OMNET - HIGH SPEED DATA COMMUNICATIONS FOR PDP-11 COMPUTERS

C.F. Parkman and J.G. Lee

GENEVA
1979
Propriété littéraire et scientifique réservée pour tous les pays du monde. Ce document ne peut être reproduit ou traduit en tout ou en partie sans l'autorisation écrite du Directeur général du CERN, titulaire du droit d'auteur. Dans les cas appropriés, et s'il s'agit d'utiliser le document à des fins non commerciales, cette autorisation sera volontiers accordée.

Le CERN ne revendique pas la propriété des inventions brevetables et dessins ou modèles susceptibles de dépôt qui pourraient être décrits dans le présent document; ceux-ci peuvent être librement utilisés par les instituts de recherche, les industriels et autres intéressés. Cependant, le CERN se réserve le droit de s'opposer à toute revendication qu'un usager pourrait faire de la propriété scientifique ou industrielle de toute invention et tout dessin ou modèle décrits dans le présent document.

© Copyright CERN, Genève, 1979

Literary and scientific copyrights reserved in all countries of the world. This report, or any part of it, may not be reprinted or translated without written permission of the copyright holder, the Director-General of CERN. However, permission will be freely granted for appropriate non-commercial use. If any patentable invention or registrable design is described in the report, CERN makes no claim to property rights in it but offers it for the free use of research institutions, manufacturers and others. CERN, however, may oppose any attempt by a user to claim any proprietary or patent rights in such inventions or designs as may be described in the present document.

CERN - Service d'Information scientifique - RD/422 - 2000 - janvier 1980
ABSTRACT

Omnnet is a high speed data communications network designed at CERN for PDP11 computers. It has grown from a link multiplexor system built for a CII 10070 computer into a full multi-point network, to which some fifty computers are now connected. It provides communications facilities for several large experimental installations as well as many smaller systems and has connections to all parts of the CERN site. The transmission protocol is discussed and brief details are given of the hardware and software used in its implementation. Also described is the gateway interface to the CERN packet switching network, "Cernet".
ACKNOWLEDGEMENTS

The people named below have played greater or lesser parts in the early design of the CII 10070 based, "star" data-link network and in its subsequent extension into an independent, multi-point network.

CII 10070 based data link system - software:
R. Cooper  H. Davies  A. Jeavons  P. Villemeoes  D. Wiegandt

CII 10070 based data link system - hardware:
J. Joosten  C. Parkman  J-M. Sainson  R. Zurbuchen

SMO operating system development:
A. Jeavons  M. Standley  T. Streater

Omnet software development:
G. Lee  J. Nerdal  C. Piney  M. Standley  T. Streater
A. Vascotto

Omnet hardware development:
C. Parkman

Omnet operations and maintenance:
J-L. Bretin  S. Fisher  A. Lovell  C. Parkman  J-C. Pinasseau
R. Pordes  M. Sheehan  CERN Computer Centre Operations Staff

Construction:
The CERN/DD Electronics Workshop has undertaken all aspects of the construction of the special hardware used in Omnet.
# TABLE OF CONTENTS

Abstract .................................................. iii  
Acknowledgements .......................................... iv  

## 1. INTRODUCTION ........................................ 1  
   History ...................................................... 1  
   Overview of the Current System ......................... 2  

## 2. TRANSMISSION PROTOCOL ............................... 4  
   Introduction ............................................... 4  
   Transfer Phase 1 (Initiation) ............................ 5  
   Transfer Phase 2 (Parameter Block) ....................... 6  
   Transfer Phase 3 (Data Transfer) ......................... 7  
   Status Byte Contents ..................................... 8  
   Software Status Codes .................................... 9  
   Initiate Codes ............................................. 9  
   Acknowledgement Codes After Parameter Block .......... 10  
   Acknowledgement Codes After User Data Transfer ...... 11  
   Transfer Termination Codes ................................ 11  
   Hardware Status Codes .................................... 12  
   Examples of Complete Transfers ......................... 13  
   Normal Data Transfer .................................... 13  
   Errored Data Transfer .................................... 14  
   Multiplexed Data Transfer ................................ 15  

## 3. HARDWARE ............................................... 17  
   Data Link Interface Hardware ............................ 17  
   General Characteristics ................................... 17  
   PDP11 Data Link Interface ................................. 18  
   General ..................................................... 18  
   Data Path ................................................... 19  
   Status Path ................................................ 21  
   Line Transmission Equipment ............................... 21  
   Co-axial Cable: 5 Megabaud ................................ 22  
   Twisted-quad Cable: up to 5 Megabaud .................... 22  
   Telephone lines: up to 36 kilobaud ...................... 23  
   Switching Nodes ........................................... 23  

- v -
4. SOFTWARE

SMO (Small Machine Organiser) .......................... 24
   Design Criteria ........................................ 24
   System Structure and Generation ...................... 24
   Operator Communication - "KEYTASK" .................. 25
   Multi-Tasking Features ................................. 25
   Input/Output System .................................... 26
   Re-entrant Tasks in SMO Systems ..................... 26
   Debug and Trace Facilities ............................. 27
   Library Utility Routines ............................... 27
   Omnet Software Under SMO .............................. 27
   Multiplexor Operation .................................. 28

The RSX-11 System ....................................... 29
   Omnet Implementation Under RSX-11 ................... 29

Bootstrapping and Down-line Loading of Tasks .......... 30
   Hardware and Software Bootstraps .................... 30
   Directly Connected Machine ............................ 31
   Machine Connected via a Multiplexor ................. 31
   Bootstrapping Procedure ............................... 32
   Dynamic Task Loading for SMO and RSX-11S Systems .. 32

5. OMNET/CERNET GATEWAY ............................... 33

   General .................................................. 33
   Hardware ................................................ 33
   Gateway Software ....................................... 34
      Functional Description ............................... 34
      Flow-Control Omnet to Cernet ....................... 35

Appendix

A. OMNET TOPOLOGY .................................... 37

   Configuration Diagram ................................ 38

B. PERFORMANCE MEASUREMENTS ......................... 39

   Omnet .................................................. 39
   Omnet/Cernet Gateway .................................. 40

C. RELIABILITY ......................................... 41

   Error Rate ............................................. 41
   Failure Rate .......................................... 41

D. TEST FACILITIES .................................... 42

   Basic low-level Tests - "Checklink" ................. 42
   High-level Test - "Testlink" .......................... 42
   Omnet/Cernet Gateway Tests ......................... 42
      Basic hardware tests ............................... 43
      High-level tests ................................... 43
E. OPERATIONS .............................................. 44

F. PDP11 DATA LINK INTERFACE REGISTER LAYOUT ............ 45
   Command Register ........................................ 45
   Error Register .......................................... 47
   Configuration Register .................................. 48
   Remote Status Register ................................. 48
   Local Status Register .................................. 48
   Memory Address Register ................................ 49
   Word Count Register .................................... 49

BIBLIOGRAPHY ................................................. 50

INDEX ......................................................... 52

Abstract .................................................... i

Acknowledgements .......................................... ii
1. INTRODUCTION

CERN (The European Organisation for Nuclear Research) is a modern high-energy physics laboratory, whose operation relies on the use of a large amount of distributed computing power. The control of experimental electronics and recording of data at physics facilities is managed by one or more minicomputers; there are over one hundred such machines in use at CERN. This distribution of computing power would have lead rapidly to the duplication of facilities, had not the development of data communications kept pace with, and in many cases anticipated, the need for the centralisation of many of the computing facilities.

1.1 HISTORY

In 1969 work started at CERN on the implementation of a computer system designed to support two large experimental facilities (the Omega\(^1\) and Split Field Magnet spectrometers). In the initial configuration, two EMR 6130 computers\(^2\) were to be connected to a CII 10070 computer\(^3\) by high speed, serial data links, to provide on-line analysis of samples of experimental data and a DEC\(^4\) PDP11/20 similarly connected to provide interactive graphics.

The computing requirements of the two original projects soon changed, and by 1971 three other PDP11/20 computers had been connected, also by high speed links, to the CII 10070.

The services developed on the CII 10070 for PDP11 support became increasingly popular and the demand for connections by user groups unconnected with the original customers grew to such a level that the original scheme of connecting each individual data link to a separate input/output channel controller became uneconomical, both in cost and in operational overheads. The solution adopted was to implement a front-end link multiplexer based on a minicomputer. The natural choice for this machine was the PDP11, since the necessary high-speed data link already existed.

Rather than adopt an ad hoc solution to the link multiplexing problem, the opportunity was taken to develop a more general message switching system. In the absence of a suitable, commercially

\(^1\)Hence "OMega NETwork"

\(^2\)The EMR 6130 is a 16-bit small computer. Electro-Mechanical Research was then a division of Schlumberger.

\(^3\)The CII 10070 is a French (Compagnie d'Informatique Internationale) built Xerox Data Systems Sigma 7.

\(^4\)DEC - Digital Equipment Corporation
available operating system for the PDP11, a highly flexible, multi-
tasking, operating system was written: "SMO" - the Small Machine
Organiser. Many of the initial ideas for SMO came from an earlier
system which had been used to implement the interactive graphics
system mentioned above.

At the same time (1975/1976), the PDP11 data link interface, which
formed the basis of the link hardware, was rebuilt to reduce its
physical size and to improve its performance. Backwards compatibility
was retained with the existing hardware at the serial line level.

In this form the Omnet of today was born. It then grew until at the
end of 1976, some twenty four computers had been connected either, as
originally by direct links, or by one of three multiplexor PDP11s.

During the course of 1978, it became apparent that the CII 10070
was coming to the end of its useful life as a physics support
computer. The CERN Computer Centre had received its first large IBM
370 system and work was in progress on the implementation of "Cernet",
a generalised packet switching network. Small computer support which
had hitherto been provided by the CII 10070 for PDP11s, was to be
transferred to the IBM machines.

