGN AND OPERATION OF A HIGH LEVEL MULTICOMPUTER COMMUNICATIONS PACKAGE

by

Paul Robert Bodenstab

This work was supported in part by the Joint Services Electronics Program (U.S. Army, U.S. Navy and U.S. Air Force) under Contract DAAB-07-72-C-0259.

TABLE OF CONTENTS

DESIGN AND OPERATION OF
A HIGH LEVEL MULTICOMPUTER COMMUNICATIONS PACKAGE

Paul Robert Bodenstab

1. INTRODUCTION

The need for a means of communication between tasks in either a multiprogramming or time-sharing environment becomes necessary when one wants to coordinate the activities of several real time interdependent tasks. Likewise, when a computing system is composed of several interconnected computers the same need for communication among them arises. In the Coordinated Science Laboratory at the University of Illinois, the computing system is composed of Digital Equipment Corporation's PDP-10 and PDP-11/40 computers connected together via a recently built high speed (approx. one megabyte per second) channel interface [1]. Due to the Advanced Automation Laboratory's current interest in machine vision and industrial robotics which will require the execution of extensive interdependent control tasks on both computers, the necessity for a high level communication package which will allow any task on one computer to communicate with any task on the other computer is readily apparent.

---

[1] See reference 10.

2. DESIGN OBJECTIVES AND CONSIDERATIONS

The concept for a multicomputer communications package (MCCP) is not new. However, for a particular package to be effective it must not only exploit as many hardware and/or software features as possible of the system configuration for which it is intended but it should also interface well at the user level and with the working environment. There are problems, however, implementing such considerations.

The foremost problem is the difference of computers and the need of the MCCP to interface with both operating systems. The PDP-10 supports a large time-sharing system while the PDP-11/40 supports a customized multiprogramming operating system [1]. The two computers differ also in their characteristic word size. Although the PDP-10 handles 36 bit words and the PDP-11 uses 16 bit words this difference is not as significant as it may appear since the channel interface hardware allows several modes of data packing and unpacking to occur between the two machines [2]. There still exists a problem of efficient preformatting and/or postformatting of messages once an intracomputer transfer has been made.

E. Dijkstra [3] has shown that a fundamental condition for intertask communication requires that they share a common store, i.e., memory location(s) that are accessible by each task. In order to prevent deadlocks or other time dependent problems associated with the accessing of this resource among several competing tasks, Dijkstra has shown that

---

[1] See references (2) and (7)
[2] See reference (10)
[3] See reference (5)

mutual exclusion must be achieved. This idea simply asserts that a sharable resource must be used exclusively by one task at a time. Satisfaction of this principle requires that no two or more tasks enter their critical sections [4] at the same time and this further implies that processes be queued in regard to the execution of their critical sections. Hansen [5] has determined the necessary conditions which any scheduling algorithm must meet:

1. The resource in question can be used by one process at a time at most.

2. When the resource is requested simultaneously by several processes it must be granted to one of them within a finite time.

3. When a process aquires the resource, the process must release it again within a finite time. Apart from this, no assumption is made about the relative speeds of the processes; processes may even be stopped when they are not using the resource.

4. A process should not consume processing time when it is waiting to aquire the resource.

Thus, these conditions must also be met by any general communications package.

Additional problems arise when an objective of the communication package includes a high level user and working environment interface. The calling protocol to initiate a message transfer or retrieval should be consistent, or nearly so, on both computers to facilitate ease of use.

---

[4] A critical section is simply a commonly shared program stream that accesses a resource

[5] See Chapter 3 in reference (6)

Equally important, the execution speed of the MCCP should be made as fast as possible since it will be used in the course of controlling real time events.

Considering the programming of the MCCP itself, a great deal of thought should be given to writing structured code that will lend itself to simpler debugging procedures and to future expandability if the need arises.

3. SOFTWARE ARCHITECTURE

An attempt has been made to make the software architecture comprising the MCCP as symmetrical as possible on the two computers. This would not only convey an easier understanding of the philosophy behind the communication mechanisms but would also facilitate in the actual programming since the basic communication algorithms would be the same while the actual coding of these algorithms would be different. In reality, the attempt was only partially successful since portions of the communication package had to be custom written to interface with the particular operating system. Still, when viewed from a system level, the architecture as shown in Figure 3-1 retains much symmetry.

The MCCP is composed of basically four parts, two interprocess communication facilities (IPCF) [1] and two intracomputer communication facilities (ICCF) where one of each exists on each computer. The function of each IPCF is to provide an easily used facility for allowing intertask communication on the same computer. The function of each ICCF is to provide a facility for allowing intra-IPCF communication. This basic structure, then, is capable of allowing any task to communicate with any other task regardless of whether the destination task exists on the same computer or the other.

The operation of the IPCF is very analogous to the operation of a post office, it must be able to receive messages and route them to the

---

[1] The acronym IPCF was first conceived and used by Digital Equipment Corp. in reference (2)

