PSEUDO-DEVICES

by

Paul Primmer - Hewlett-Packard

ABSTRACT

This paper will explain what a pseudo-device is and how it is used in Hewlett-Packard data communication products. Since there are several layers of software on either side of a pseudo-device, it will be necessary to also explain briefly CS, DITs, Monitors, and Attachio. There are differences in the various data communication subsytems and their use of pseudo-devices so this paper will use DS/3000 for all examples.

PSEUDO–DEVICES

SLIDE 1

SLIDE 1

The objective of this presentation is to give an understanding of pseudo-devices. Pseudo-devices can not be explained in a standalone fashion. That is, the software they link to must also be explained as well as some general I/O concepts. DS/3000 was chosen as an example product to help explain pseudo-devices but it is important to note that there are differences between the different data comm. products.

PSEUDO-DEVICE

AN EFFICIENT MEANS FOR MULTIPLE PROCESSES TO SHARE A NON-SHARABLE DEVICE.

SLIDE 2

Pseudo-devices are an efficient means for multiple processes to share a non-sharable device. To properly explain this definition of pseudo-devices requires an understanding of some lower level software and hardware that make up HP's data comm. products.

NON-SHARABLE DEVICES

Intelligent Network Processor - INP

Synchronous Single Line Controller - SSLC

Hardwired Serial Interface - HSI

These three communication devices are accessed through calls to the Communication System Intrinsics (CS). CS does an exclusive open on the above devices similar to an exclusive file open.

SLIDE 3

Currently there are 3 hardware devices available for synchronous data communication: Intelligent Network Processor - INP, Synchronous Single Line Controller - SSLC, and the Hardwired Serial Interface - HSI. All three are accessed by a set of system intrinsics called CS which stands for Communication System. Like the file system, which provides a common set of high level intrinsics for access to dis-similar hardware (i.e tape, disc, and line printers), CS provides a common set of calls for accessing these 3 synchronous devices. In order to use one of these devices a COPEN procedure is called which does an exclusive open on the device making it non-sharable.

CS

The Communication System intrinsics are used by all current HP data comm. products to provide the Binary Synchronous protocol.

BISYNC:

```
                  ENQ--->
                          <---ACK0
  BCC ETX TEXT STX--->
                          <---ACK1
  BCC ETX TEXT STX--->
                          <---NAK
  BCC ETX TEXT STX--->
                          <---ACK0
                  EOT--->
```

SLIDE 4

All current HP data comm. products are based on IBM's Binary Synchronous Communication protocol which was first introduced in 1966 and has since become the industry de facto standard for medium and high speed data communication. What BISYNC offers is an effective protocol for sending blocks of text and a means of recovering from line errors. This handshaking between stations is accomplished by using special control characters and a predefined line protocol.

DS CONVERSATIONAL BISYNC

```
                ENQ--->
                        <---ACK0
BCC ETX TEXT STX--->
                        <---STX TEXT ETX BCC
BCC ETX TEXT STX--->
                        <---NAK
BCC ETX TEXT STX--->
                        <---STX TEXT ETX BCC
                EOT--->
```

TEXT FOR DS:

HEADER (16 BYTES)
+
APPENDAGE (VARIABLE)
+
DATA (USER DEFINED)

TEXT

SLIDE 5

DS/3000 uses a modified form of BISYNC called conversational. This conversational form improves efficiency by answering a correctly sent block of text with text. BISYNC can be thought of as the transport mechanism for this text block. For DS/3000 this text block always contains a 16 byte header which contains information such as: the from PIN, the to PIN, what type of message, how long a message, and is the data compressed. In addition to the header, which is always present, there is an optional appendage section which can be thought of as a header extension. Finally, there is the users data. These 3 parts are combined by the upper level DS intrinsics and passed to the lower level CS for transport over the data link.