In order that Omnet could survive the eventual loss of the CII
10070, it had in some way to be connected to the central IBM system.
It was decided to implement a gateway connection through Cernet,
rather than to attempt another direct connection to the large main-
frame computer. By late 1977 prototype gateway hardware and software
had been implemented and the CII 10070's facilities were gradually
transferred to the IBM. The first on-line sample analysis of
experimental data via Omnet and Cernet to the IBM was made in April
1978. Finally, at the end of that year, the CII 10070 was
disconnected from Omnet, which henceforward relied on a combination of
the IBM computers and a large PDP11 for its support and for the users'
facilities.

1.2 OVERVIEW OF THE CURRENT SYSTEM

Omnet provides inter-communication for local clusters of PDP11s,
communication between Omnet computers and, via Cernet, a means of
communicating with the Central CDC and IBM computers. It covers
virtually the entire CERN site, involving distances of over 4
kilometres. Facilities for task to task communication are provided
which are optimised for data links with speeds of greater than 1
Megabaud, although Omnet links have been used successfully at
considerably lower speeds (down to 4.8 kilobaud). With a 5 Megabaud
line speed, average task to task rates of some 200 kilobytes per
second have been achieved using 5 kilobyte data blocks.

The Omnet transmission protocol provides synchronisation between
the sending and receiving tasks and data flow control on a message
basis. The user's data is read directly into the receiving task's input buffer, thus avoiding costly memory to memory copying operations. Although data blocks of any length up to 32 kilobytes are, in theory, allowed, a practical restriction of 10 kilobytes has been imposed to allow message switching in a computer of reasonable memory size.

The Omnet software has been designed to be simple and small. The complete package in a PDP11 running the SM0 operating system, extends to only 1.5K words or 2K words if message switching is required. There is a high degree of optimisation of data transfers between tasks in adjacent machines.

Specially designed hardware is used to provide the physical links from the computer's input/output buses over a private network of high-quality data transmission cables. The hardware has been implemented in as simple and cheap a manner as possible in order to reduce the cost of a connection. A full PDP11 connection, including two computer interfaces and the line transmission equipment ("modem"), but excluding the cable, costs in the order of 7000 Swiss Francs.
2. TRANSMISSION PROTOCOL

2.1 INTRODUCTION

The transmission protocol is designed around the special hardware which is able to mix true data and control ("status") information on the same transmission lines. Communication on the full-duplex (4-wire) transmission lines is full-duplex for control information and half-duplex for data. The sending computer normally initiates each transfer, the receiver having the possibility of suspending ("lantening") the data transfer because of lack of buffer space or because the receiving task had not previously performed a read operation.

The status information is transmitted by single bytes, input and output under program control. Details of the structure of these bytes are given below. Data is transmitted in blocks of an even number of bytes (i.e. a whole number of 16 bit computer words), normally by the Direct Memory Access (DMA) method.

The protocol provides for two forms of data transfer: the "safe" mode (ITA(0) - see below) where the data transfer is not deemed to be complete until an acknowledgement has been received by the sender from the ultimate destination and the "unsafe" mode (ITA(1) - see below), where the transfer is assumed, by the sender, to be complete once the data has reached the nearest message switch. The "unsafe" mode has the obvious danger of a lack of end-to-end flow control, and the possibility of lost data.

The protocol comprises three phases:

1. An initiation phase: this determines the type of information that will be sent and resolves any conflicts that might arise due to both computers attempting to initiate a transfer simultaneously.

2. A "parameter" block: a block of twenty bytes of information containing the sending and receiving tasks' identifiers and computer numbers as well as the number of bytes that are to be transferred.

3. The data transfer: the "real" data is transferred to the receiving machine.

Acknowledgement blocks consist of only the first two phases, i.e. are "dataless" transfers. Certain functions (e.g. machine deadstart and bootstrap request) are signalled by the transmission of a single status byte.

For a complete explanation of the status codes, see below.
2.2 TRANSFER PHASE 1 (INITIATION)

The sender transmits an initiate status ("ITA") which determines the type of transfer requested. If the value of the ITA is not 6 or 7, the receiver should reply with an acknowledgement status ("IT") with code 0.

Sender ........................................ Receiver

\[\text{====================>}
\]
Initiate transfer (ITA)

\[\text{<====================}\\
If ITA code < 6 then acknowledge IT(0)

If the sender detects an initiate status instead of an acknowledgement, the conflict case has occurred, (i.e. the machines have attempted to initiate a transfer simultaneously). Both machines are conscious of this occurring and the conflict is resolved by the master waiting while the slave sends the usual acknowledgement.

-----------------------

\[\text{5In the diagrams that follow:}
\]

\[\text{====================> indicates a status transfer (one byte)}
\]

\[\text{--------------> indicates a data transfer (DMA)}
\]

\[\text{6Data transfers are half-duplex i.e. they may be made in both}
\]

\[\text{directions, but not simultaneously. Status transfers are full-duplex}
\]

\[\text{i.e. they may be made in both directions simultaneously.}
\]

\[\text{7Each machine connected to Omnet has a unique machine number which the}
\]

\[\text{program is able to read from the read only memory which also carries}
\]

\[\text{the Hardware Bootstrap program. In the conflict case the machine with}
\]

\[\text{the higher number is taken to be master and so carries on with the}
\]

\[\text{transfer.}
\]

- 5 -
2.3 TRANSFER PHASE 2 (PARAMETER BLOCK)

When the sender receives the acknowledgement he initiates a data write of 20 bytes containing the following information:

**Bytes**

0 - 7 : Sender's program identifier (8 character EBCDIC string)

8 - 15 : Receiver's program identifier (8 character EBCDIC string)

16 - 17 : Number of bytes of data to be sent

18 : Sender's computer number

19 : Receiver's computer number

This is followed by a "Stop" status to terminate the transfer.

**Sender**

```
-----------------------
20 byte parameter block
```

```
----------------------- STOP status
check for error
if recoverable (e.g. parity) then

<=---------------------- acknowledge IT(2) (repeat)
else

<=---------------------- acknowledge IT(3) (give up)
if no error then

<=---------------------- acknowledge IT(0)
```

- 6 -
2.4 TRANSFER PHASE 3 (DATA TRANSFER)

Sender  

------------------------------->
N bytes of user data

------------------------------->  STOP status
                           check for error
                           if recoverable (e.g. parity)
                           then
<==========================  acknowledge IT(2) (repeat)
                          else
<==========================  acknowledge IT(3) (give up)
                          if no error then
<==========================  acknowledge IT(0)
2.5 **STATUS BYTE CONTENTS**

Each status byte has three fields:

1. **Bit 7** (Most Significant Bit) - Hardware or Software status:
   
   0 - The status byte is a "software" status which is forwarded by the hardware interface to the link software.
   
   1 - The status byte is a "hardware" status which is decoded and handled within the interface hardware (e.g. clear link, read link configuration).

2. **Bits 6-5** - Status type:
   
   0 0 - Stop: follows data write operations.
   
   0 1 - Initiate "ITA": sent as the first command in a transfer to determine the transfer type.
   
   1 0 - Acknowledgement "IT": sent by the receiver in response to an initiate or stop status.

3. **Bits 4-2** - Status code (integer in range 0-7)

   Bits 1 and 0 (Least Significant Bit) are unused.
2.6 SOFTWARE STATUS CODES

Bit 7 of the status byte is zero.

2.6.1 Initiate Codes

The transfer type is signalled by the initiate transfer status - ITA ("Interruption d'Attention"):

<table>
<thead>
<tr>
<th>ITA Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0(***</td>
<td>Normal data transfer with a request for delivery confirmation - &quot;safe mode&quot;.</td>
</tr>
<tr>
<td>1(***</td>
<td>Data transfer, but no confirmation is required. If delivery is impossible, the block should just be thrown away - &quot;unsafe mode&quot;.</td>
</tr>
<tr>
<td>2(***</td>
<td>The sending machine has deadstarted. All pending transfers should be cleared and tasks waiting for a block to arrive should be re-activated.</td>
</tr>
<tr>
<td>3(***</td>
<td>Delivery notice of a data block that was multiplexed and delivery confirmation requested (type 0). The final status is given in the byte count field of the parameter block.</td>
</tr>
<tr>
<td>4(***</td>
<td>Indicates that a task which had been written to but replied &quot;later&quot;, is now ready for the data - &quot;later wake-up&quot;.</td>
</tr>
<tr>
<td>5(***</td>
<td>This is sent automatically to indicate that a task has terminated. All pending transfers should be cleared (either read operations or write operations which have had &quot;later&quot; replied to them).</td>
</tr>
<tr>
<td>6(***</td>
<td>Machine requests to be bootstrapped.</td>
</tr>
<tr>
<td>7(***</td>
<td>Machine crash or deadstart. Clear any pending transfers.</td>
</tr>
</tbody>
</table>

Notes:

1) *** Transfers which comprise phases 1, 2 and 3.
2) ** Transfers which comprise phases 1 and 2
3) * Transfers which comprise only the initial status.
### 2.6.2 Acknowledgement Codes After Parameter Block

The following table gives the acknowledgement codes - IT ("InTerruption") and associated replies that can be returned after the reception of a parameter block:

<table>
<thead>
<tr>
<th>IT Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All correct, proceed with the data transfer if the transfer initiate ITA = 0 or 1, otherwise finish.</td>
</tr>
<tr>
<td>2</td>
<td>A recoverable hardware error. The sender should repeat the parameter block.</td>
</tr>
<tr>
<td>3</td>
<td>An irrecoverable hardware error (or more than ten repeats). The transfer is abandoned.</td>
</tr>
<tr>
<td>4</td>
<td>The receiver recognises the receiving task identifier but that task has not yet declared a read buffer. The sender must then wait to be woken up when the read buffer is declared by a type 4 transfer - &quot;later&quot;.</td>
</tr>
<tr>
<td>5</td>
<td>The intended recipient program is unknown.</td>
</tr>
<tr>
<td>6</td>
<td>As for 4, but no space is available to remember the sending program's identifier, so he cannot be &quot;woken-up&quot; later.</td>
</tr>
<tr>
<td>7</td>
<td>The sender is attempting to send more bytes than the receiver has requested to read.</td>
</tr>
</tbody>
</table>
2.6.3 Acknowledgement Codes After User Data Transfer

The following table gives the acknowledgement codes ("IT") that can be returned by the receiver after the data transfer has been completed:

<table>
<thead>
<tr>
<th>IT Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All correct. The data has been given to the receiving task.</td>
</tr>
<tr>
<td>2</td>
<td>A recoverable hardware error. The sender should repeat the data block.</td>
</tr>
<tr>
<td>3</td>
<td>An irrecoverable hardware error. Abandon the transfer.</td>
</tr>
<tr>
<td>4</td>
<td>The data block was addressed to a machine other than the receiver, which will attempt to forward it. If the transfer was of type 0, an acknowledgement in the form of a type 3 transfer will be returned when the block reaches its final destination.</td>
</tr>
</tbody>
</table>

2.6.4 Transfer Termination Codes

Each data write operation is followed by a status ("Stop") which indicates whether the sender has detected any errors. The following table gives the possible codes:

<table>
<thead>
<tr>
<th>STOP Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No errors were detected.</td>
</tr>
<tr>
<td>2</td>
<td>A recoverable hardware error.</td>
</tr>
<tr>
<td>3</td>
<td>An irrecoverable hardware error. Abandon the transfer.</td>
</tr>
</tbody>
</table>
2.7 HARDWARE STATUS CODES

Bit 7 of the status byte is 1.