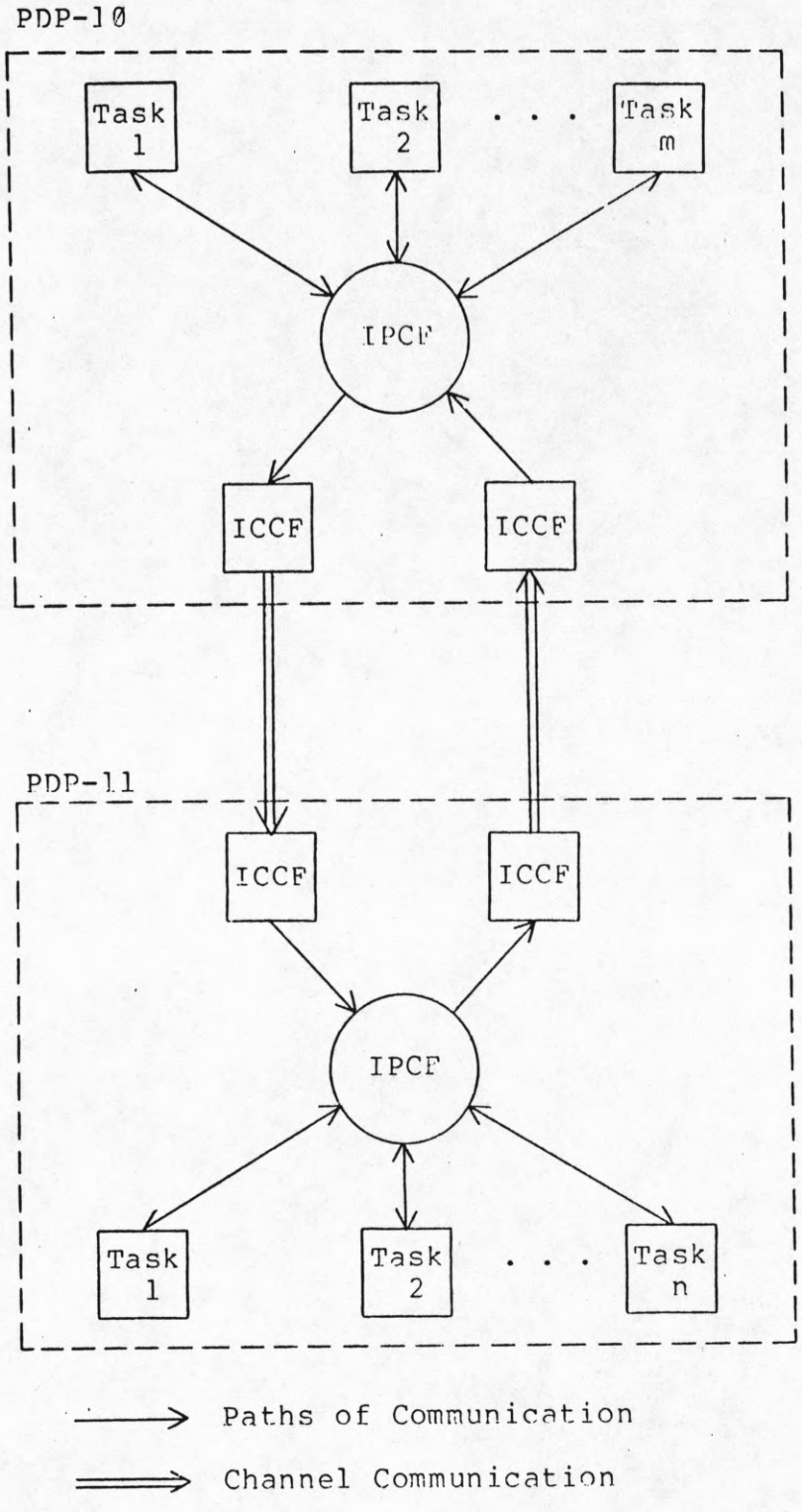

Figure 3-1. Software Architecture of the MCCP.

appropriate destination task. The ICCF consists primarily of channel driving and receiving routines and is used to relay messages dispatched by or intended for a particular IPCF.

The lowest level channel software has already been written prior to the completion of this project. The level of this software is compatible with both the standard DEC I/O UUO monitor calls on the PDP-10 side and with the I/O emulator trap routines on the PDP-11 side.

4. COMMUNICATION FACILITIES

In this chapter the discussion focuses on the actual algorithms and control mechanisms used to implement the MCCP.

4.1 IPCF

As was previously mentioned, the IPCF serves as a facility for transferring messages from one task to another on the same computer. As such, it represents the essential foundation on which the rest of the MCCP is based.

4.1.1 PDP-10 IPCF

The PDP-10 IPCF already exists as a standard feature of the DEC system-10 monitor [1]. It is supported by three monitor calls (UUOs) which allow a user to either send, receive, or query for the presence of a message. A message is a semistructured packet of information which consists of what might be analogous to an "envelope" and "data" portions. The envelope portion consists of a structured four word control block that contains such information as source and destination addresses, the length and a pointer to the data portion, and some assorted flags. The data portion consists of up to a 12 word block of user defined data.

The protocol for using the IPCF requires a user job to declare an ASCIZ [2] symbolic name and aquire a process ID (PID) through the [SYSTEM] INFO facility of the IPCF. PIDs serve as the source and

---

[1] Since this facility is well documented in Chapter 7 of reference (2) only a brief description of its features are presented here

[2] An ASCIZ string is equivalent to an ASCII string terminated by a null character

destination addresses used in the envelope portion of the message packet. The [SYSTEM] INFO facility can also be used for such purposes as finding the names associated with other PIDs, finding the PIDs associated with other names, etc.... A feature of the IPCF which allows a job to have several PIDs will be shown in a later section to have a crucial role in the operation of the MCCP.

Intertask communication is accomplished through the use of a "mailbox" in the form of a short linear first in-first out (FIFO) queue created by the monitor for each job that requests a PID from [SYSTEM] INFO. Message packets are placed in the receiver job queue and remain there until that job retrieves them.

4.2.2 PDP-11 IPCF

When the PDP-11 multiprogramming operating system was written no provision was made for allowing easy and systematic intertask commui-cation. Thus the creating of the PDP-11 IPCF was necessary to overcome this handicap. It was decided that the structure and operational protocol of this IPCF should be designed to mimic, with certain limitations, the behavior and structure of the PDP-10's IPCF. This decision implied that the overall characteristics and usage protocol of the entire MCCP would become consistent on both computers. Since the PDP-10 IPCF has been found quite easy to use then the ease of use of the overall MCCP should follow likewise.