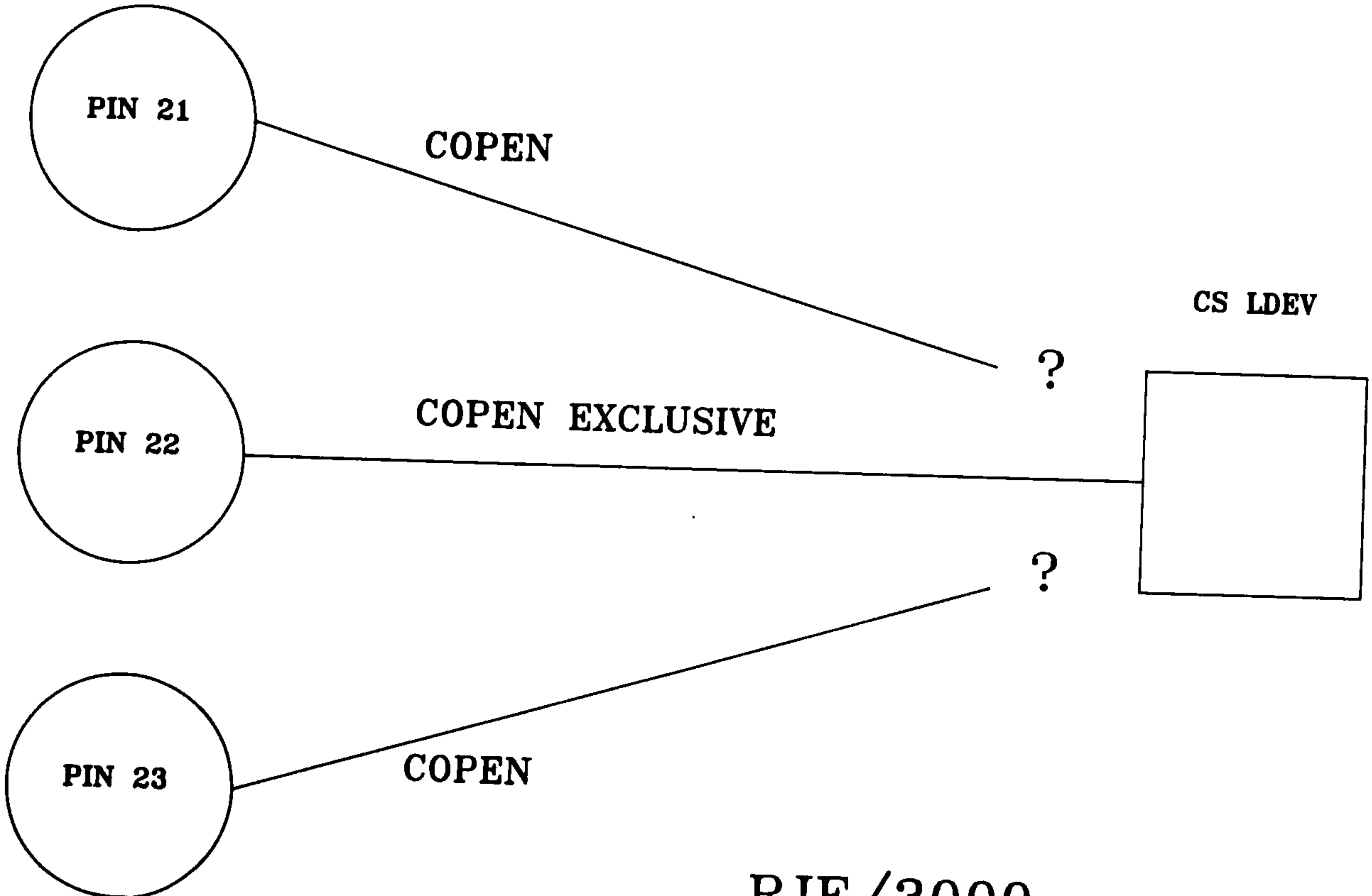
CS INTRINSICS

COPEN	EXCLUSIVE OPEN OF LINE
CCLOSE	CLOSES LINE
CWRITE	WRITES DATA TO LINE
CREAD	READS DATA FROM LINE
CCONTROL	ALLOWS VARIOUS CONTROLS OF THE LINE TO BE PERFORMED

SLIDE 6

Rather then writing a special communication protocol for each data comm. product, a common set of intrinsics were developed. This is analogous to the common set of intrinsics for the file system. Of particular note is that the COPEN intrinsic does an exclusive open. Exclusive access is required due to the nature of the communication link which requires specific responses at specific times. If this device were shared among several processes, the result would be total confusion.