These are hardware functions decoded and executed within the hardware.

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Read Link Configuration - PDP11 only. The Unibus address of the Remote Status Register (see appendix F) is switched to a read only register for one read cycle only, in order to read the link speed setting etc. Nothing is transmitted on the serial link, only the local hardware is affected.</td>
</tr>
<tr>
<td>7</td>
<td>Clear Link. The major part of the non-computer bus related circuits are reset to zero. Both the local and the remote link interfaces are affected.</td>
</tr>
<tr>
<td>13</td>
<td>Clear Buffer. The input data FIFO buffers and related circuits are cleared. Both the remote and local link interfaces are affected.</td>
</tr>
<tr>
<td>17</td>
<td>Remote Bootstrap - PDP11 only. The receipt of this status forces a suitably equipped receiving computer to execute a Hardware Bootstrap program.</td>
</tr>
</tbody>
</table>
2.8 EXAMPLES OF COMPLETE TRANSFERS

2.8.1 Normal Data Transfer

The following diagram shows a complete transfer of one data block between adjacent computers, on the assumption that no errors occur and that the receiving task is installed:

\[
\begin{array}{c|c}
\text{Sender} & \text{Receiver} \\
\hline
\text{Initiate status ITA(0)} & \text{Acknowledgment IT(0)} \\
\hline
\text{Transfer 20 byte parameter block} & \text{Acknowledgment IT(0)} \\
\hline
\text{Status STOP(0)} & \text{Acknowledgment IT(0)} \\
\hline
\text{Transfer N bytes user data} & \text{Acknowledgment IT(0)} \\
\hline
\text{Status STOP(0)} & \text{Acknowledgment IT(0)} \\
\end{array}
\]
2.8.2 **Errored Data Transfer**

The following diagram shows a complete transfer of one data block between adjacent computers in the case that an error is detected by the receiver. As before, it is assumed that the receiving task is installed:

**Sender**

```
Initiate status ITA(0)
```

```
Acknowledge IT(0)
```

```
Transfer 20 byte parameter block
```

```
Status STOP(0)
```

```
Acknowledge IT(0)
```

```
Transfer N bytes user data
```

```
Status STOP(0)
```

```
Acknowledge IT(2)  *** Error detected by receiver
```

```
Repeat N bytes user data
```

```
Status STOP(0)
```

```
Acknowledge IT(0)  *** Successful transfer
```
2.8.3 **Multiplexed Data Transfer**

The following diagram shows a complete transfer of one data block from one computer to another via an Omnet multiplexor computer using ITA(0) type transfers. An error free transfer is assumed as is the presence of the receiving task.

**Stage 1:** The data block is sent to the multiplexor:

<table>
<thead>
<tr>
<th>Sender</th>
<th>Multiplexor</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>=================</td>
<td>-------------</td>
<td>===================</td>
</tr>
<tr>
<td>Initiate status ITA(0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledge IT(0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Transfer 20 byte parameter block</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status STOP(0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledge IT(0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Transfer N bytes user data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status STOP(0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledge IT(4)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Stage 2, the multiplexor forwards the block:

<table>
<thead>
<tr>
<th>Sender</th>
<th>Multiplexor</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>=======</td>
<td>------------</td>
<td>----------</td>
</tr>
<tr>
<td>Initiate status ITA(0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledge IT(0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transfer 20 byte parameter block</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status STOP(0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledge IT(0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transfer N bytes user data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status STOP(0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledge IT(0)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Stage 3, the multiplexor informs the sender of the successful completion of the transfer:

| Initiates ITA(3) |
| Acknowledge IT(0) |
| Transfer 20 byte parameter block |
| Status STOP(0) |
| Acknowledge IT(0) |
3. HARDWARE

3.1 DATA LINK INTERFACE HARDWARE

3.1.1 General Characteristics

1. The separation of data from control information.

Marker bits are added to the serial word. Data is input and output by Direct Memory Access (DMA) transfers, and control information ("status"), under program control. The status path is full duplex, control information may flow in either direction at any time. The data path is, however, half-duplex; data may flow in both directions, but not simultaneously. This restriction has its origins in the design philosophy of the original Omnet computers' input/output channels. There is the beneficial effect, especially with the PDP11, that only one DMA controller is required in the interface. This "lightens" the implementation considerably and has allowed, even with the use of standard interface components, the construction of the interface on one DEC standard system unit.

2. Hardware flow control of data transmission.

A first-in, first-out (FIFO) buffer memory is placed in the receiver's data input path to buffer fluctuations in memory access time. If this buffer fills beyond a preset limit, a "hold" status byte# is sent to the remote transmitter which then stops transmission. When the buffer empties, a "not-hold" status byte is sent to resume transmission. The handling of the "hold" and "not-hold" signals, being entirely within the hardware, is completely transparent to the software.

3. Hardware detection of errors.

A lateral parity bit is generated for every byte.

4. Bit synchronous, byte asynchronous transmission.

#The use of a extra bit in each serial word (see point 4) in order to be able to signal the "hold" or "not-hold" functions, is obviously rather inefficient. Every byte of information transmitted carries this overhead. In addition, the "hold" and "not-hold" control bit combinations are transmitted on a special serial byte, used for that purpose only. The more efficient way would have been to invent two extra hardware status bytes (see section "Status Byte Contents") and thus allow one of the control bits to be suppressed. The use of the existing system is due entirely to historical reasons and the wish to retain backwards compatibility.
Each computer word is disassembled into 8-bit bytes which are serialised as 13 bits:

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Start bit</td>
</tr>
<tr>
<td>2-9</td>
<td>8 bits of data or status</td>
</tr>
<tr>
<td>10-11</td>
<td>Transfer type (encoded thus:</td>
</tr>
<tr>
<td></td>
<td>0 = data</td>
</tr>
<tr>
<td></td>
<td>1 = status</td>
</tr>
<tr>
<td></td>
<td>2 = not-hold</td>
</tr>
<tr>
<td></td>
<td>3 = hold</td>
</tr>
<tr>
<td>12</td>
<td>Lateral parity (odd)</td>
</tr>
<tr>
<td>13</td>
<td>Stop bit</td>
</tr>
</tbody>
</table>

5. Interfaces for CII 10070, EMR 6130, PDP11 and LSI11 computers.

There is a large degree of commonality between the various interfaces, in both actual components and in the detailed design.

6. Change of line speed and/or transmission cable type.

This is made by changing only the line transmission system ("modem").

7. Interface construction.

The interfaces are constructed using transistor-transistor logic elements of the standard, Schottky, and low-power Schottky sub-families. Commonly available integrated circuits and other components are used as far as possible.

3.1.2 PDP11 Data Link Interface

3.1.2.1 General

The PDP11 interface implementation is described here in detail since it is most relevant to the present use of Omnet, but is essentially similar to the others. The LSI11 link interface is identical to the PDP11 version with the exception of the actual computer bus adaptation.

The PDP11 interface hardware is constructed on one PDP11 system unit, using as far as possible, Omnet and Digital Equipment Corporation standard interface components. Data is transferred by the interface at the Non-Processor Request (NPR) level by Direct Memory Access. It is programmed via seven registers accessible from the
PDP11 Unibus\textsuperscript{9}, although by using the possibility of separating the read and write functions of one Unibus address, only four Unibus addresses are in fact used. Two interrupts may be produced, each at a unique vector, to signal the arrival of a software status byte from the remote end and DMA word count zero (data transfer complete). These interrupts may be separately enabled by setting bits in the link's Command Register. A read-only register (the Configuration Register), is used to provide such information as: link serial number (unique for each link interface), transmission speed (may be used by the SMO software to determine the time-out period required for the various phases of the transmission protocol), etc.

3.1.2.2 Data Path

Data is input and output using DMA transfers. The DMA controller can perform Unibus data transfers, either to\textsuperscript{10} or from\textsuperscript{11} memory. In order to send data down the serial line to the remote end, transfers are initiated in the following manner. Firstly, the memory address of the start of the data block to be sent is loaded into the interface's Memory Address Register. This register, which is 18 bits long to accommodate the complete 256K bytes of address space, functions as an 18 bit up-counter, incrementing by two after every DMA cycle, the PDP11 being a byte addressing machine. Secondly, the length of the data block is loaded into the interface's Word Count Register. This is a 16 bit register which also functions as a counter, incrementing by one after every DMA cycle. Since it too is an up-counter, it is loaded with the 2's complement of the word count in order that it may count up to zero at the end of the transfers. Once the Memory Address and Word Count Registers are loaded, the DMA transfers are started by setting a bit ("GO(OUT)") in the interface's Command Register.