This decision also suggested a possible software structure which satisfied the above objectives and yet could be easily partitioned into

well understood functional modules. They were: a shared data base to be used as the common store, and three routines similar to the PDP-10's IPCF send, query, and receive UUOs which perform various operations on this data base. This, indeed, was the approach taken.

To insure reliable intertask communication the mechanisms within the IPCF use a combination of hardware and operating system features to guarantee both mutual exclusion and the automatic scheduling of each task's critical section.

A simple and effective way to insure mutual exclusion of the sharable resource is to force the removal of the critical section from a user's program. By keeping the location of the data base unknown to the user and by requiring all users to use standard system routines to gain access to the resource only in an indirect manner then system integrity can be insured. It follows then that this philosophy causes these entire system routines themselves to become critical sections.

The PDP-11 hardware allows a simple way to handle these routines. Instead of using true reentrant subroutines, software interrupt routines are used instead (these routines are called via the TRAP instruction). Mutual exclusion results from the fact that once an interrupt occurs the computer hardware can be made to temporarily ignore further interrupt requests. After an interrupt, the hardware pushes the old program counter word (PC) and program status word (PS) onto a stack and reloads the PC and PS with words from a specific interrupt vector location. If the new PS causes the interrupt routine to run at a higher priority than other user tasks, then all pending interrupts of lower priority are temporarily ignored until

the trap routine finishes execution and the old PC and PS are restored. Thus, this characteristic also insures the automatic scheluding of execution of each trap routine (which are critical sections).

Another way of accomplishing this goal but without having to change program priority is to use a powerful operating system command which can stop multitasking and cause the processor to execute only the task which issued the command [3]. Thus, once a trap routine is entered, it can request exclusive use of the processor for the duration of the routine. Just before the return from interrupt is made, the routine can request that normal time slicing resume.

The latter approach was taken in the course of the IPCF development.

The use of a data base and the general protocol necessary to access parts of it implies two levels of isolation between any two communicating tasks. First, the task that wishes to send a message to another task must set up his message in his own memory space and then invoke the IPCF send routine which copies the message into a portion of the data base. Second, the destination task must invoke the IPCF receive routine which copies its current message in the data base into the task's memory space. It might be felt that such double buffering would be unnecessarily slow and possibly jeopardize the objective of controlling real time tasks. However, since a user message is restricted to being 32 words or less the time consumed in moving the message back and forth is negligible even in a potentially high use environment.

---

[3] See .LOCK and .UNLOCK EMTs in reference (8)

By developing an efficient queue structure which groups together messages for a common destination task and by using dynamic storage allocation algorithms for controlling the data base, an efficient and high speed message handler has been developed. The queue structure is diagrammed in Figure 4-1.

Whereas the PDP-10 uses PIDs as source and destination addresses for message packets, the PDP-11 uses only user declared three character Radix-50 [4] symbolic names. A master task table contains the user declared task name (analogous to a PID) and a pointer to the current message buffer for each task that requests use of the IPCF. The message buffers themselves form an easily manipulated buffer ring [5]. The addition or deletion of a message packet to or form the data base simply requires modification of the buffer pointers and updating the storage allocation routines to either get or free the buffer area. Therefore successive message buffers for any given task do not have to occupy successive memory locations in the data base. Thus the integrity of the queue structure is free from any problems due to either time dependent behavior of the tasks using the IPCF or to varying lengths of message packets.

In the following sections an explanation of operation is given for each of the four user accessible IPCF routines.

---

[4] For a definition of Radix-50 see reference (3)
[5] See Chapter 11 in reference (9)

