CERN 76-08
Data Handling Division
4 May 1976

ORGANISATION EUROPÉENNE POUR LA RECHERCHE NUCLÉAIRE
CERN EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

A SYSTEM FOR COMMUNICATION BETWEEN
A CDC 6000 AND A PDP 11 COMPUTER


GENEVA
1976
© Copyright CERN, Genève, 1976

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.

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.
A SYSTEM FOR COMMUNICATION BETWEEN
A CDC 6000 AND A PDP 11 COMPUTER


GENEVA
1976
ABSTRACT

A description is given of the hardware and software employed for the interchange of binary data between a PDP 11 and a CDC 6000 computer, for the bibliographic-data acquisition system of the CERN Library. The data is transmitted as buffers of 8-bit bytes with sequence and checksum verification. Initially tested under standard Intercom, the software was subsequently updated to communicate with CERN's programmable front-end concentrator Supermax. For this, a special interface was designed, providing full-duplex transmission and capable of distinguishing between the pure binary computer data and terminal-oriented system messages. The various stages in the development are briefly described and a more detailed account is given of the current system, particularly the discrimination between data and system messages and the use of DMA channels. A complete set of circuit diagrams is included.
CONTENTS

INTRODUCTION  1

1. DESCRIPTION OF THE SOFTWARE  1
   1.1 Operation  1
   1.2 PDP 11 line-driver to Intercom  2
   1.3 The special interface  3
   1.4 The link-driver
      1.4.1 Buffer format  5
      1.4.2 Protocol  6
   1.5 User experience  8

2. DESCRIPTION OF THE HARDWARE.  8
   2.1 Design features  8
   2.2 Transmitter  8
   2.3 Receiver  10
   2.4 Data separation  10
   2.5 The DMA channels  11
   2.6 Implementation  11

REFERENCES  12

APPENDIX: SCHEMATIC DIAGRAMS  13
INTRODUCTION

A system has been implemented for computer-aided cataloguing and related operations of the Scientific Information Service at CERN\textsuperscript{1).} Basic operations such as input and modifying of bibliographic records are carried out interactively on a PDP 11. Permanent storage and large data processing facilities are provided by a CDC 6000 computer to which the PDP 11 is connected.

Transfers can proceed in parallel with the interactive operations of the PDP 11, and can be initiated either automatically or under operator control. The data-transfer operations are structured so as to allow the error-checking protocol to be independent of the basic character transfer logic.

One of the main problems in communicating with the CDC 6000 system is that 'system messages' can be interspersed amongst data buffers destined for the remote computer. To discriminate between these two types of input, a special interface has been constructed; this provides full duplex transmission over two DMA channels into the PDP 11.

All unmarked data is stored via the 'system message' DMA channel. On receipt of a preset character, the receiver terminates the system message and stores subsequent characters via the 'data message' channel. All transfers can terminate on byte-count overflow or time-out.

1. DESCRIPTION OF THE SOFTWARE

1.1 Operation

In order to transfer data between the PDP and CDC machines, there are basically two problems to be solved:

i) To transfer one character: this depends on
   a) hardware: the input/output and interrupt architecture of the machines, and
   b) the operating system: the data transfer formalities for device drivers.

   The routine that performs this function is termed "line-driver".

ii) To transfer one buffer automatically, verifying content and sequencing: this depends only on the chosen data-transfer protocol and the software interface to the line-driver.

   The routine that performs this function is termed "link-driver".

An outline of the software system and the role of link- and line-drivers are given in Fig. 1: the numbers in circles in the following paragraphs refer to numbered elements in this figure.
On the CDC 6000 side, the link-driver ① is part of a FORTRAN program ② that runs under the CDC interactive system, Intercom ③. This link-driver exchanges buffers with Supermux ④, the CERN front-end concentrator system ⑤. Supermux serves as line-driver on the CDC 6000 side.

Within the PDP, several interactive programs ⑤ ⑥ can insert requests into the link-queue, by calling link-Q handler ⑦. This handler controls calls to the link-driver ⑧, which in turn calls the line-driver ⑨. There is a single-user program ⑩ that communicates directly with the line-driver, thereby allowing character-based, teletype-like communication between an interactive user and Intercom.

The line-drivers control the operation of the input/output interfaces ⑪ ⑫.

Within the PDP 11, the programs and link-queue handler run under a modified version of the PDP 11 operating system, DOS ③; this system allows time-sharing between several re-entrant or single-user tasks, under a multi-level software priority scheme.