The DMA controller then obtains mastership of the Unibus and, performing a DATI cycle, fetches one word from the location pointed to by the Memory Address Register. This word is loaded into a 16 bit register in the interface (the Data Output Register). Either the upper or the lower 8-bit byte, depending on the condition of another bit ("WB") in the Command Register, is then loaded into the parallel to serial converter ("serialiser") together with the parity bit, which is generated to ensure that an odd number of bits are transmitted. The serialiser appends the appropriate control bits and the serial byte is sent to the line transmission equipment. The other byte is then sent.

\textsuperscript{9}"Unibus" is the Digital Equipment Corporation name for the unified, asynchronous bus system used by their PDP11 series minicomputers. Both "Unibus" and "PDP11" are registered trade names.

\textsuperscript{10}In PDP11 terminology, DATO transfers - Data Output from the Unibus Master

\textsuperscript{11}DATI transfers - Data Input to the Unibus Master
in a similar manner.

This process will continue, unless interrupted by the arrival of a "hold" condition from the remote end (see below), until the Word Count Register reaches zero.

Data arriving from the remote end of the link is treated in a similar way, but in the reverse direction. The DMA controller is set up in an identical manner to that described above, except that since DATO cycles are to be performed, a different ("GO(IN)") bit in the Command Register is set. When serial data arrives from the remote end, it is converted into a parallel byte by a serial to parallel converter ("deserialiser") which also determines, by the state of the serial control bits, whether it is a byte of data, status or is a "hold" or "not-hold" condition. The parity is at this point checked, and if necessary an error condition signalled by the setting of a bit in the interface's Error Register.

If the newly arrived byte is data it is then fed into an 8 bit wide FIFO memory, where it falls down to the lowest unoccupied position, in this case the first. Since this is the first byte to arrive it is immediately passed on to the 16 bit Data Input Register and strobed into either the upper or lower position, again depending on the state of the "WB" bit in the Command Register. When the second data byte arrives and is strobed into the vacant position of the register, the DMA transfer is triggered and the 16-bit data word is transferred into the memory location pointed to by the Memory Address Register. This process then continues, the Memory Address Register incrementing by two and the Word Count Register by one after every cycle, until either no more data arrives, or the Word Count Register reaches zero.

If data arrives in the FIFO\textsuperscript{12} buffer faster than it is being transferred to memory, the buffer will tend to fill. At a certain preset limit (normally one quarter full) the link's serialiser is commanded to send a byte to the remote end carrying the "hold" control bit combination. The remote end temporarily stops its DMA transfers. When the receiver's FIFO empties below the limit, the link's serialiser will be commanded to send the "not-hold" combination of control bits, and the data transfer is resumed.

As mentioned above, the 16-bit data words may be assembled or disassembled into 8-bit bytes in either order. This is to allow transmission of character or integer orientated data.

It should be noted that the DMA transfers are made one Unibus cycle at a time, with the controller relinquishing control after each word has been transferred. Mastership of the Unibus is requested each time

\textsuperscript{12}The FIFO normally used is of 63 bytes length, constructed using LSI circuit elements. It has the possibility of generating a signal at one quarter, one half or three quarters full which may be used to send the "hold" condition.
a word is to be input or output. This is the simplest form of DMA control and avoids any possibility of problems with mass storage devices on the Unibus being unable to obtain mastership of the Unibus in time to transfer their data.

3.1.2.3 Status Path

Transmission protocol control information or "status" is transferred in single 8-bit byte units. In order to send a status byte, the required value is simply loaded (written) into the interface's Local Status Register. On receipt of the status, the interface generates a parity bit and passes it or to the serialiser, where the serial control bits, identifying it as a status byte, are appended and the serialisation is started.

In the other direction, when the deserialiser identifies a serial byte arriving from the remote end as a status byte, the contents of the byte are examined to determine if it is a hardware or software status. If it is the former (bit 7 being 1), the byte is decoded and the appropriate action taken (see the chapter "Transmission Protocol"). If the latter (bit 7 at 0), the byte is strobed into the link's Remote Status Register. Its arrival in that register will set a flag ("DONE") in the Command Register and can cause a computer interrupt to be generated, if desired.

The Remote Status Register and the parallel output register of the deserialiser are used together to form a two position FIFO buffer for status bytes. Should the computer be unable to treat a status byte immediately, it will be stored in the Remote Status Register until the software reads it and acknowledges its receipt by explicitly clearing the register. If a second status should arrive whilst the first is still waiting, it will be stored in the deserialiser until the Remote Status Register is cleared, when it will be transferred to that register and the "DONE" flag again set.

3.2 LINE TRANSMISSION EQUIPMENT

The basic computer interfaces send and receive serial data to and from the transmission equipment in unmodulated binary form at transistor-transistor ("open collector") logic levels. The interfaces require an external clock signal for the transmission and reception of serial data. This clock is normally provided by the line transmission equipment. The job of the line transmission equipment is, therefore, to provide the necessary clock signals, to perform the level conversion to and from the transistor-transistor logic levels, to modulate and demodulate the data (if required) and to drive and receive signals from the transmission lines."
The connection from the transmission equipment to the interface comprises two short (approximately two metre) cables, each of two twisted pairs. Each cable carries a data and a clock signal for the transmit and receive directions.

Three types of line transmission equipment are used in the Omnet system. They are associated with three speed ranges and transmission cable types.

3.2.1 **Co-axial Cable: 5 Megabaud**

The system used to transmit at 5 Megabaud over co-axial cable was developed at CERN. It uses the phase-code modulation technique which reduces the transmitted bandwidth and, in addition allows the transmission clock to be reconstituted at the receiving end. Transformer coupling is used to eliminate problems with ground levels and the transmission of the direct current components of the signals. Emitter coupled logic is used for the modulation, demodulation and clock generation circuits.

This system has been used successfully at 5 Megabaud on links of upto 1.3 kilometres using high quality cable.

3.2.2 **Twisted-quad Cable: up to 5 Megabaud**

This system was designed to use the extensive network of high quality twisted quad cables which already existed at CERN. Transistor-transistor logic elements are used for the clock generation and synchronisation circuits and proprietary integrated circuits as line driver and receiver elements. Straight binary data is transmitted without modulation. The receiver clock is, therefore, synchronised locally to the incoming data. Optical isolation is used for the same reasons as the transformer coupling in the co-axial system.

This system is normally used at 2.5 Megabaud line speed for links of up to 1.5 kilometres. For longer links, if maximum speed is essential, line repeaters are used. Otherwise the line speed is reduced to suit the cable length.

This system was adapted from that developed at CERN for the Cernet computer network and incorporates open-line detection circuits.

\[\text{13 All transmission lines used are 4-wire, providing a full duplex path between computers.}\]
3.2.3 Telephone lines: up to 36 kilobaud

The use of voice grade telephone lines as point to point links is very interesting due to their universal availability. At CERN, every office, laboratory and experimental area is equipped with such lines. Base-band modems are used to exploit the voice grade lines at as high a speed as possible. They are connected to the computer interfaces by means of a level converter box, which provides the CCITT V24 interface required by the modem.

On short lines within CERN, speeds of up to 36 kilobaud are used. The same system has also been used at 9.6 kilobaud for a link of 12 kilometres of voice grade telephone lines outside CERN.

3.3 SWITCHING NODES

The Omnet message switching nodes\textsuperscript{14} are PDP11/10 computers with the following configuration:

1. 28K words of core memory
2. Console terminal (Decwriter II)
3. High-speed paper tape reader
4. 96 words of read only memory (for the paper tape and link bootstrap programs)
5. Unibus display (a passive device for monitoring the state of the Unibus signals - used as a diagnostic aid)
6. Address coincidence detector ("stack-trap") - used by the SMO operating system to protect its task stacks
7. Data link interfaces, as required, together with the appropriate line transmission systems. The interfaces are equipped with control panels which allow the links to be independently cleared and switched off-line and which give, as a diagnostic aid, a display of important internal signals.

\textsuperscript{14}The terms "node", "multiplexor" and "message switch" tend to be used interchangeably in the commonly used Omnet terminology.
4. SOFTWARE

There are two different operating systems for the PDP11 that are used in Omnet. In this chapter, we outline the main features of each system and describe how the Omnet software is implemented.

4.1 SMO (SMALL MACHINE ORGANISER)

The SMO system performs a vital role in Omnet and therefore its features will be described in some detail. All multiplexing nodes in Omnet are PDP11/10 machines running this system. SMO is a memory-based operating system that was developed at CERN and is designed to run in small PDP11 configurations. SMO does not support the PDP11 memory management unit and therefore usable memory is limited to 28K words. The system is written in a high level language, PL-11\(^{15}\), and all program development and system generation work is carried out on a larger support machine. All user tasks for SMO are also written in the PL-11 language and are compiled and built on the support machine.

4.1.1 Design Criteria

The original requirement, which gave rise to the SMO system, was to provide multiplexing software for the CII 10070 data links. It was quickly realised, however, that it would be just as easy to provide a general purpose operating system and then implement the multiplexing software within its framework. In this way, data link multiplexing could be performed in parallel with the execution of other tasks. The operating system would be multi-tasking but, in the interests of performance, would not have extensive memory protection schemes. High speed inter-task communication was essential as was the ability for tasks to separate their code and data segments. This latter requirement was accomplished by means of a general, dynamic memory allocation scheme. The input/output structure would be simple and the allocation of system resources would be with a "secure" and "release" mechanism.

4.1.2 System Structure and Generation

In order to minimize the support requirements, the SMO system has a very modularised structure. Each code unit is separately compiled onto a binary library file consisting of unlinked, relocatable loader units. Most units comprise three or four subroutines and have a well defined and documented interface to other modules. All these modules

\(^{15}\)PL-11 is a structured, high level assembler language developed at CERN for the PDP11 computer.
make up the system library which is common for all configurations.

The configuration dependent information is contained in a special module, known as "LOWCORE" because of its position in the final system. This module is created by a program called "SYSGEN" which takes as input the user's specification for his configuration. The "LOWCORE" module then contains all the tables required by the system and references to the modules in the library file needed for that particular configuration. These modules are then extracted automatically from the library and linked to produce a core-image ready for loading into the target machine. The combined system generation and link-edit process takes about 15 seconds on the CERN IBM 3032 allowing users to modify their system configuration "in real time".