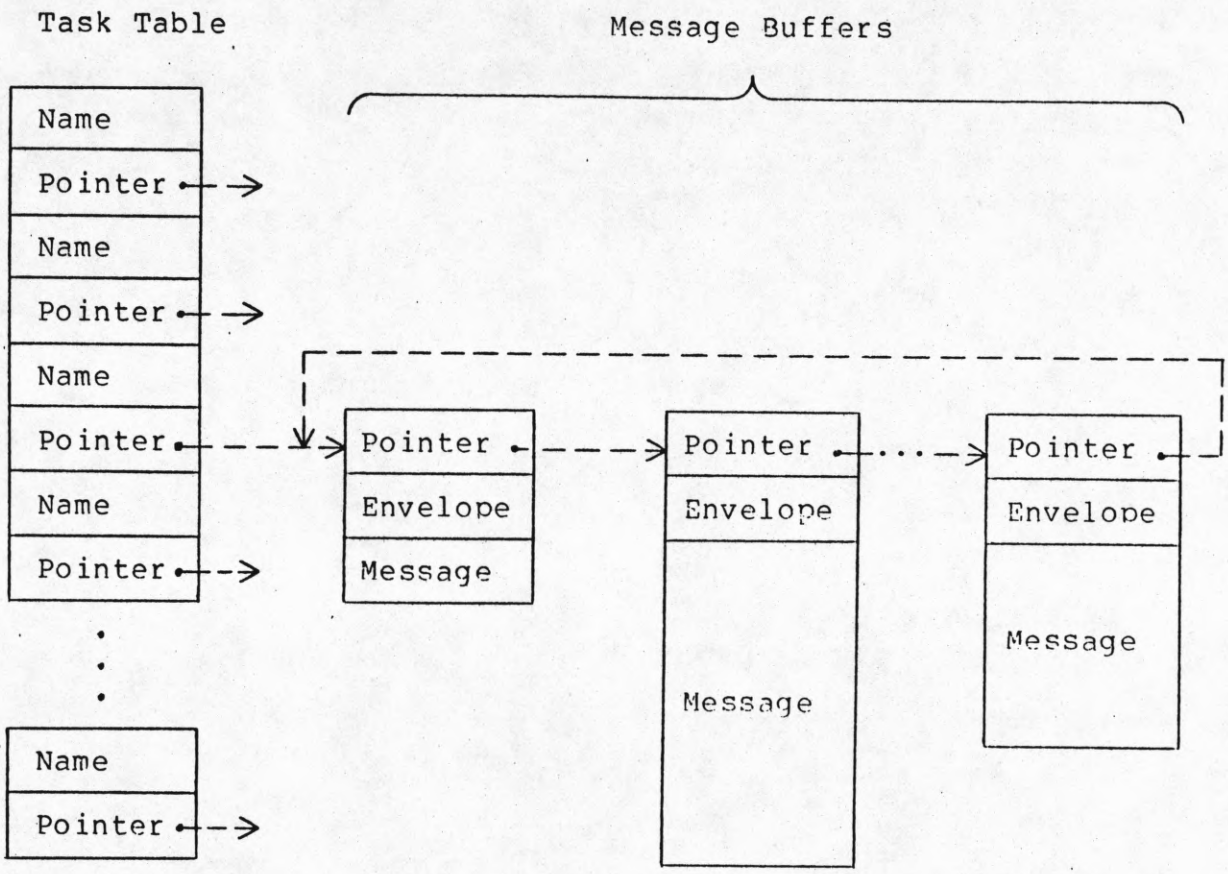

Figure 4-1. Queue Structure.

4.1.2.1 Initialization Routine

This routine is provided to reinitialize the entire IPCF package. It clears the task table and the entire data base thereby erasing any messages that still existed there. This routine is intended to be called by either a system task or by a single user root task. Due to its extreme nature it is intended to be executed only once, preferably, after the operating system has crashed. Normally, user programs do not have to use it.

4.1.2.2 Send Routine

The send routine takes a user message and routes it to the appropriate receiver. A message can be sent to three general destinations: to another task on the PDP-11, to some task on the PDP-10, or to the PDP-11's IPCF [SYSTEM] INFO facility. If the destination address specified in the envelope portion of the message packet is zero then the message is sent to [SYSTEM] INFO, otherwise a search is performed on the task table to see if the given destination address is indeed a valid one. If it is, room for the user's message in the data base is made via the storage allocation routines, copied there, the appropriate buffer pointers are updated, and a return is made to the user's program. If the given destination address is found to be invalid (the task table does not contain the address) the IPCF assumes that the message is intended for the PDP-10 and sends it there via the ICCF. It will be shown later that if the destination address is still invalid on the PDP-10 the message is then flushed. Thus, this routing algorithm allows user tasks to be unaware of the physical computer in which the destination task resides.

The [SYSTEM] INFO facility provides three basic services to all users. First, user tasks declare intentions of using the IPCF by performing a LOGON request to [SYSTEM] INFO. This request simply enters the task's name into the task table thereby making it a valid address. Second, tasks can perform the ACTIVE request which determines the validity of a potential destination address. Third, user tasks declare their end of use of the IPCF by performing the LOGOFF request. This request removes the task name from the task table and flushes any of its outstanding messages from the data base.

4.1.2.3 Query Routine

This routine allows a task to query for information concerning the present message in its queue. It returns such information as the name of the sending task and the length of the message portion of the packet.

4.1.2.4 Receive Routine

This routine allows a task to receive a pending IPCF packet. The current message in the task's queue is copied into a portion of the task's memory space specified in the request. Once the message is retrieved from the queue, the message buffer is removed from the buffer ring, and buffer pointers are updated, and the storage allocation routines return the buffer area back to the unused data base pool.

4.2 ICCF

The ICCF serves as the facility for sending and receiving messages from each computer over the high speed channel interface. Its responsibilities include reformatting messages either before they are sent or after they are

received and to relay the received message to that computer's IPCF. Each ICCF can be viewed as being similar to any other user task except that a high degree of specialized interaction occurs between the ICCF and IPCF.

4.2.1 Message Reformatting

The channel interface hardware can operate in several different modes of data packing and unpacking while making transfers between the two machines. However, the ICCF utilizes only the mode which packs/unpacks PDP-11 words, right justified, into/out of successive 18 bit PDP-10 bytes as shown in Figure 4-2. The actual message reformatting that occurs in a PDP-11 output buffer and PDP-10 input buffer is shown in Figure 4-3. The reformatting that occurs in a PDP-10 output buffer and PDP-11 input buffer is essentially identical but is applied in reverse order.