1.2 PDP 11 line-driver to Intercom

At the time when the programming for the connection was started, Supermux was not operational. It was therefore decided to simulate a binary connection via Intercom 3.4.1.

This version of Intercom did not support 8-bit transfers but only allowed for the transfer of 6-bit BCD characters. Thus it was necessary to split each 8-bit byte into two 4-bit bytes, which were then mapped onto the 16 BCD characters B-Q. In addition, the line-speed was restricted to 110 baud. Transfers were therefore extremely slow since an effective rate of only five PDP 11 characters per second can be achieved under these conditions.

In addition to these problems, the following difficulty was discovered: Intercom is basically a half duplex system. In fact, any character received as input while an output is underway is treated as an "escape". The effect of "escape" is to interrupt output from Intercom; the next character received is then interpreted as follows:
- carriage return = resume transfer
- A = abort current program
- S = abort current output buffer

Other characters are ignored.

Since it takes 1/10 second for a character, output by Intercom, to be detected at the other end of the line, there was no way of preventing the PDP 11 from sending a character and thereby interrupting an Intercom output. It is not always possible to predict when Intercom will start outputting, since systems messages from Intercom may be sent at any time when Intercom is not actually receiving a buffer.

These characteristics meant that the PDP 11 line-driver had to identify an input/output clash and then time-out to see if output from the CDC 6000 to the PDP 11 had been suspended by Intercom. If suspension had occurred, the PDP 11 driver had to send a carriage return to Intercom in order to cancel the suspension. Once the message from the CDC 6000 terminated, the PDP 11 had then to retransmit the whole buffer that caused the clash.

The driver also had to differentiate between data buffers sent by program and system messages initiated within Intercom. At any time when Intercom is running, messages may be sent to connected terminals; these "system messages" have no unique format to identify them. The solution was therefore to impose a unique format on the buffers sent by the CDC 6000 line-driver ("data messages"). Since Intercom follows the line-printer convention of interpreting the first character of each output buffer as a carriage control character, a characteristic carriage control sequence can be used to tag data messages, if this is different from that used by Intercom for its system messages. Even this solution is not fool-proof since, in certain circumstances, Intercom modifies the chosen carriage control sequence. This of course results in lost buffers; the PDP 11 driver also had to time-out to identify this state.

By use of these and other ad hoc solutions to undocumented problems, a usable driver was constructed and was as reliable as the system with which it communicated. All the same, it was unwieldy and slow. Communicating in this way with Intercom brings to mind Dr. Samuel Johnson's comment about a woman's preaching: it "is like a dog's walking on its hinder legs. It is not done well; but you are surprised to find it done at all".

1.3 The special interface

The restrictions imposed by standard Intercom prompted CERN to develop a special concentrator, called Supermux\textsuperscript{7}. It is designed to give added flexibility to communicate with Intercom. It allows

- full duplex transmission
- 8-bit character transfers
- speeds of up to 9600 baud.

In order to solve the problems encountered in communicating directly with Intercom, and to take full advantage of the Supermux facilities, a special hardware interface was designed. Complete hardware details of the special interface are given in Section 2, and only the main characteristics will be defined in this section. Numbers in brackets refer to blocks in Fig. 2.
- For compatibility with Supermux, the transfer speeds of the interface are software selectable (1 & 2).

- The speeds are stored in the relevant status registers (3 & 4).

- For simplicity, output is on a character basis (5).

- The receiver operates on a DMA basis and resolves the data message/system message uncertainties encountered in the previous section.

Two separate DMA inputs can be enabled; one for system messages (9), the other for data messages (8). The start of a data message is identified by the receipt of a software selectable "start of data message" character; this character is stored in a register in the keyword comparator (6).

A data message terminates on one of two conditions:

- time-out: when a delay of more than 1/4 second has elapsed since the previous character;

- buffer full: when the character count preset in the byte counter (7) is reached.

A system message will terminate on any of three conditions:

- start of data message: the "start of data message" character causes termination of a system message; all subsequent characters are stored in the data message buffer until this data message terminates;

- time-out;

- byte count overflow: when 256 characters have been received.

The message is read into the buffer whose address is stored in the relevant DMA register (8 or 9); on end-of-message, the relevant interrupt is generated (10 or 11).

Loading and examination of the interface registers from the unibus is done via (12) in exactly the same way as for normal PDP 11 device registers.
This special interface, in conjunction with the added facilities of Supermux, greatly simplified the form of the line-driver for PDP 11/CDC 6000 communication. Thanks to the subdivision of data transfer functions into line- and link-drivers, the introduction of the new line-driver did not entail any changes to the existing link-driver, which had been extensively tested under standard Intercom.