4.1.3 Operator Communication - "KEYTASK"

As with most systems, the user dialogues with SMO via an operator communications task which provides him with a set of commands to control the system. In SMO, the user specifies in the system generation input the commands that he wishes to include from the system library. He can also request to have user written commands included from his private library. Standard commands available include:

1. The activation, termination, and interruption of tasks.

2. Dynamic loading and creation of tasks.

3. Display of current tasks, devices, loaded code segments and system status.

4. Dumping memory, task register blocks, and stacks.

5. Modification of memory and insertion of breakpoints on execution.

6. Time handling.

4.1.4 Multi-Tasking Features

The kernel of SMO provides the task switching environment and any number of tasks can compete for system resources. Task priority, a number in the range 0 - 127, is used for scheduling and the allocation of central processor time. The kernel handles allocation of resources ("secure" and "release"), task activation and termination, as well as time and calendar handling. The kernel also provides the standard input/output interface to stop a task until input/output has terminated or an optional timeout has occurred. SMO tasks can either be
embedded in the system at system generation time or can be loaded dynamically into memory via the data link.

A mailbox facility provides inter-task communication and, in addition, tasks are able to exchange buffers by passing the address of the buffer to the partner task. This avoids any copying operation in memory and this is used extensively by the SMO linktasks which run in the Omnet multiplexor machines.

4.1.5 Input/Output System

The device independent part of the input/output system uses the standard philosophy of associating a "device block" with each device. Some information in the block is device independent (e.g. owner task, busy status) while the remainder may be device dependent (e.g. terminal characteristics).

Also associated with each device type is a re-entrant driver containing two parts. The first part is an "EMT" routine (see below) which is responsible for initiating input/output activity on the device. The second part is an interrupt handler responsible for re-activating the task on input/output termination, or in the case of programmed input/output, initiating the next operation. In the system generation input, the user can specify a standard device type by giving the name and interrupt vector address or he can declare a "user device" and specify explicitly the driver and device block to be used. Drivers specified in this way will be included from the user's library file in the link-edit step for the system generation.

4.1.6 Re-entrant Tasks in SMO Systems

The SMO system provides several features to support re-entrant tasks in an easy and convenient manner. The code and data segments of a task can easily be separated by making use of the SMO dynamic memory buffer allocation. All memory space above the "resident" part of the system to the top of memory is available for dynamic allocation. Memory is allocated in units of four bytes using an algorithm that minimises core-fragmentation by coalescing adjacent buffers when a buffer is released.

The re-entrant code handling routines also use the memory allocation system to place dynamic code when it is requested. Loading is done transparently for the user when he requests a code segment. If the code segment is already present in the system, a user count is incremented. When a task releases a code segment, either explicitly via an EMT call or by terminating, the user count is decremented. Only when the user count is zero is the code released and the space freed. This then allows tasks to overlay and to share code as required.
4.1.7 Debug and Trace Facilities

The system checks parameters that the user provides on monitor calls and provides proper task abort and termination sequences. The system also checks itself for internal inconsistencies and aborts itself (system crash), if errors are detected. The SMO system crash handling outputs the current registers, gives an error message and dumps a backtrace containing the most recent monitor calls made and their arguments. Any program is allowed to insert traces using a special trap instruction to facilitate user fault-finding.

4.1.8 Library Utility Routines

Many utility subroutines form part of the resident system. They are coded in a re-entrant manner and are called EMT routines after the instruction used to access them. The user can decide at system generation time which EMT routine he wishes to include and a table is generated which is used at run time to interpret EMT calls and branch to the appropriate routine. EMT routines are available for terminal input/output (including echo and control character handling), communications input/output, formatting input/output, dynamic code loading, etc.

4.1.9 Omnet Software Under SMO

For each physical Omnet interface on an SMO machine, there is an associated Linktask. Linktask is re-entrant and conditional compilation of the source code is used to generate either an "end user" (single link), a multiplexor or a Cernet gateway version.
The source machine initiates the transfer by sending a parameter block to the multiplexor Linktask-1. The parameter block of 20 bytes identifies the source and destination tasks and indicates the number of data bytes to be transferred. Linktask-1 requests a buffer of this size and, if this request is successful, it replies to the source task to begin the data transfer. If the buffer request is not satisfied, then the Linktask in the source machine is told to wait ("later") until a "later wakeup" request is issued from the multiplexor. When the data transfer is complete, the ownership of the data buffer is passed to Linktask-2 where it joins a chain of multiplex work blocks. Linktask-2 is then activated by Linktask-1.

Each Linktask operates in an asynchronous manner and can be activated in one of two ways:

1. Data from the link. (i.e. a status from a remote machine)

2. By a request from another Linktask. In this case, it finds the next item of work by scanning two work chains:
   a) A chain of multiplex data blocks, i.e. blocks of data to be forwarded to local or remote tasks.
   b) A chain of user connect blocks. These represent requests for link input/output from other user tasks within the multiplexor.

When it finds a multiplex block to be treated, Linktask-2 initiates the transfer of the data to the target task.

The provision in SMD for tasks to exchange data buffers merely by the transfer of address and ownership, offers a high speed inter-task communication mechanism. This in turn means that the message
switching software can be implemented in an elegant way whilst retaining a high level of performance in terms of data transfer rates. (See Appendix "Performance Measurements").

4.2 THE RSX-11 SYSTEM

RSX-11 is a real time system for the PDP11 written by the Digital Equipment Corporation. It exists in a disc based version, RSX-11M and a memory only version, RSX-11S. These systems have been adopted as a standard at CERN and are used in many of the data acquisition machines at physics experiments. We shall not describe the RSX-11 system itself in any detail but rather concentrate on how the Omnet software is implemented under RSX-11M/S. For details of the RSX-11M system, we refer the reader to the documentation issued by the Digital Equipment Corporation.

4.2.1 Omnet Implementation Under RSX-11

The Omnet data link software is implemented as a standard RSX-11 driver. Much of the source is written in PL-11 with a few specialised system interface routines and device tables written in MACRO-11. Besides the basic input/output functions, the Link Driver handles the multiplexing and demultiplexing of data link messages between the requesting tasks in the RSX machines.

The Omnet protocol allows the Link Driver to set up the data transfers directly to and from the user's buffer, thereby avoiding costly intermediate storage of data. The status exchange over the data link is handled by an interrupt dispatcher and a number of service routines.

Tasks interface with the Link Driver in the standard way by issuing Q10 directives. This causes a "work request" in the form of an input/output packet to be queued to the driver. The driver dequeues the input/output packets one at a time and treats the input/output request.

The driver occupies about 1.7K words. In addition to this, each "logical link" requires a small buffer which is allocated from the RSX system buffer pool.

The performance of the Link Driver is a function of many variables, the central processor utilization, protocol synchronisation, Unibus load etc. On a directly connected link between two PDP11/40s a maximum throughput of approximately 200 Kbytes per second was measured with a central processor load of about 50%. At normal transfer rates

\[ \text{MACRO-11 is the DEC PDP11 assembly language.} \]
of 40 Kbytes per second, the central processor load is about 10%; the driver is, therefore, capable of handling several logical links simultaneously.

As in SMO, a diagnostic trace facility is used to record all link operations and monitor calls made by the Link Driver.

4.3 Bootstrapping and Down-Line Loading of Tasks

Many of the PDP11s connected to Omnet are machines with no secondary storage. Therefore, facilities have been provided to allow the down-line loading or "bootstrapping" of these systems from a "host" machine via the data link. The same facilities are used for dynamic task loading in SMO and RSX-11S systems and for loading stand-alone test programs into an empty machine. The host machines in Omnet are RSX-11M systems which are able to access the central IBM machines. This access is via the Omnet/Cernet gateway. In principle, any RSX-11M system can be used as a bootstrap host ("boot-host") although, for operational convenience, a central PDP11/60 is used to store all RSX-11S and SMO system image files. RSX-11S systems are generated on the PDP11/60 itself whereas SMO systems and tasks are built on the IBM 3032 and are transferred to the PDP11/60 with a Cernet file transfer utility. Bootstrapping a typical SMO system of about 8K words takes in the order of 10 seconds real time.

4.3.1 Hardware and Software Bootstraps

Bootstrapping in Omnet is always a two stage request involving a "Hardware Bootstrap" and a "Software Bootstrap". The exact form of a "bootstrap request" depends on whether the machine to be loaded is connected directly to the host PDP11 or whether it communicates via an intervening SMO multiplexor. The two cases are treated below. In either case, the task which treats the bootstrap requests in the host machine is the File Manager. This is a passive process which receives remote file access requests in the form of data link messages and processes them on its local file system.

At the line level, the Software Bootstrap does not perform any sophisticated error recovery, apart from data re-transmission in the case of parity error. In the case of any protocol desynchronisation, the program simply saves the last received software status and halts the computer. At the file access level, any errors (e.g. file not found) are indicated by a plain language message being listed on the user's terminal.

- 30 -
4.3.2 Directly Connected Machine

Every PDP11 in Omnet has a small read only memory containing a 30 word program; this is the Hardware Bootstrap. On executing this program a bootstrap request status byte - ITA(6) is sent to the Link Driver in the RSX host machine. The Hardware Bootstrap then executes a DMA read request on the link. In the RSX machine, the status is passed by the Link Driver to the File Manager which issues a single write operation. The message written by the File Manager is a stand-alone program called the Software Bootstrap. When the Software Bootstrap has been received in the target machine, control is passed from the Hardware Bootstrap to location (octal) 40 - the start address of the Software Bootstrap.

Once in execution, the Software Bootstrap examines the switch register to see if the default system is to be loaded or if it should prompt the system console with a "BOOTFILE?" request. This allows the user to specify the file to be loaded as well as the name of the remote loading machine. The Software Bootstrap then issues standard file access requests to the File Manager to open and read the required file. The first blocks of data are the header blocks which contain the start address of the system being loaded. After reading these header blocks, the software bootstrap copies itself to the top of memory so that the system can be loaded from location zero upwards. The file blocks are then loaded in sequence until "end of file" is reached. Control is then transferred to the start address of the loaded system.