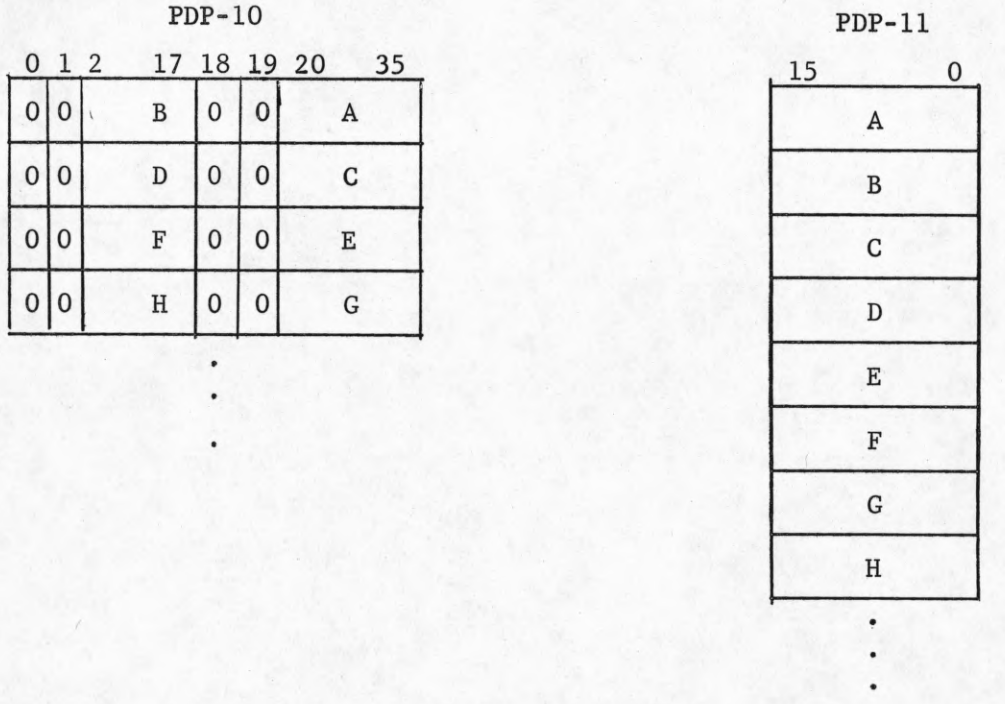

Figure 4-2. Data Packing Between the PDP-10 and PDP-11

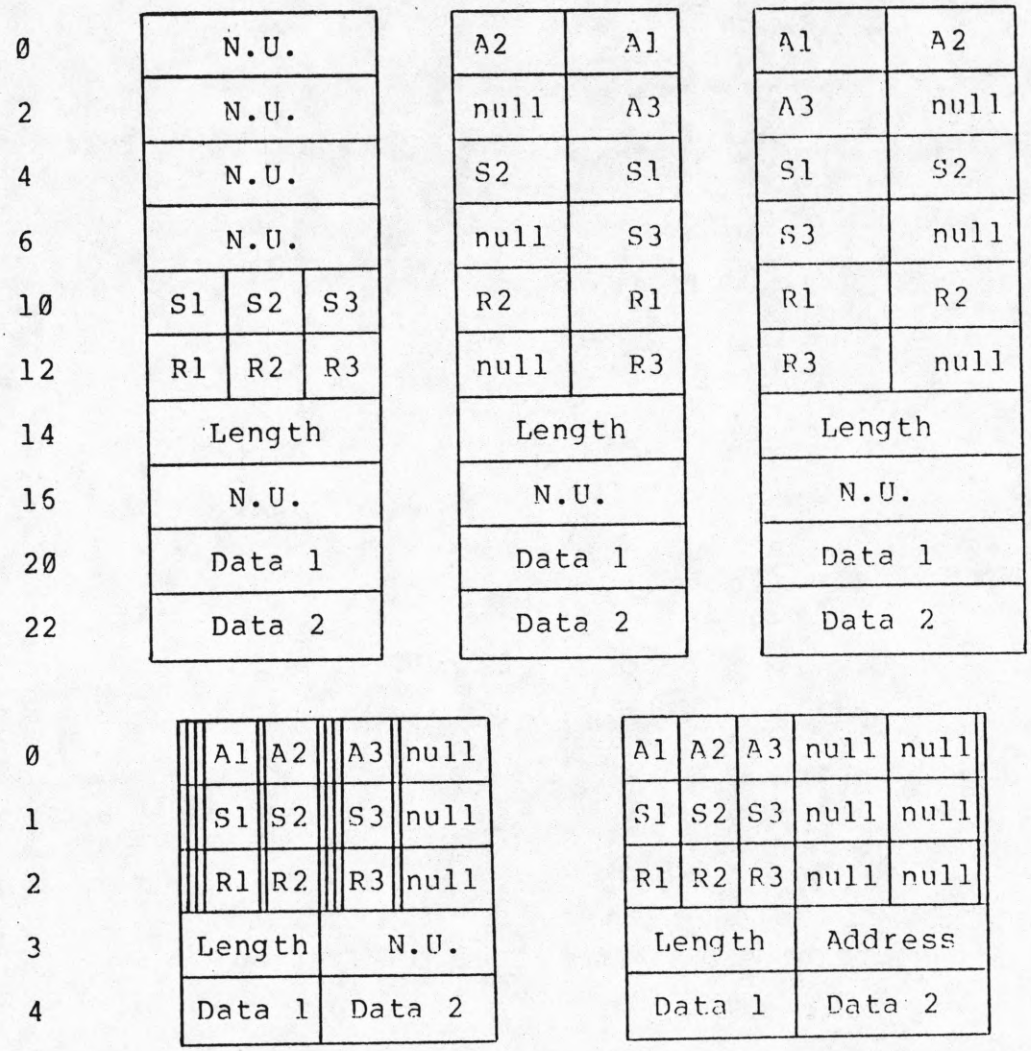

Figure 4-3. Message Reformatting.

A Radix-50 to ASCIZ conversion of the sender's and receiver's task names is required to conform with the PDP-10's ASCIZ orientation. A convenient PDP-11 monitor call allows rapid conversion and will fill successive bytes, as shown in Figure 4-3, with the ASCIZ equivalents. Since word 22 relative to the start of the buffer can contain a Radix-50 name required for certain [SYSTEM] INFO requests, it is copied into word 6 and also converted into an ASCIZ string. In the event that this word does not contain a name its conversion will be meaningless and will be later ignored.

Just prior to the channel transfer, the PDP-11 bytes containing the ASCIZ equivalents of the various task name characters must be swapped such that the characters will again be in the same lexical order once the PDP-11 words are mapped into the PDP-10 input buffer. Once the transfer is made the PDP-10 side uses byte manipulation instructions to fully left justify the ASCIZ fields within the buffer word and thus make the name compatible with the PDP-10 ASCIZ convention. The PDP-10 must also update the address word of the received message packet so it points to the memory location following itself. Once this is done, the received message packet is virtually in the proper PDP-10 IPCF format.