1.4 The link-driver

1.4.1 Buffer format

The link-driver incorporates all the data transfer protocol. It must ensure the correctness both of the buffer contents and of buffer sequencing. In any transfer there is a sender and a receiver; buffers from the sender are called "messages" and those from the receiver are termed "responses". After sending a message, the sender waits for a response; this carries the status of the preceding transfer.

There are four valid responses:

- message successfully received (Fig. 3b)
- transfer successfully terminated (Fig. 3c)
- checksum error detected (Fig. 3d)
- sequence error detected (Fig. 3d).

\[ \begin{array}{c}
000 \\
1 \text{A} c \\
1 \text{T} c \\
1 \text{C} c \\
1 \text{S} c
\end{array} \quad \begin{array}{c}
\text{null} 3a \\
\text{accept} 3b \\
\text{terminate} 3c \\
\text{checksum} \\
\text{sequence} \quad \text{errors} 3c
\end{array} \]

Fig. 3 Special buffers

In order to allow this form of error detection, all transfer buffers have the same form (Fig. 4). The first byte of every buffer contains the buffer number (modulo 255, plus 1); this is followed by one byte containing a count of the number of data characters in the buffer, and then the data characters. The last character in the buffer contains a cyclic checksum built up from the preceding characters. Figure 3 shows the form of five special buffers: the degenerate (null) message and the four responses described above. The way in which these are used defines the data transfer protocol.

\[ \text{Fig. 4 General form of a transfer buffer} \]
1.4.2 Protocol

Every message is followed by a response which determines the next action to be taken. There are three rules linking messages and responses, these are shown in Fig. 5:

The '1E' (5a); the 'E' rule (5b); and the '1iE' (5c).

These are used to block buffer-transfers into logical records; the '1E' and the '1iE' are used only on the receiver side, to initiate ("open i") and terminate ("close i") the transfer of a logical record. All messages and responses, apart from the first and last on the receiver side, are controlled by overlapping 'E's.

The 'E' rule states that every output is followed by an input.

a) Output E

The 'E' rule on the output side therefore requires that a message be followed by a response. This response determines the next action to be performed, as follows:

i) Response = "accepted" (Fig. 3b)
Action = The next output-'E' is performed

ii) Response = "checksum error" (Fig. 3d) or,
response incomprehensible because of checksum or sequence error
Action = The last output-'E' is sent again

iii) Response = "sequence error" (Fig. 3d); the sequence number of the required buffer is used as sequence number for this response
Action = The output-'E' for the correct buffer in the sequence is performed.

---

Fig. 5 Protocol rules
b) **Input 'E'**

i) The 'E' rule on the input side means that, before input can be expected, the response to the previous message must be sent. Once input has been received, the status (accept, checksum/sequence error) of that input is stored for use by the following input 'E'.

ii) The '1[\|]E' rule is only applicable on the receiver-side and represents the second part of the input-"E": accept input and set status for the following input-"E". This is used to initiate input of a logical record.

iii) The '1[\|]E' completes the receipt of a logical record; it is the input-"E' rule followed by a final response to indicate "transfer successfully terminated" (Fig. 3c).

c) **Null message**

The null message (Fig. 3a) is used by the transfer side for status information in three cases:

i) to start a logical record: the '1[\|]E' always expects to receive the null message; any other message will be treated as a sequence error;

ii) to finish a logical record: the '1[\|]E' always expects that the final message will be the "null" message;

iii) to abort a logical record: the input-"E' never expects the null message; the '1[\|]E' rule will be applied if this is encountered, but with the final response: "accepted", with sequence number zero.

d) **Buffer transmission**

By use of the 'E' rules, the transfer of logical records can be synchronized between sender and receiver; since the sender's 'E's overlap those of the receiver, output and input are safely interlocked. To abort a transfer -- perhaps because of too many errors -- only a "null" message needs to be sent: the whole transfer of the logical record can then be restarted if necessary.

If the sender and receiver do not both expect the same number of buffers in the logical record, one of two cases can arise:

i) The sender tries to send too few buffers. The null message, sent after the last output buffer to terminate the logical record, will be interpreted by the receiver as an "abort". The "accept" instead of the expected "terminate" response will inform the sender that the transfer did not terminate successfully.

ii) The sender tries to send too many buffers. When the '1[\|]E' receives a non-null message, it will respond with the "sequence error" response, requesting buffer number zero. The sender must then enter its record-abort procedure.