4.3.3 Machine Connected via a Multiplexor

The operation of the Hardware Bootstrap proceeds as explained above except that the ITA(6) status is received by the Linktask in the SMO multiplexor. In response to this, Linktask generates a special message that is addressed to the File Manager in the "Boot-host" machine. In principle, there is one "Boot-host" for all machines in Omnet but it is possible to change the "Boot-host" of any particular machine either by using a utility program in the nearest SMO multiplexor or at the multiplexor's system generation time.

The bootstrap request message is received by the File Manager as a normal read request and results in the Software Bootstrap being written in a single write link operation. From then on, the dialogue proceeds in an identical manner to that explained in the section above.

- 31 -
4.3.4 Bootstrapping Procedure

When bootstrapping a machine, bit 1 of the switch register is used to select the mode of operation. If bit 1 is unset, then the default system is loaded. If bit 1 is set, then the user is prompted with 'BOOTFILE?'. The reply is a file specification of the form:

<BBBB><YYY><[n,m]><filename.ext>

<BBBB> is a 4 character Omnet machine name, <YYY> an RSX-11 device specification, and <[n,m]> an RSX-11 UIC code.

4.3.5 Dynamic Task Loading for SMO and RSX-11S Systems

Loadable tasks for SMO are generated on the CERN central IBM machines, the task image then being transferred to an RSX-11M system. Tasks for RSX-11S are generated on the RSX-11M system directly. Both systems incorporate a loading task ("LOAD" option for SMO and "LKL" for RSX-11S) which issue the remote file access commands to open and read the specified file. As with bootstrapping, the remote process which treats these commands is the File Manager on an RSX-11M machine.

Tasks for SMO systems are loaded into dynamically allocated memory and are relocated by the loading task. The relocation information for the task is contained in the task image file generated on the IBM system. For RSX-11S systems, a partition is specified when the task is built (TKB time) and thus the loaded image is already relocated to a particular address.

As with bootstrapping, errors in file access are indicated by a plain language message which is output on the user's terminal.
5. OMNET/CERNET GATEWAY

5.1 GENERAL

Cernet is a CERN-wide, generalised, packet switching network using Modcomp II and Classic computers as switching nodes. Special purpose connections exist for IBM 370 and CDC 6000 series computers, whilst minicomputers\(^{17}\) are connected using the Camac interfacing system. Omnet PDP11 computers can communicate with any Cernet connected computer by means of the special "gateways" described below.

The Omnet/Cernet gateway performs the protocol conversion "on-the-fly". It assumes that the high level software ("Transport Manager") in the user machine has obeyed the Cernet End-to-End protocol and that messages have been broken into Cernet packets before transmission over Omnet. The Omnet software sends the packets with the normal Omnet protocol to a pseudo-task ("NETINP") in the pseudo-machine ("CH01"). These Omnet messages are routed by Omnet to the gateway nodes, where the special software recognises the task and machine names. The Omnet data blocks, which are in fact the Cernet packets, are then transmitted to Cernet by a special hardware interface which obeys Cernet line protocol. Incoming Cernet packets are treated in the reverse manner, being routed to their Omnet destination from the pseudo-task "NETOUT" in the pseudo-machine "CH01".

In the present Omnet configuration there are two "gateways" to Cernet providing the means of communication with the CERN central IBM and CDC computers.

5.2 HARDWARE

The PDP11 Cernet Data Link Interface which connects to the PDP11 Unibus, is designed to implement the Cernet line protocol. Data is transferred using DMA, whereas control information is sent and received under program control.

Bit synchronous, byte asynchronous serial transmission is used with the data serialised as 19 bits:

- Start bit
- 16 data bits
- Lateral parity bit
- Stop bit

The hardware is designed to transmit packets of data with a maximum length of 1023 words. Data and control information are protected by

\(^{17}\)Digital Equipment Corporation PDP11 and VAX, Norsk Data Nord 10 and Hewlett-Packard 21MX.
the lateral parity bit, data is also protected by a cyclic redundancy check made on each packet transmitted.

The interface is organised as two separate logical devices, the "Sender" for transmission of data and control information and the "Receiver" for the reception of data and control from the remote computer. These two logical devices are physically integrated to economise on space, Unibus loading, components etc.

The interface is controlled by sixteen registers addressable from the Unibus. Two registers (the Packet Word Count Registers) are loaded automatically by the interface hardware. Two other registers (Cyclic Redundancy Code Register and Configuration Register) are included for test and operational purposes. Two interrupts at unique vectors can be produced, one each for the Receiver and the Sender. Each interrupt has several sources, each of which may be enabled independently.

The interface features first-in, first-out (FIFO) buffer memories in both the data and control paths, the data FIFOs being able to store one complete data packet of up to 1023 words.

5.3 GATEWAY SOFTWARE

5.3.1 Functional Description

The SMO gateway software is built up of three levels of routines:

1. The input and output interrupt handlers, which drive the special hardware connecting to Cernet.

2. The Cernet line protocol routines for input and output.

3. Two routines which perform the wrapping and unwrapping of Omnet message headers and which control the above two levels of routines.

The Unibus interface is treated by the higher level software as four independent devices in order that data traffic may flow in both directions and to allow control words to be transferred amongst this data. The operating system device control blocks for these devices contain the values of the addresses of the various control registers of the interface and are set up at SMO system generation time. The device blocks are also used to transfer information between the various software levels.

The input interrupt handler will perform a logical OR into a reserved word of the device block of any incoming control word. If a
task is waiting for such a control word, it is woken up by a call to
the monitor. If the task is not yet ready, and this is a request for a
word count, the interrupt handler itself will return a control word
with zero word count as an acknowledgement. Unexpected data received
and DMA done interrupts are ignored, but lead to entries in the SMD
system trace buffer.

On output, a timer is set for the interrupt expected on DMA done.
This is not strictly necessary for normal operation, but is included
in order to register potential errors.

The packet input/output routines implement the standard Cernet
line-protocol; they contain a large number of optional "debug"
statements controlled by one variable which may be set to inhibit
printing in the normal mode of operation. In addition, several checks
are carried out during transfers which can give rise to messages which
are always listed on the computer's console.

The Cernet packet receive routine is the master in all transfers.
It sets itself ready to receive a word count, and then issues a "ready
for word count" control word. If no response is received within a
given time, the request is re-issued; if a zero word count is
received, a delay occurs before the request is repeated. On receipt of
a good count, a buffer is allocated to take the expected data, a
"ready for data" control word issued and a timeout started again. An
error will restart the whole process (up to a certain maximum number
of times). On successful receipt of word count and data, an
"acknowledge" control word is issued, and the packet input is
complete.

Once a Cernet packet has been received, a standard Omnet header is
appended and the destination within Omnet extracted from the lower
byte of the Cernet destination word. The destination and source tasks
are defined as NETOUT and NETINP. The block is then forwarded as an
Omnet ITA(1) type transfer.

The Cernet packet send routine is a slave which obeys the control
words it receives from the Cernet connection. It ignores any Omnet
data transfers destined for the pseudo machine CN001, of a type other
than ITA(1). Only the Omnet data block is passed on to Cernet, as
this is assumed to be a complete Cernet packet in the correct format.
The sending of a packet is complete only when a control word
containing an "acknowledge" has been received from Cernet.

5.3.2 Flow-Control Omnet to Cernet

Cernet is designed with end-to-end flow control on a message
(normally multi-packet) basis. It assumes sufficient capacity in the
communications subnet to absorb the packet traffic such messages will
entail. There is no flow control on a packet by packet basis.
Omnet, on the other hand, requires every data block transfer to have been preceded by a read request from the receiving task, or at least an outstanding "later"\textsuperscript{18} to allow any unsolicited data to be held queued. If this is not the case, an error message is issued and the data block is discarded.

In practice, this means that once the Omnet software has issued a request for data to a Cernet Transport Manager, it may not be able to digest the data sent to it, as fast as it is being sent. This will lead to data being discarded, higher level error recovery by re-transmission of the current message and consequent loss of performance.

The solution to this problem which has been adopted has been the implementation of multi-buffering techniques in the high level software with the Omnet software having a sufficient number of outstanding "later" requests in order to ensure the availability of buffer space.

\textsuperscript{18}"Later" is Omnet terminology for the process by which the transmission protocol can answer a transfer initiate with an ITA(4) acknowledgement. This informs the sender that the receiving task is not ready for data and that he will be "woken up later" by an ITA(4) initiate status when the receiver is ready. See the chapter "Transmission Protocol".
Appendix A

OMNET TOPOLOGY

The Omnet configuration is symmetrical in two basic halves. The two central Cernet gateway nodes, which form the hub of Omnet, are independent of one another; each has a separate bootstrap path to the Omnet host computer. The cross link between the two gateway nodes provides the path for inter-Omnet communication since the Omnet host (which runs RSX-11M) has no multiplexing possibilities.

In general the line speed used is 2.5 Megabaud. The exceptions are:

• Intercommunication within clusters of PDP11s, where co-axial cable links are used at a line speed of 5 Megabaud.

• Long links, greater than 1.5 kilometres of twisted quad cable, where the complication of line repeaters is not justified.

• Telephone line links.
A.1 CONFIGURATION DIAGRAM

The following is the Omnet configuration as at December 1979. The names EMC, OMEGA etc., refer to the experimental, or other, facility.