4.2.2 PDP-10 ICCF

The PDP-10 ICCF is composed of two parts, a channel receiving routine and channel driving routine. The receiving routine basically accepts messages sent by the PDP-11, it postformats them according to Figure 4-3, and then either relays the message to the appropriate destination task via the PDP-10 IPCF or it performs some internal bookkeeping. The channel

driving routine basically receives message packets from the IPCF and again either performs some internal ICCF bookkeeping or preformats the message and transfers it to the PDP-11's ICCF.

The above bookkeeping refers to the manipulation of an internal task table the ICCF must maintain. Since the sender/receiver addresses of an IPCF packet on the PDP-11 are names while those on the PDP-10 are PIDs, some sort of mechanism must exist that allows a PDP-10 packet to be properly routed to the PDP-11 via the PDP-10's IPCF and vice versa. The basic mechanism exists as a feature of the PDP-10's IPCF. A user job has the capability of possessing several different names and can be referenced by each PID associated with each name. Thus, if during a LOGON request on the PDP-11, a message is sent to the PDP-10 containing the name of the PDP-11 task then the channel receive routine can request the PDP-10's [SYSTEM] INFO to assign a PID to this name and then enter both name and PID into the task table. However, this name/PID combination belongs now only to the ICCF itself and not to any PDP-11 task directly. But now, when some PDP-10 task requests [SYSTEM] INFO for the PID associated with the name of a PDP-11 task, it will be receiving a PID of the ICCF. When that PDP-10 task sends a message packet to this PID the ICCF will retrieve it and perform a search through the task table to find the name of the PDP-11 task associated with it and then relay the message to the PDP-11. Likewise, if during a LOGON procedure on the PDP-10 a message is sent to the ICCF by a user's job containing the task's name and PID then the ICCF can simply enter both into the task table. Now, when a PDP-11 task sends a packet to a job that is actually on the PDP-10, the ICCF will receive the message from across the channel and will

again perform a search through the task table replacing the sender/receiver names with the proper associated PIDs. The packet can then be relayed to the proper PDP-10 destination task.

The channel receiving and driving routines are part of one job. In order to make the ICCF immune to any errors due to the time dependent behavior of the communicating tasks, the Software Interrupt System of the PDP-10 was utilized [6]. The ICCF consists of the channel receiving routine as the main background task while the channel driving routine exists as the interrupt task. The software interrupt system can be initialized by a user to generate interrupts on various I/O and non-I/O conditions. One of these conditions is the receiving of an IPCF message packet. By associating the channel driving routine solely with the receiving of IPCF packets and processing them while the channel receiving routine is associated solely with the receiving of channel packets and redispatching them, then both routines maintain only a unidirectional link to the IPCF and thus any possible cross interference between the two tasks is eliminated.

There is an implicit priority built into this structure with regard to the operation of each part of the ICCF. Regardless of the state that the background task is in it will be interrupted whenever an IPCF packet is received and execution will not resume until the interrupt task finishes. Thus in a highly interactive environment, the channel driving routine can "hog" the use of the channel for brief periods of time. For

---

[6] See Chapter 3 in reference (2)

the working environment for which this system is designed this inherent priority is an advantage since the PDP-10 will often execute command or message generating tasks while the PDP-11 will normally execute slave or message receiving tasks.

4.2.3 PDP-11 ICCF

The PDP-11 ICCF is far simpler in structure than that of the PDP-10. Whereas both the channel driving and receiving routines resided together in one job on the PDP-10, the two routines are separated on the PDP-11. The channel driving routine is logically located within the IPCF module itself since all outgoing messages across the channel are initiated only by the IPCF rather than some user task. The channel receiving routine, however, is located in a separate job whose basic purpose is to monitor the channel, reformat, and relay all incoming messages to the IPCF. The two routines also maintain an unidirectional link to the IPCF so cross interference is also eliminated.

The one feature that the channel receiving routine has that has no equal on the PDP-10 is the ability to command the operating system to load a program into core and start multitasking. This feature is utilized whenever the routine attempts to relay a message to an invalid address. Through an error condition received due to the attempt to send the message, the routine interprets the receiver taks name as also being part of the file name containing the program of the destination task. Through several monitor calls the channel receive routine instructs the monitor to load the program and start its execution. This feature is particularly useful for bringing up an entire user multitask communication package from the PDP-10 side alone.

This feature is also necessary due to the structure of the PDP-11 multitasking operating system. From the command console a user can load and execute only one job. If other tasks are also meant to be executed for multitasking, then it is the responsibility of that first root task to issue the proper monitor calls to load and execute these other tasks. Since the MCCP will serve as part of the basic operating system then this feature is a necessity once the system itself is brought up.

4.3 PDP-11 User Initialization Routine

Due to the PDP-11/40 memory management hardware user's programs, IPCF, and ICCF cannot automatically access each other once they are loaded into core and execution is initiated. Rather, each job that intends to use the IPCF in some capacity (the ICCF even has to do this) must execute a standard subroutine that can be linked with his program. The subroutine essentially executes monitor calls that manipulate the user's page maps such that both the user task and IPCF are mapped to each other.

5. PDP-11 IPCF USERS MANUAL

The communications handling software being developed for the PDP-10 and PDP-11 machines is written such that it mimics both the protocol as well as a subset of features already existing in the Interprocess Communications Facility (IPCF) residing in the PDP-10 monitor [1]. As a result, intertack communication from PDP-11 to PDP-11, PDP-11 to PDP-10, or PDP-10 to PDP-11 will just as easily be accomplished from a user's level as it is in the IPCF on the PDP-10 alone. Since the PDP-10 IPCF already exists the following discussion focuses primarily on the PDP-11 IPCF. It will be shown that the IPCF mimics the behavior of a post office in the manner in which it handles messages. Topics in the following discussion are explained only to the extent necessary to be able to use the PDP-11 IPCF or to indicate differences with the corresponding PDP-10 IPCF.