This simple protocol ensures the integrity of transmitted data, and allows recovery from most of the common errors that can occur during transmission.
1.5 User experience

By the end of 1975 the link-driver had been in use for over a year, and the current line-driver for six months. During this time, over 10,000 bibliographic records have been successfully interchanged between the two machines. The error checking and retransmission procedure have ensured the complete integrity of the transmitted data, but the support programs have not always proved sufficiently flexible to allow for repeatedly corrupted buffers, and transmission has then had to be abandoned. This situation has only arisen when there were genuine hardware errors on the CDC 6000 side of the line; this has not adversely affected the over-all efficiency of the system, which can build up queues and store data locally until the connection is usable again.

2. DESCRIPTION OF THE HARDWARE

2.1 Design features

The receiver-transmitter circuitry is based on the "asynchronous full duplex interface" used by Supermax². The crystal clock and programmable clock dividers that control the line speeds are copied directly, whereas the actual receiver-transmitter circuits are replaced by the more modern TMS 6011 integrated circuit, which also uses a clock frequency of 16 times the transmitted or received baud rate.

All other circuits for input-output, as well as status and command, have been redesigned and adapted for interfacing to the PDP 11 unibus³.

To keep the load on the unibus as low as possible, a buffer is used where the data lines are split up to form internal input and output buses. These buses are also connected to the outside of the card, so that the other circuits of the data link can connect to them.

Of the 18 address lines, 14 are coupled to an M 105 device selector where a group of 16 addresses is selected. The remaining 4 lines go to a set of bus receiver circuits for distribution to the different final address-decoders. The result is that connection of the data link represents less than 1 standard load on the unibus.

2.2 Transmitter

In common with other MOS circuits, the TMS 6011 is a relatively slow circuit, compared to TTL logic. This means that the unibus signals are much too fast to be accepted directly by the transmitter, necessitating an intermediate buffer register. This is connected in such a way that it forms a 3-byte push-down memory with the two internal registers of the TMS 6011.

The control is done by two flip-flops, acting on the Read-In and the TRBE (Transmitter Buffer Empty) signals. They are synchronized by the transmitter clock (see Fig. 6). For each byte sent out, the two flip-flops are set and reset. Besides controlling and synchronizing, their signals are used both as "Done Flag" and to send an interrupt to the computer for each byte sent out.
Fig. 6 Special interface - Block diagram

The TMS 6011 transmitter converts the parallel data to serial CCITT format. A group of different line drivers coupled to the output gives a choice between TTL, CCITT, or current mode output levels.

The differential current mode output is used for a direct coupling to the Supermux system. The parameters of the transmitted word are defined by the status and command register, a read-write register controlled by the final address decoder. It allows for the selection of the transmission speed, number of stop bits, and parity. The format is the same for both transmitter and receiver (Fig. 7).

<p>|
| RECEPTOR: STATUS &amp; COMMAND REGISTER |</p>
<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARITY</td>
<td>3</td>
<td>RECEIVER</td>
<td>1</td>
<td>SPEED</td>
<td>0</td>
<td>DONE</td>
<td>INT.</td>
<td>EN</td>
<td>TIME</td>
<td>OUT</td>
<td>OVER</td>
<td>FLOW</td>
<td>CHAR</td>
<td>LOST</td>
<td>TEST</td>
</tr>
</tbody>
</table>

Fig. 7 Transmitter-Receiver formats
Four bits are used for speed selection, and one bit is used to select an internal connection to the receiver (test loop). The number of bits/word and parity polarity can be selected by jumpers. The Interrupt Enable and Done bits conform with the PDP 11 protocol.

2.3 Receiver

The receiver parameters, except for speed and test loop, are selected in common with the transmitter. These two selections are made by the receiver status and command register. Speed selection is also done by programming the receiver clock. The test loop bit enables an input to be connected to the transmitter, so that the test loop will only be established if both receiver and transmitter have the appropriate bit selected.

When a character is received it becomes available on the output pins of the TMS 6011, which does the serial to parallel conversion, at the moment that half of the first stop bit is passed. One clock pulse later, the Data Ready (DR) flag is set. In this way even if the fastest speed possible is selected, there is still a minimum period of

$$\frac{1}{2} \times \frac{10^6}{\text{baud rate}} = \frac{10^6}{192 \times 10^2} = 52 \ \mu\text{sec}$$

available before reception of the next byte starts. With two stop bits this separation time is extended to 156 \( \mu\text{sec} \).