---

**Diagram Description:**

- **IBM | CDC**
- **CERNET**
- **MODCOMP | MODCOMP**

- **SFM**
  - 11/45
  - 11/20

- **AFS**
  - 11/60
  - 11/60
  - 11/40

- **NRTH AREA NODE**
  - 11/34
  - 11/34
  - 11/60
  - 11/04
  - 11/45
  - 11/40

- **GATEWAY NODE 1**

- **OMNENET HOST**
  - 11/60

- **NODE**
  - LSI11 11/20
  - 11/04 11/10
  - 11/20 11/10
  - 11/20 11/34
  - 11/20 11/40
  - 11/34 11/04

- **GATEWAY NODE 2**

- **OMEGA NODE**
  - 11/40
  - 11/34
  - 11/40
  - 11/40

- **EMC NODE**
  - 11/70
  - 11/70
  - 11/70

- **ERASME NODE**
  - 11/60

- **11/34**

- **11/34**

- **11/35**

**Notes:**

- *** Link used for bootstrapping gateway node 2 only
- NODE indicates that there is a local cluster of computers connected around a PDP11/10 node.
Appendix B

PERFORMANCE MEASUREMENTS

B.1 OMNET

The following measurements were made of Omnet's data throughput, using the Omnet high-level test, "Testlink". The hardware was, where appropriate, a standard PDP11/10 Omnet node, a PDP11/34 running RSX-11M and a PDP11/04 running SMO. ITA(1) type transfers were used throughout.

The table shows the effect of the line speed and of the operating system overheads on the link throughput. Note that at 5 Megabaud line speed, the theoretical maximum data transfer rate is 385 kilobytes per second.

<table>
<thead>
<tr>
<th>Data Block Size (bytes)</th>
<th>Throughput (Kbytes/sec) at line speed (Mbaud)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SMO-SMO</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td>200</td>
<td>18</td>
</tr>
<tr>
<td>500</td>
<td>38</td>
</tr>
<tr>
<td>1000</td>
<td>69</td>
</tr>
<tr>
<td>2000</td>
<td>115</td>
</tr>
<tr>
<td>5000</td>
<td>177</td>
</tr>
<tr>
<td>10000</td>
<td>229</td>
</tr>
</tbody>
</table>
B.2 **OMNET/CERNET GATEWAY**

The figures given below were extracted from the results of a series of tests made to measure the overall performance of Cernet. They represent the average transfer rates between a PDP11 program and the IBM file manager.

<table>
<thead>
<tr>
<th>Data Block Size (Kbytes)</th>
<th>Throughput (Kbytes/second)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Write</td>
</tr>
<tr>
<td>2</td>
<td>4 (10)</td>
</tr>
<tr>
<td>5</td>
<td>11 (19)</td>
</tr>
</tbody>
</table>

The figures given in parenthesis were obtained using a version of the test program working in a double buffered mode.
Appendix C

RELIABILITY

C.1 ERROR RATE

This is very difficult to measure, but from tests performed on a typical link using the 5 Megabaud transmission system, it would seem that the rate of detectable errors is less than one bit in 10^8 X 11. This represents one error every four months in a typical installation, assuming a 5% utilisation.

C.2 FAILURE RATE

The PDP11 interface has, from statistics collected over a period of some years, a mean time between failures of about four years. The failure rate of any particular installation depends heavily on such considerations as the cooling, the power supply stability and on the amount of manual interventions on the link hardware (i.e. the amount the hardware is disturbed for reasons unconnected with itself).

A policy of long-term burn-in for all of the components has enabled a very low integrated circuit failure rate to be achieved. The majority of problems found are to be in the areas of cable connections, bad contacts and the occasional integrated circuit failure.

The most difficult problems to solve are such things as residual software problems, routing errors and the misuse of Omnet facilities for which, as yet, no checks are made in the software (such as attempting to transmit too large data blocks etc.).
Appendix D

TEST FACILITIES

D.1 BASIC LOW-LEVEL TESTS - "CHECKLINK"

A series of low-level MACRO-11 test programs are used to test the basic functions of the link hardware. Data transfers and status exchanges are tested with the link in "local-loop" (i.e. with the serial data output connected to the serial data input). These test programs are assembled into one large program, together with some very simple monitor routines providing error logging and keyboard input facilities.

Although the quantity of data that may be circulated is limited by the size of the input FIFO buffer (data may not be output and input simultaneously), experience has shown that tests in local-loop are sufficient to uncover the vast majority of problems. Tests at a higher level remain essential not only for on-line fault-finding, but to be sure that any new hardware is in a working state.

D.2 HIGH-LEVEL TEST - "TESTLINK"

Testlink is a task which runs under both RSX-11M and SMO, thereby allowing tests to be made with minimum disturbance to other users of the computers. It provides the means of task to task communication either to another copy of itself in the same machine or to a similar task in another machine. If ITA(1) type of transfers are used, Testlink is also able to "bounce" messages off the nearest Omnet node. Testlink allows the mode of transmission, the block length, data checking etc., to be chosen.

D.3 OMNET/CERNET GATEWAY TESTS

Here again the tests are at two distinct levels, although the high level test is less convenient than the Omnet-only "Testlink" program mentioned above.
D.3.1 Basic hardware tests

These consist of a series of low-level MACRO-11 test programs, each of which is designed to test a particular feature of the gateway interface hardware. Unlike the equivalent Omnet tests, the large FIFO buffer memories and the full-duplex DMA controller allow up to one complete packet of data to be circulated in "local-loop".

D.3.2 High-level tests

The method used to test the gateway mechanism is simply to run a special task in a "user" PDP11 running RSX-11M, connected to the gateway in question. This task, "Bounce", is able to communicate using the full Cernet end-to-end protocols, with partner tasks running in the central IBM and CDC computers. This program should be considered, not as a hardware diagnostic facility the same sense as Testlink for an Omnet link, but rather as a diagnostic aid for non-specialist users.
Appendix E
OPERATIONS

Since the interconnection of Omnet and Cernet a considerable effort has been made to integrate Omnet into the CERN Computer Centre operations. In its early days, the test and diagnostic facilities were limited and it was very much a user-operated system. The "private" Omnet nodes, i.e. those closely associated with one experimental facility, are still intended to be operated, where necessary, by the user. The "public" nodes, such as the two gateways to Cernet, are operated by the Cern Computer Centre operations staff.

In order to make this possible, the Omnet topology was designed to allow as much redundancy as possible, given Omnet's inherent limitations, especially the lack of redundant paths and automatic re-routing. In addition, a carefully worded operations guide was written giving, as far as possible, unambiguous procedures in the form of flow-charts, for the user and operator to follow in the case of problems.

A link monitor program ("LKMON") was implemented. This program, running in the Omnet host computer, provides the operators with a frequently updated over-view of the state of the various Omnet links. Every fifteen minutes transfers are made to all possible machines (all nodes and PDP11s running RSX-11M and the appropriate File Manager program). The results of these tests are displayed on television monitors in several convenient places. Long term statistics may be extracted from LKMON, in order to assess the performance over an extended time period.

The two Omnet/Cernet gateway nodes provide a large measure of mutual redundancy, but do not entirely back each other up. In the case a catastrophic failure of one node, however, it would be possible to connect all the users data links to the remaining machine. Similarly, (and more practically) in the case of the failure of one of the Cernet gateway connections, Cernet-bound traffic can be re-routed across the inter-gateway link to the good machine. This requires the reloading of the affected machine with a different version of the operating system carrying modified link routing tables.
Appendix F

PDP11 DATA LINK INTERFACE REGISTER LAYOUT

The PDP11 Data Link Interface is controlled via seven registers accessible from the Unibus. Details are given below of the usage of each register.

"Read" implies that a bit may be read from the Unibus, without changing its contents. "Write" implies, unless otherwise stated, that the bit may be set or reset by loading with one or zero respectively.

"INIT" refers to the Unibus general initialisation signal.

F.1 COMMAND REGISTER

This register is used by the program to control and obtain the status of the DMA controller and by the hardware to flag the arrival of software status bytes and any error conditions.

Unibus Address: N (Default 767730)

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>GO(OUT)</td>
<td>Read/write. When set the DMA controller requests Unibus mastership and transfers data using DATI cycles from memory to the interface. Resets READY (bit 07). Reset by INIT, by GO(IN) (bit 08), and by READY (bit 07) being set.</td>
</tr>
<tr>
<td>01</td>
<td>Not Used</td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>Not Used</td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>XBA16</td>
<td>Memory Address Extension bit 16 - read/write. Forms with the MAR and XBA17 an 18 bit counter. Reset by INIT.</td>
</tr>
<tr>
<td>04</td>
<td>XBA17</td>
<td>Memory Address Extension bit 17 - read/write. Forms with the MAR and XBA16 an 18 bit counter. Reset by INIT.</td>
</tr>
<tr>
<td>05</td>
<td>WB</td>
<td>Word/Byte - read/write. Controls the order of assembly and disassembly of 16 bit words into</td>
</tr>
</tbody>
</table>
bytes. When reset, the low order byte is transmitted first and the incoming bytes are reassembled with the first to come in the low order position (character mode).

06 RIE Ready Interrupt Enable - read/write. When set enables the READY interrupt.

07 READY Read only. When set the interface is ready for a DMA operation. An interrupt will be made, if enabled (RIE - bit 06), except after INIT. Reset by GO(OUT) and GO(IN), set when the Word Count Register counts to zero and the current transmission is complete, set also by INIT and NEX (Error Register bit 07).

08 GO(IN) Read/write. When set the DMA controller requests Unibus mastership and transfers data using DATO cycles to memory from the interface. Resets READY (bit 07). Reset by INIT, by GO(OUT) (bit 00), and by READY (bit 07) being set.

09 STTXRDY Read only. When set the serialiser is ready for output, the Local Status Register may be loaded with the status to be transmitted. Set by INIT and by the end of the serialisation process, cleared at the leading edge of the serial start bit.

10 Not Used

11 Not Used

12 Not Used

13 ERROR Read/write. This bit is the logical OR of the error conditions signalled in the Error Register. Reset by INIT and by resetting the error condition(s) in question.

14 DIE Read/write. When set enables the DONE (bit 15) interrupt. Reset by INIT.

15 DONE Status input done - read only. When set indicates that a software status byte has arrived from the remote end and may be read in the Remote Status Register. An interrupt will be made, if enabled (DIE - bit 14). Reset by INIT, by remote and local Clear Link and by writing into the Remote Status Register.
F.2 ERROR REGISTER

This byte register contains all the interface error flags. All bits are read only.