5.1 IPCF Structure

The PDP-11 IPCF structure can be divided into two functional groups:

1. A data base to store messages
2. A collection of various routines to manipulate this data base

Each routine has a corresponding UUO associated with it. Basic intertask communication is accomplished by: 1.) copying user message packets into an appropriate buffer in the data base after the IPCFS. UUO is executed

---

[1] See Chapter 7 of reference (2)

and 2.) returning an appropriate message for the data base and copying it into the receiver's task memory space once the IPCFR. UUO is executed.

5.2 Message Packets

All message packets handled by the PDP-11 IPCF are analogous to letters sent to and received from a post office. Each packet consists of an "envelope" portion and a "message" portion. The format of both parts is shown in Figure 5-1. The message portion of the packet contains user defined information (or information needed by [SYSTEM] INFO) and can consist of up to 27 words

```
Word 0- ENVEL:   Flags           ; See Table 5-3
     1-          Sender's PID    ; Can be 0
     2-          Receiver's PID  ; Required
     3-          Message Length  ; Length of "MESS" portion
     4-          MESS            ; Address of message

Word 0- MESS:    info            ; Beginning of message
     1-          info
     2-          info
     .            .
     .            .
     .            .
```

Figure 5-1. Message Packet Format

5.3 Process ID (PID)

A PID is a user-declared symbolic name. The symbolic name must be at most a three character RADIX-50 name.

5.4 Queues

The IPCF will create a message queue for each task that declares a PID to the [SYSTEM] INFO facility. Any packet received from the IPCF will be moved to the receiver's defined addressing space in the same format as in Figure 5-1.

5.5 [SYSTEM] INFO

The [SYSTEM] INFO facility acts as a central information utility for the IPCF and performs several functions connected with task names. The various functions are listed in Table 5-1. The PID of [SYSTEM] INFO will always be 0.

5.5.1 [SYSTEM] INFO Request Format

The message portion of any packet sent to [SYSTEM] INFO must conform to the format shown in Figure 5-2.

Table 5-1
[SYSTEM] INFO Functions

Function	Name	Description	Argument
0	.IPCII	Enter name into task table	Name
2	.IPCIA	See if name exists in task table	Name
4	.IPCID	Remove name from task table	Name

```
Word 0- MESS:  Function       ; See Table 5-1
     1-        Function arg.  ; See Table 5-1
```

Figure 5-2. [SYSTEM] INFO Request Format

5.5.2 [SYSTEM] INFO Reply Format

[SYSTEM] INFO never returns any explicit information. The success or failure of a [SYSTEM] INFO request is returned implicitly in the type of return made from the IPCFS. UUO.

5.6 Error Codes

Any errors detected within the IPCF or as a result of a user's improper use of the IPCF will cause an error code to be returned in the IP.CFE within the user's packet descriptor block. The error will be one of those listed in table 5-2. Any error will cause an error return to be taken at the conclusion of the UUO execution.

Table 5-2
Error Codes (Returned in IP.CFE)

Value	Mnemonic	Reason
3	IP.CNP	No packet in receive queue
5	IP.CTL	Data too long for buffer
6	IP.CDU	Destination unknown
11	IP.CRR	No room in receiver's quota
14	IP.CIS	Invalid sender PID
16	IP.CUF	Unknown function
20	IP.CPF	Task table full
75	IP.CDP	Duplicate name
76	IP.CNN	Unknown name

5.7 IPCF Initialization Routine

The IPCFI. UUO is used to completely reinitialize the entire IPCF facility. It causes the task table to be cleared as well as the entire data base. This UUO is intended to be used by only a system task of a single user root task. Due to its extreme nature, it is not recommended for general use. Normal user tasks do not have to use it.

5.8 Packet Sending Convention

The IPCFS. UUO is used to send an IPCF packet. If there is room in the receiver's queue, for a packet being sent, it is put into the queue. If there is no room, the UUO will take the error return.

The form of an IPCFS. call is:

```
MOV     #ENVEL,RO       ; Must use RO as
IPCFS.                  ; linking register
error return
normal return
```

5.9 Packet Query Convention

The IPCFQ. UUO is used to "query" the status of the input queue and return information in the packet descriptor block about the next packet in queue. The returned information consists of the name of the sender task and the length of the message portion of the packet.

The format of the call is:

```
MOV     #ENVEL,RO       ; Must use RO as
IPCFQ.                  ; linking register
error return
normal return
```

5.10 Packet Receiving Convention

The IPCFR. UUO is used to receive an IPCF packet. The packet descriptor block should specify the length and starting address of a block of memory in the user's addressing space. If the length of the message portion of the packet in the queue is greater than that specified in the packet descriptor block then as much data as possible will be transferred to the user's memory space and the remainder will be lost.

The format of the call is:

```
MOV     #ENVEL,R0       ; Must use R0 as
IPCFR.                  ; linking register
error return
normal return
```

5.11 Using the IPCF

The IPCF is analogous to a post office in the manner in which it handles messages from one task to another. Thus, tasks can be classified according to the way in which they use messages:

1. Sources of messages
2. Sinks of messages
3. Source and sink of messages

If a task only generates and sends messages then it is not required to have a PID. This is analogous to the situation of sending a letter without a return address. However, a task that expects to receive a message of any kind is required to have a PID. It is encouraged, however, to always use a PID.