USER PROCESSES

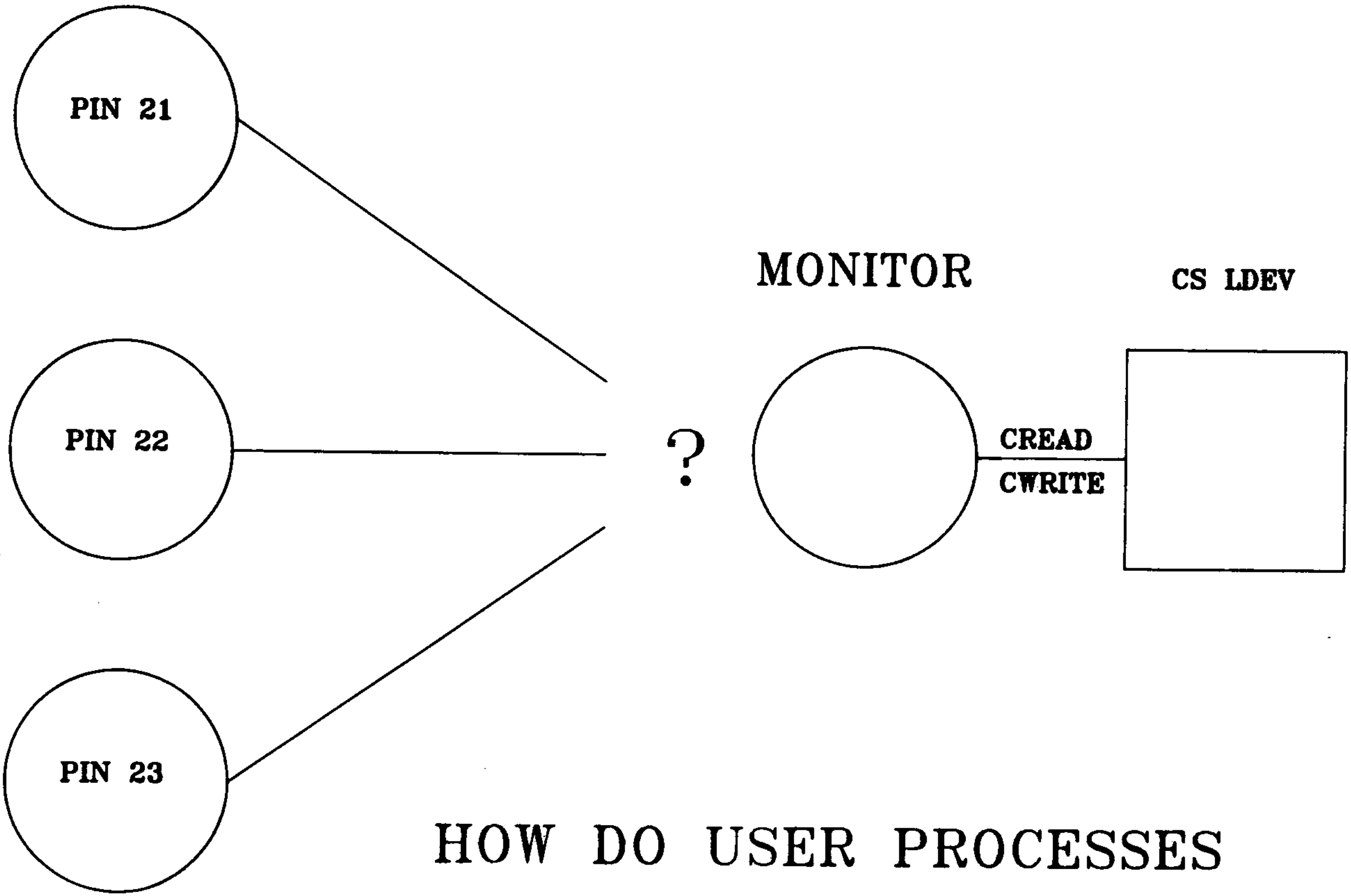


RJE/3000

SLIDE 7

This exclusive open is evident in the product RJE/3000 which is nothing more than a large program issuing CS intrinsic calls. But this scheme would be unacceptable for DS/3000 which allows multiple users at both ends to be using the same communication link.