Unibus Address: N+2

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>OFFLINE</td>
<td>When set the link is off-line, no data or status can be transmitted or received. This applies only to those links equipped with the optional control panel. Those without panels are always on-line.</td>
</tr>
<tr>
<td>01</td>
<td>NREMCLK</td>
<td>No Remote Clock. When set this bit indicates that no serial transmission clock is being received from the remote end (modulated systems only), the transmission line is broken or the remote end is switched off (all systems).</td>
</tr>
<tr>
<td>02</td>
<td>CNTE</td>
<td>Count Error. An odd number of data bytes has been received from the remote end. Reset by INIT, remote or local Clear Link or Clear Buffer.</td>
</tr>
<tr>
<td>03</td>
<td>STORVN</td>
<td>Status Overrun. Three or more status bytes have been received. Reset by INIT, remote or local Clear Link or Clear Buffer.</td>
</tr>
<tr>
<td>04</td>
<td>DOVRN</td>
<td>Data Overrun. A data byte has arrived from the remote end with the input FIFO buffer full. Reset by INIT, remote or local Clear Link or Clear Buffer, by reading the Unibus address N+2.</td>
</tr>
<tr>
<td>05</td>
<td>PE</td>
<td>Parity Error. A parity error has been detected in the serial data stream coming from the remote end. Reset by INIT, remote or local Clear Link or by reading the Unibus address N+2.</td>
</tr>
<tr>
<td>06</td>
<td>BNE</td>
<td>Buffer Not Empty. One or more bytes of data are detected in the input FIFO buffer at DMA word count zero. Reset by INIT, remote or local Clear Link or Clear Buffer.</td>
</tr>
<tr>
<td>07</td>
<td>NEX</td>
<td>Non-existent memory. The Unibus transfer control &quot;hand-shake&quot; has broken down (no SSYN response to the master within 20 microseconds). Sets READY (Command Register bit 07). When set causes Unibus mastership to be relinquished. Reset by INIT and by reading the Unibus address N+2.</td>
</tr>
</tbody>
</table>
F.3  CONFIGURATION REGISTER

This one word register gives details of the data link hardware configuration, it is read only, read at the Unibus address N+2 subsequent to the hardware status "Read Link Configuration".

Unibus Address: N+2

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>07-00</td>
<td>Serial Number</td>
<td>This is a unique serial number for every interface. It is read as two digits of 1248 binary coded decimal.</td>
</tr>
<tr>
<td>09-08</td>
<td>Link type</td>
<td>This defines the actual link hardware standard. The two bits are encoded thus: 0 - &quot;mark 2&quot; Omnet link (now obsolete), 1 - &quot;mark 4&quot; Omnet link (current version).</td>
</tr>
<tr>
<td>11-10</td>
<td>Not used</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Bootstrap posn.</td>
<td>This bit signals the Unibus address of the Hardware bootstrap ROM. 0 - 773700, 1 - 770700.</td>
</tr>
<tr>
<td>13</td>
<td>Not Used</td>
<td></td>
</tr>
<tr>
<td>15-14</td>
<td>Link Speed</td>
<td>Defines the transmission speed of the link (set by manual switches). These bits are encoded thus: 0 - &lt; 300 Kbaud, 1 - &lt; 3 Mbaud, 2 - &gt; 3 Mbaud.</td>
</tr>
</tbody>
</table>

F.4  REMOTE STATUS REGISTER

Unibus Address: N+2

This byte register holds software status bytes received from the remote end, bits 00-06 only (bit 07 of the status having been used to distinguish hardware and software status). The Remote Status Register is read only, although it may be cleared by writing any value into it.

F.5  LOCAL STATUS REGISTER

Unibus Address: N+3

Loading this write-only, byte register will cause the value loaded to be sent over the serial link as a byte of status - exception: the Read Link Configuration hardware status.
F.6 MEMORY ADDRESS REGISTER

Unibus Address: N+4

This register, together with bits 03 and 04 of the Command Register, forms an 18 bit counter which is used to point to the Unibus address to or from which the current DMA transfer is to be made. It increments by two after every DMA cycle. Bits 01-15 are read/write, bit 00 is not used. The register is cleared by INIT.

F.7 WORD COUNT REGISTER

Unibus Address: N+6

This register is a 16 bit counter used to count the number of DMA cycles made. It increments by one after every DMA cycle. All bits are read/write. The register is cleared by INIT.
BIBLIOGRAPHY


Brobecker.S The Cernet Transmission System (Internal Note)


Davies.H Distributed Computing in a High Energy Physics Laboratory (Internal Note)


Digital Equipment Corporation RSX-11M Operating System Manuals (various)

Digital Equipment Corporation PDP11 Processor Manuals (various)

Digital Equipment Corporation Unibus Interface Manual

Downie.M CERN - Network Project Note 79: The PDP11 Bounce/Echo Program


Gerard.J CERN - Network General Note NGN/79/13: Cernet Data Rates


Nerdel.J Omnet/76/2: RSX-11M Link Driver for the Omega/Muon Network

Parkman.C Omnet Operations Guide
Parkman.C CERN - Omega Development Note HW-15: PDP11 Address Coincidence Detector

Parkman.C CERN - Omega Development Note HW-21: PDP11 Data Link Interface (Mark 4)

Parkman.C CERN DD/ES/OS Technical Note TN29: PDP Omnet Data Link Interface (Mark 4a)

Parkman.C CERN - Network Project Note NPN34: PDP11 Cernet Data Link Interface

Parkman.C CERN - Omega Development Note HW-16: EMR 6130 Data Link Interface

Piney.C CERN - Network Project Note NPN30: The Line Protocol Implementation

Piney.C The Omnet/Cernet Gateway Software (Internal Note)

Pordes.R LKMON - Omnet link monitoring task: (Internal Note)

Russell.R CERN 72-21 (part) The Omega Project: A case history in the design and implementation of an on-line data acquisition system


Standley.M The Omega Computer Network (Internal Note)

Standley.M The Small Machine Organiser (Internal Note)


Zurbuchen.R, Beard.M The Omnet Co-axial Cable Transmission System (Internal Note)
INDEX

acknowledgement blocks ... 4
Asynchronous ... 17, 33
boot-host ... 30
BOOTFILE? ... 31
bootstrap
  hardware ... 30
  remote ... 12
  request ... 4
  software ... 30
Bootstrapping ... 30
Bounce ... 43

Cable
  co-axial ... 22
  telephone ... 23
  twisted quad ... 22
Cernet ... 2
  line protocol ... 33, 35
Checklink ... 42
CII 10070 ... 1, 18

cler
  buffer ... 12
  link ... 12
clusters
  local ... 2
codes
  acknowledgement ... 10-11
    initiate ... 9
communications
  subnet ... 35
conflict case ... 4-5
CRC ... 34
Cyclic redundancy check ... 34
data ... 18
DATI ... 19
DATO ... 19

Debug and trace facilities ... 27
deserialiser ... 20
device block ... 26
Direct Memory Access ... 4
DMA ... 4, 17
DONE ... 21
duplex
  full ... 4, 17

emitter-coupled logic ... 22
EMR 6130 ... 1, 18
error
  irrecoverable ... 11
  rate ... 41
  recoverable ... 11

failure rate ... 41
FIFO ... 17, 20, 34
File Manager ... 30
first-in, first-out ... 17, 34
flow control ... 17
  Omnet/Cernet ... 35
gateway ... 2
  hardware ... 33
  Omnet/Cernet ... 33
  software ... 34

hardware
  common components ... 18
  data link interface ... 17
Hardware Bootstrap ROM ... 19
hold ... 17-18, 20

IBM
  3032 ... 25
  370 ... 2
interfaces ... 18
interrupt
  handlers ... 34
IT ... 5
  codes ... 10-11
ITA ... 5
  codes ... 9

KEYTASK ... 25

later ... 10, 36
  wake-up ... 9, 36
lateral parity ... 17
latering ... 4
line transmission ... 21
lines
voice grade ... 23
linktask ... 27
LKMON ... 44
LOCORE ... 25
LSII1 ... 18

machine
deadstart ... 4
number ... 5
MACRO-11 ... 29
memory management ... 24
mode
safe ... 4
unsafe ... 4
modem ... 21
modulation
phase-code ... 22
Multi-tasking ... 25

nodes
configuration ... 23
switching ... 23
not-hold ... 17-18, 20

Omega ... 1
Omnet
coverage ... 2
history ... 1
nodes ... 23
number ... 5
operations ... 44
topology ... 37-38
optical coupling ... 22

packet switching ... 2
parameter ... 4
block ... 4
block contents ... 6
parity
lateral ... 17
parity bit ... 19
PDP11 ... 18
Data link interface ... 18
Performance
measurements ... 39
phase
initiation ... 4-5
second ... 4, 6
third ... 4
PL-11 ... 24
Protocol
End-to-end ... 33
Transmission ... 4
re-entrant tasks ... 26
read link configuration ... 12
Receive
master ... 35
Register
Command ... 19
Configuration ... 19
Data Input ... 20
Data Output ... 19
Error ... 20
Local Status ... 21
Memory Address ... 19
Remote Status ... 21
Word Count ... 19
RSX-11 ... 29
RSX-11M ... 29
RSX-11S
loadable tasks ... 32
serial number ... 19
serialiser ... 19
Small Machine Organiser ... 2
SMO ... 2, 24
Design Criteria ... 24
input/output ... 26
linktask ... 27
loadable tasks ... 32
Split Field Magnet ... 1
status ... 4, 18
bit usage ... 8
byte contents ... 8
hardware ... 8, 21
hardware codes ... 12
software ... 9, 21
stop ... 11
Stop bit ... 18
stop codes ... 11
synchronous ... 17
SYSGEN ... 25
Test facilities ... 42
Testlink ... 42
Transfer
termination ... 11
transformer coupling ... 22
transistor-transistor
logic ... 21
Transmission Protocol ... 4
Transport Manager ... 33
Unibus ... 18
Mastership ... 21
up-counter ... 19