Thus the user must adhere to the following general guidelines if he is planning to use the IPCF:

1. Allow the size of the user task stack to be at least 10 words long
2. Reserve memory for sending and/or receiving messages
3. Declare intentions of using IPCF (by requesting a PID through [SYSTEM] INFO)---optional
4. Use IPCF
5. Declare end of use of IPCF (by notifying [SYSTEM] INFO)---optional, need only be done if step 2. was followed

It should be noted that the IPCFR. and IPCFQ. UUOs never block. Therefore if a user needs a message at a particular time in his program, he must execute a busy wait loop until that message is received.

The various flags in the high byte of the first word in the envelope portion of the message packet (as shown in Table 5-3) provides the user with additional flexibility in using the IPCF. When the user sets the IP.CFS and/or IP.CFR flags the IPCF interprets the second and third words of the envelope as a pointer to the actual sender and/or receiver addresses.

Table 5-3
Packet Descriptor Block Flags

Bit	Name	Meaning
0-7	IP.CFE	Error code field
8-10	----	Not used
11	IP.CFP	Request is privileged
12	IP.CFO	Send above quota
13	IP.CFR	Indirect Receiver's PID
14	IP.CFS	Indirect Sender's PID
15	----	Not used

Currently, each task's queue can hold a quota of five messages at one time. More messages can be placed in this queue if the sender task sets either the IP.CFP of IP.CFO flags. Excessive use of these flags is highly discouraged since if the destination task does not empty its queue fast enough or if there are bugs in the receive program, the entire data base can fill up and a system crash is a real possibility.

Any of the IPCF UUOs may alter a user's envelope portion at times after the UUO call is initiated. Thus the user must not assume no changes have occurred once the return from the UUO is made.

The IPCF does not associate each declared task name with any sort of job number. Thus a user task can impersonate any other task simply

by using any valid task name. This feature certainly opens the door to malicious use of the IPCF. However, if used in a clever way, very complex intertask control communication can be achieved.

The MCCP requires all PDP-10 tasks, that intend to send or receive messages from the PDP-11, to send a "LOGON" message to the PDP-10 ICCF. The format of this message as well as the format of other important types of messages is summarized in Figure 5-3.

	PDP-11 Task	PDP-10 Task
Logon Procedure for a PDP-11 task	ENV: ? ? 0 2 MES MES: .IPCII Name	ENV: ? Your PID PID of ICCF 2,,MES MES: 1,,.IPCII Name
Logon Procedure for a PDP-10 Task		ENV: ? Your PID PID of ICCF 2,,MES MES: 0,,.IPCII Your name
Logoff Procedure (PDP-10 or PDP-11)	ENV: ? ? 0 2 MES MES: .IPCID Name	ENV: ? Your PID PID of ICCF 2,,MES MES: 0,,.IPCID Name

? = The settings of these words depends on the user.

Figure 5-3. MCCP Special Message Packet Formats.

6. CONCLUSION

The multicomputer computer communication facility has been designed to provide high level use as well as to interface well in a user environment where it is necessary to control several interdependent real time control tasks. The typical user only has to set up messages according to the particular IPCF protocol his task will be using. The user's task does not have to be aware of the actual computer the receiving task is on, the MCCP will perform the appropriate reformatting of the message and transfer the message across the channel interface automatically when the need arises.

The MCCP was written using both the PDP-10 and PDP-11 assembler code. This provides a means for obtaining the fastest possible execution speed of the MCCP and therefore communication time among real time tasks was minimized.

The structure of the program code itself was made very modular to insure easy comprehensibility, debugging, and future expandability if the need should occur.

The major limitation of the system lies in the maximum message length each computer can handle. Since DEC set the maximum message packet length on the PDP-10 to be 16 words, the PDP-11 IPCF was designed to handle a maximum length of 32 words. Thus when a maximum packet is sent to either computer, full compatibility is achieved through the data packing and reformatting that occurs between the two machines via the channel hardware.

Possible improvements include providing a specialized means to transfer large blocks of data (**>>** 32 PDP-11 words) and a mechanism that

causes a PDP-11 task to be blocked until a message packet is received through the IPCFR. UUO. The first improvement would allow whole program files or large data files to be transferred quickly, and the second improvement would free each user from having to write his own busy wait routines and would also cause less CPU time to be consumed.

REFERENCES

1. Digital Equipment Corp., DECsystem-10 Assembly Language Handbook, third edition, DEC-10-NRZC-D, Digital Equipment Corp., Maynard, Mass., 1973.

2. Digital Equipment Corp., DECsystem-10 Monitor Calls Manual, DEC-10-OMCMA-A-D, Digital Equipment Corp., Maynard, Mass., 1974.

3. Digital Equipment Corp., Macro-11 Assembler Programmer's Manual, DEC-11-OMACA-A-D, Digital Equipment Corp., Maynard, Mass., June 1972.

4. Digital Equipment Corp., PDP-11/40 Processor Handbook, Digital Equipment Corp., Maynard, Mass., 1973.

5. Dijkstra, E. W., "Cooperating Sequential Processes," in Programming Languages, (Genuys ed.), Academic Press, 1968.

6. Hansen, Per Binch, Operating System Principles, Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

7. Jacobus, C. J., M and M System Design and Operation, M. S. Thesis, Coordinated Science Laboratory, University of Illinois, Urbana, Ill., 1975.

8. Jacobus, C. J., M and M EMT Calling Conventions, Coordinated Science Laboratory, University of Illinois, Urbana, Ill., April 1975.

9. Knuth, The Art of Computer Programming, vol. 1/Fundamental Algorithms, Addison-Wesley, Reading, Mass., 1968.

10. Selander, John M., A PDP-10 to PDP-11 Asynchronous Communications Interface, M. S. Thesis, Coordinated Science Laboratory, University of Illinois, Urbana, Ill., 1976.