USER PROCESSES



HOW DO USER PROCESSES TALK TO MONITOR?????

SLIDE 8

A way to overcome this problem is to allow one process to exclusively open the communication link and do all the reading and writing to the line. This process is called the monitor. More specifically in DS/3000 it is called DSMON and is created when the operator types "DSCONTROL ldev;OPEN". Now there is an exclusive owner of the communication device which the other processes can direct their requests. But there are two problems with processes talking to this monitor process.

INTER-PROCESS COMMUNICATION

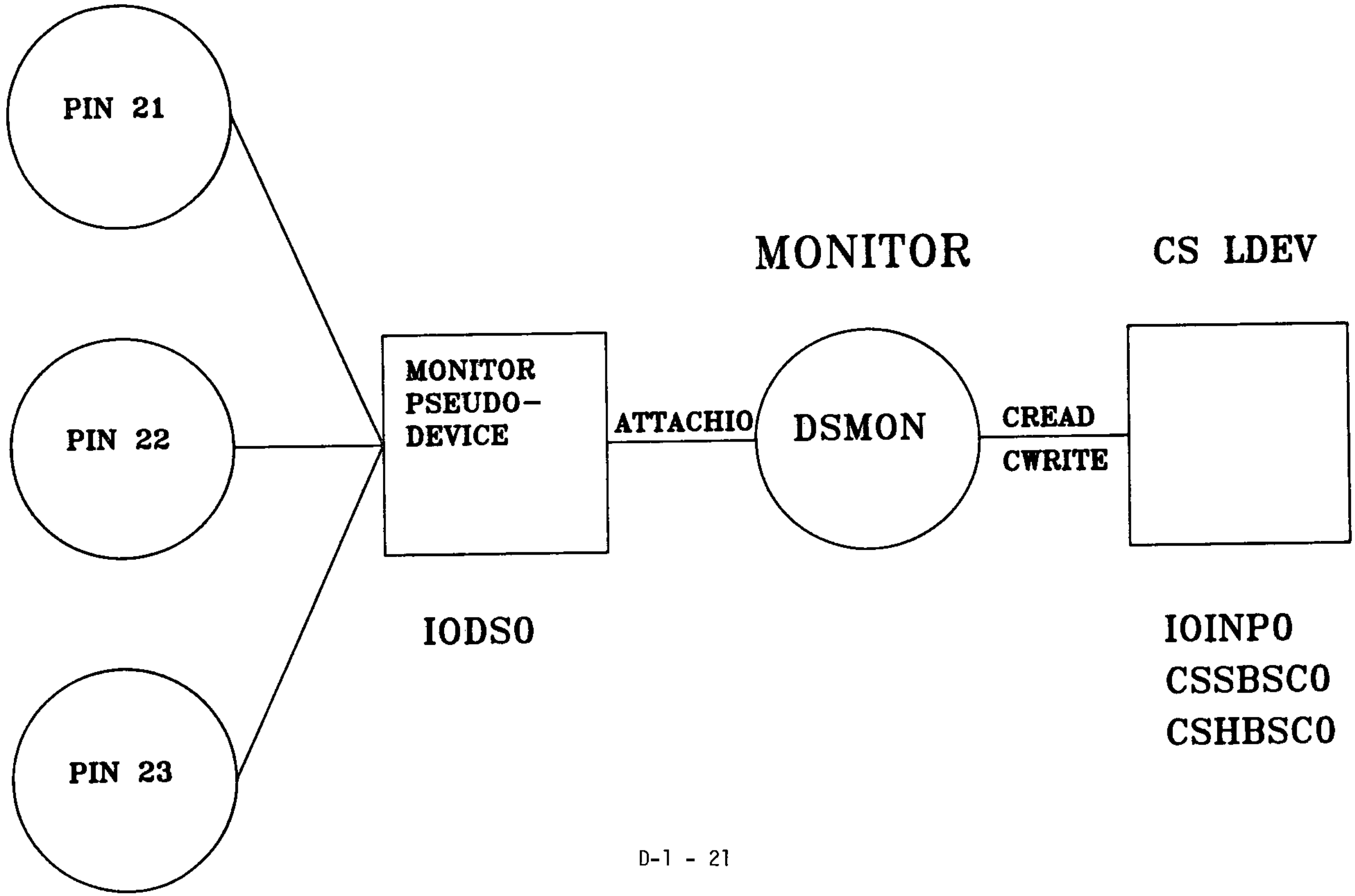
PROBLEM #1: INTER-PROCESS COMMUNICATION ONLY BETWEEN FATHER AND SON PROCESSES.