This shows that there is, with one stop bit only, already plenty of time to do some processing for the data separation and the DMA transfer, without using the 1-byte buffer storage possibility of the TMS 6011, and without influencing the reception speed.

The received byte of data is brought out of the receiver transmitter card in two different ways: once directly, which is to say buffered but without gating, and secondly by a double set of unibus transmitter gates. Each set of gates has a double enable entry to allow connection to the unibus with direct packing of two 8-bit bytes into one 16-bit computer word. Upper or lower bytes are selected by bit 00 of the selected address counter.

2.4 Data separation

The data from the direct output of the receiver goes into the "keyword comparator" where it is compared with the stored keyword. The keyword comparator is simply a 1-byte presetable register coupled to an 8-bit data comparator. The other eight inputs of the comparator can be coupled to the inputs of this card directly by a set of jumpers or to an 8-bit counter, if required.

When a match is found, the "HIT" flip-flop will be set on the next clock pulse. For keyword comparison the DARE signal is used; this is the Data Ready signal converted to TTL level. In this way the HIT flip-flop can never be set at the moment of data transfer, but is always set before the DMA transfer. Thus the keyword is always the first byte set in the data buffer.

On a match, the HIT flip-flop will generate the terminal system message interrupt. The corresponding vector-address is assured by a second flip-flop which memorizes the previous state of the HIT flip-flop (see also Fig. 8). Only after the interrupt is finished does the state of the second flip-flop follow that of the HIT flip-flop. This configuration makes fast switching possible, even within a stream of input data.
2.5 The DMA channels

The DMA transfer logic is composed of two separate address-counters and one shared byte counter. Standard circuit modules are used as far as possible. However, the requirement of having two channels has made it necessary to develop special address and byte counter cards. For the byte counter the same circuit is used as for the data comparator. The HIT flip-flop is used to select whether the counting goes up to a specified number of bytes or up to overflow.

The address counters are two bus-oriented presetable 16-bit counters. An internal bus structure makes it possible to select one of the counters to be coupled to the output, in this case the address lines of the unibus. Of course, the HIT flip-flop again defines which of the two counters is selected to the output pins, and at the same time enables the corresponding inputs. The same synchronization logic is used by both DMA channels.

2.6 Implementation

The data link occupies one BB 11 module in the PDP 11, in which the programmable data link, the data separator, the two-channel DMA, and the interface are contained. The programmable data link is constructed on a quadruple connector double-height printed circuit board which can also be used independently. The other parts of the logic are mounted on standard modules.

No special reliability tests, as such, have been undertaken; however, in the first six months of use over three million characters have been sent and received without detection of transformed or missing data due to this hardware. Neither has there been any detectable difference in the performance of the Supermax system due to the coupling of a PDP 11 using the described data link.
This interfacing technique could be of great use in a multidrop communication system where one computer sends messages to several connected computers.

These computers can all be connected by the same full-duplex line. The first character of each message could then be used to identify the recipient.

This method simplifies multiplexing and allows flexible reconfiguration of such a system.

REFERENCES


SCHEMATIC DIAGRAMS

Diagram 102 - Programmable clocks
" 103 - Transmitter
" 104 - Data interface to the unibus
" 105 - Receiver
" 106 - Data comparator or byte counter
" 107 - Selectable address counters
" 108 - Separation synchronization
" 110 - BB 11 module
" 111 - Address selection and data distribution
" 112 - Interrupts and DMA channels
Data comparator or byte counter (106)
<table>
<thead>
<tr>
<th></th>
<th>UNIBUS</th>
<th>SUPERMUX RECEIVER TRANSMITTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td>DD 4476</td>
</tr>
<tr>
<td>3</td>
<td>POWER</td>
<td>DATA COMPARATOR DD 4479</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WORD COUNTER COMPARATOR DD 4479</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMMUNICATION TIME OUT LOGIC DD 3930</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MASTER CONTROL M 786</td>
</tr>
<tr>
<td>2</td>
<td>ADDRESS COUNTERS DD 4482</td>
<td>INTERRUPTS M 782</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BUFFER A FULL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DMA REQUEST M 782</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DMA INTERRUPT</td>
</tr>
<tr>
<td>1</td>
<td>UNIBUS</td>
<td>SEPARATION SYNCHRONISER DD 4524</td>
</tr>
<tr>
<td></td>
<td></td>
<td>M 105</td>
</tr>
</tbody>
</table>

**BB 11 module (110)**