PROBLEM #2: MONITOR NEEDS TO BE AWAKENED WHEN EITHER CS LINE I/O COMPLETES OR WHEN A USER PROCESS HAS SOMETHING FOR THE MONITOR TO DO.
UNDER MPE III, WE CAN WAIT FOR EITHER I/O COMPLETION OR FOR PROCESS ACTIVATION BUT NOT BOTH AT THE SAME TIME!

SLIDE 9

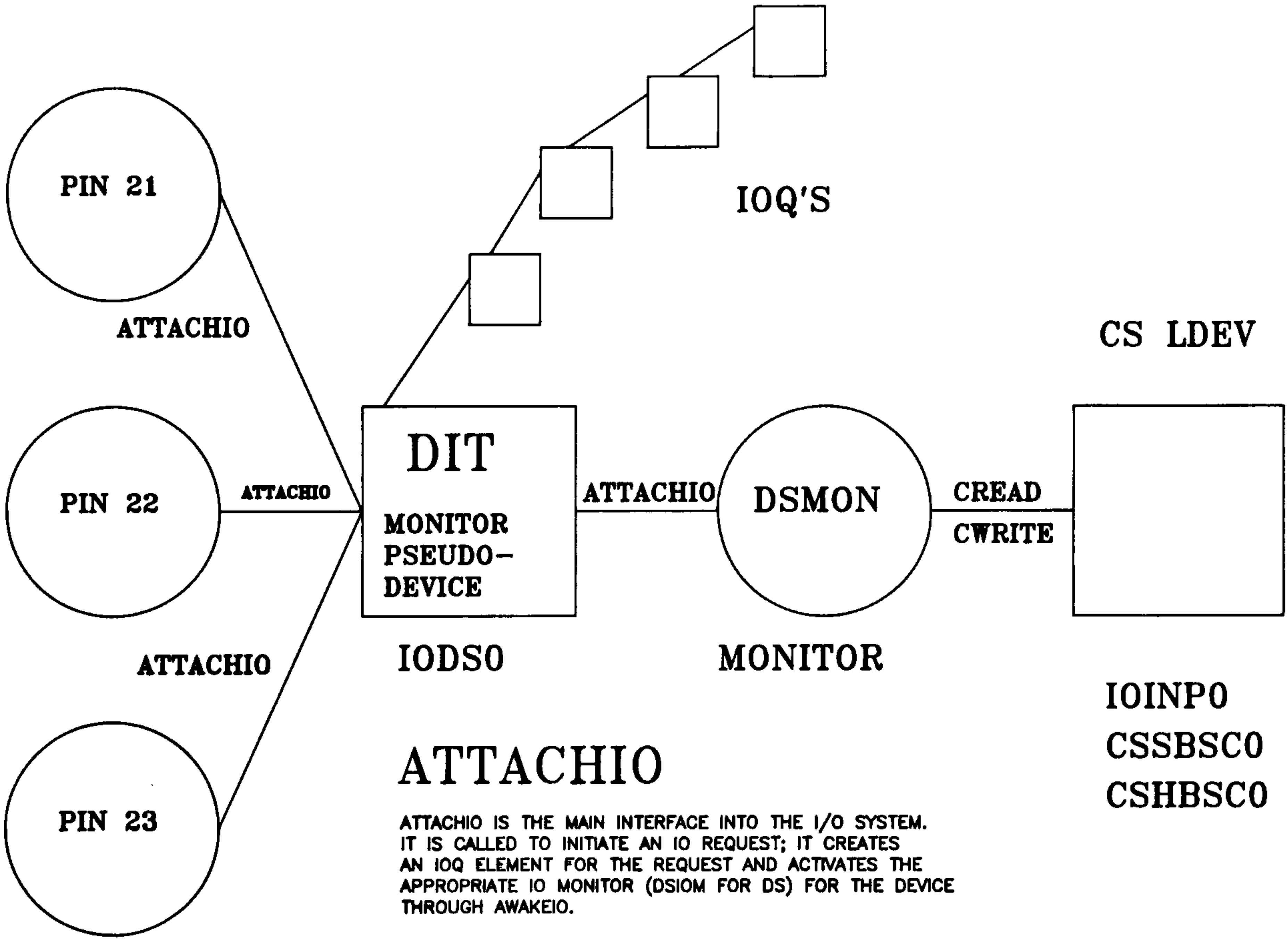
In MPE-III only processes in the same family (father/son) can communicate. Secondly, the monitor process is controlling the communication line and therefore is in an I/O wait. In MPE-III, a process can wait for I/O completion or for process activation but not both at the same time. How then will processes get the monitors attention?

USER PROCESSES



SLIDE 10

By creating a pseudo-device the monitor can issue no-wait I/O requests to both the CS device and the pseudo-device, and service the first request to complete. Now user process get the attention of the monitor by writing their request to the pseudo-device via calls to ATTACHIO. The data in the write request becomes the data for the monitor's read request. The monitor is awakened by completion of I/O and discovers that it is from the pseudo-device so it cancels its' current CS request and performs the desired I/O. In a normal idle state the monitor is asleep with a CREAD to the CS device and an ATTACHIO read to the pseudo device. When either request completes the monitor awakens for service. But who is ATTACHIO?



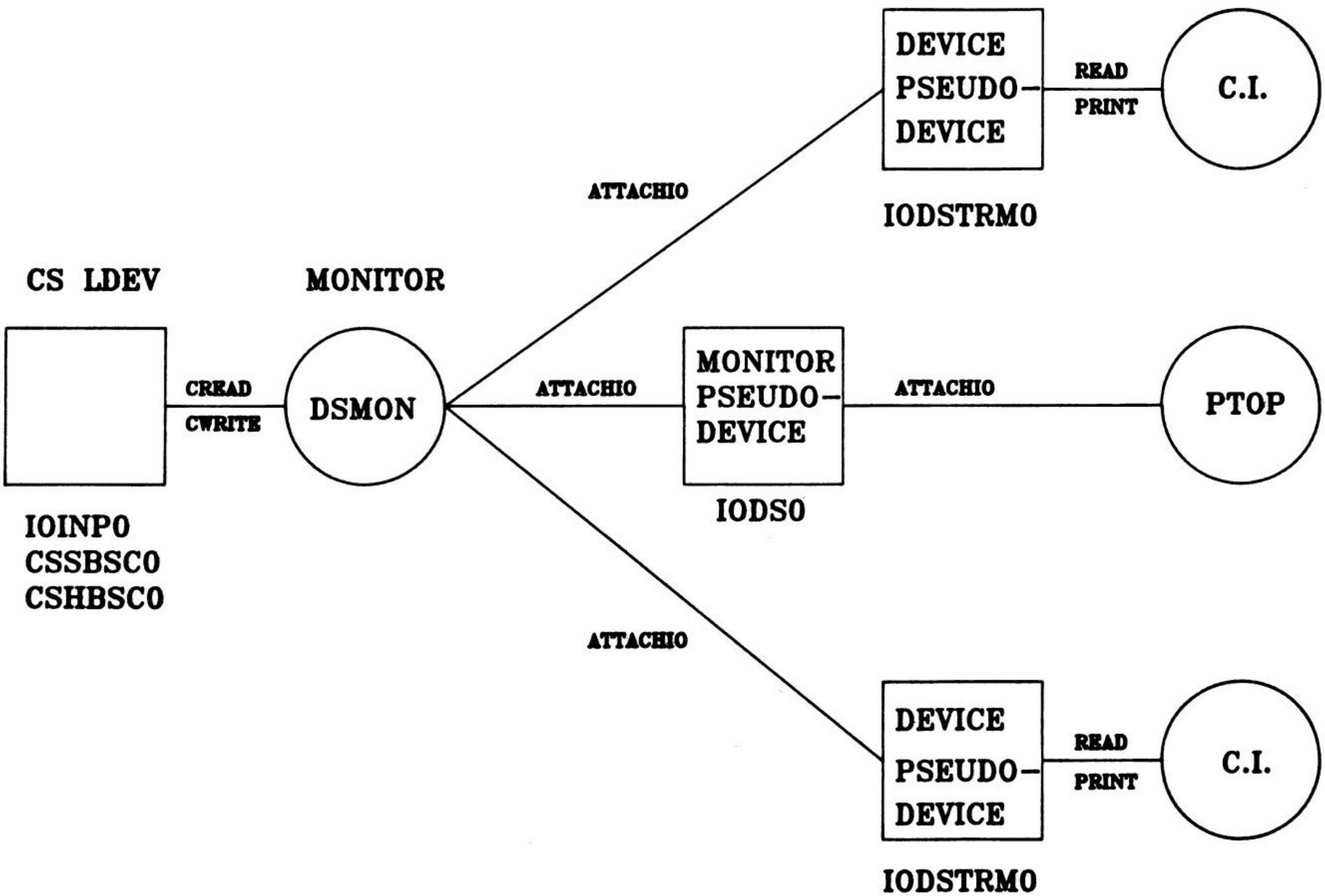
ATTACHIO

ATTACHIO IS THE MAIN INTERFACE INTO THE I/O SYSTEM. IT IS CALLED TO INITIATE AN IO REQUEST; IT CREATES AN IOQ ELEMENT FOR THE REQUEST AND ACTIVATES THE APPROPRIATE IO MONITOR (DSIOM FOR DS) FOR THE DEVICE THROUGH AWAKEIO.

SLIDE 11

ATTACHIO is the main interface into the I/O system. For example, when a user issues a file system intrinsic, that intrinsic will eventually call ATTACHIO. ATTACHIO will create an eleven word IOQ element which describes the nature of the request (i.e read, write, or control), the location of the buffer, the PIN that made the request, and any miscellaneous parameters that are device dependent. This IOQ element is in a common table for all devices, but all requests for the same device are linked together by one of the eleven words. Every device configured in the system has a Device Information Table (DIT) to keep track of what the device is currently doing and a pointer back to the IOQ table for all the requests pending. After ATTACHIO creates the IOQ element, it requests and activates the appropriate I/O monitor. In the case of DS/3000 the I/O monitor is named DSIOM. A real device monitor at this point would call the appropriate drivers to do the real physical I/O. DSIOM is the 'nerve center' for DS I/O and takes care of routing the request to either IODSO or IODSTRMO. These drivers move the data from the users buffer (pointed to by the IOQ) into DSMON's stack (pointed to by the IOQ request it made earlier). Both IOQs are flagged completed; DSMON is awakened and performs the I/O to the CS device.

SLAVE SIDE



SLIDE 12

SLIDE 12

At the slave side the monitor is in a I/O wait state when its' CREAD completes, and it reads the data into its' stack. The slave side needs some process to perform the DS request and rather than writing special code to do this, a C.I. is created at the slave end. But the C.I. is designed to talk to an interactive terminal not a monitor, so a pseudo-terminal device was created which looks and acts just like a real terminal to the C.I.. When the slave C.I. responds with an answer it calls the PRINT intrinsic which calls ATTACHIO which calls DSIOM which invokes IODSTRMO. IODSTRMO moves the buffer to DSMON's stack and DSMON CWRITES the request to the CS line.

SUMMARY

A pseudo-device becomes an efficient means for multiple processes to share a non-sharable device. By using a pseudo-device a monitor process can use no-wait I/O to act as a traffic cop to the CS device. Upper level software can use standard I/O routines (ATTACHIO) to make requests to the lower level I/O system. By isolating the different functions a layered software approach can be realized which allows multiple products to share common routines (CS).

??????????????? QUESTIONS ???????????????