USE OF 5053 DMP CONTROLLER FOR
A FAST PACKET SWITCHING NETWORK

Presented at the European Modcomp Users (MUSE) annual meeting,
Divonne-les-Bains, 1977
1. **INTRODUCTION**

At CERN a fast packet switching network is being implemented. For this eight Modcomp II machines have been acquired which function as nodes and front-ends to the host machines in the Computer Centre. Interconnection of the nodes and the experimental computers is done with CERN-made data links.

Up to 15 full duplex data links can be connected to a 5053 DMA controller. Each data link has the capability of storing a complete packet (1k word, 16 bits) in a buffer memory at the receiving end.

Asynchronous serial transmission of data and control information is done at a speed of 2.5M bit/s. Error protection is performed with a parity bit and a cyclic redundancy check word.

Given the large geographic distribution of the nodes, a REMOTE FILL over a data link is foreseen: a specially coded control word sent down a link will cause the remote node to fill itself from a read only memory which is also connected to the 5053. The program thus loaded is used to set up a dialogue with the computer which sent the command. A complete core load is subsequently sent down the link to the remote computer.

2. **CONNECTION SCHEME**

The 5053 is similar to an XDMP, but allows for 32 device addresses as well as 32 TA's and TC's. The B bus of the 5053 is connected to a BUS BUFFER which serves as an electrical separation between the 5053 and the local bus in a crate containing up to 30 modules (see Fig. 1). Each module has its own TA, TC, SI, DI and device address.

A full duplex link consists of a sender and a receiver module on each computer (see Fig. 2). This allows us to have up to 15 links in one crate.

Priority arbitration via the data bus is limited to 16 devices and therefore an independent priority circuit has been implemented to cater for 30 devices (see Fig. 1).

The A bus of the 5053 is uniquely used to connect to the FILL ROM to be described in Section 5.
3. **DESCRIPTION OF A DATA LINK**

Figure 2 shows a block diagram of a full duplex data link between two computers. It should allow for control and data flow in both directions. Furthermore, commands can be issued to it and status can be read.

3.1 **Identification of control and data words**

The problem is to send two types of 16-bit words over one serial link: a control word and a data word. The solution to this problem can be seen in Fig. 3. The differentiation between these two words is made by the DATA/CONTROL FLAG bit (DCF), immediately after the opening bit of the serial stream (serial transmission unit).

3.2 **Control and data flow from interface to interface**

The sender has a control output fifo (COF) and a data output fifo (DOF) both protected with odd parity (see Fig. 4). The outputs of these fifos are treated by the MIXER to form a serial bit stream of TRANSMISSION UNITS. A control word has priority over a data word and can as such be inserted between two serial transmission units of data words during transmission of a packet.

In order to conserve odd parity throughout the link, the parity bit of a data word is inverted before entering the mixer (DFC=TRUE), giving a protection of the DFC bit as well.

The mixer has an output for testing with the local receiver as well as a balanced output that can be used over a distance of about 10 metres. If a longer distance is required, this output goes to a DATA SET.

The DATA SET converts the incoming signal into a signal that allows for transmission over 1-2 km of twisted pair cable at 2.5M bit/s. The error rate is better than 1 in $10^{10}$ bits. At longer distances a REPEATER is put into the system.

The DATA SET at the receiver reshapes the incoming signal and sends it to the separator via the short distance balanced system. There is also an input that connects to the local sender for test purposes.

The separator deserializes the TRANSMISSION UNITS and decides if a control word or a data word has been received. It sends the word and its (inverted) parity to the related input fifo. When data and control words come out of their respective fifo's, parity is checked.
3.3 **Error protection**

As indicated in the previous paragraph, data and control words are, with their DCF bit protected by a parity bit. The parity bit serves to find simple and single errors in the fifo's and the transmission system. However, to give a more powerful protection of a data packet, a cyclic redundancy check word (CRC) is generated by the sender and sent after the last word of each packet. At the receiver this word is used for error checking on the data packet.

4. **IMPLEMENTATION ON MODCOMP**

Figure 5 shows a functional block diagram of a sender and a receiver as implemented on the Modomp (see also Table 1 for abbreviations).

4.1 **Control path**

The COF and CIF are four words deep and allow for storing the maximum numbers of control words that can reasonably be output in a bunch. The control words entering the CIF are presented one by one to the CPU by means of an SI. The control word I/O is performed by the ODX and IDX instructions.

A specially coded control word represents the packet word count which is loaded on the fly in the SWCR and the RWCR. These are counters which decrement at every data word that passes and when reaching zero, cause the CRC word to be sent and checked, respectively. Reading and writing of these counters is possible with IDX and ODX instructions after selecting them with a select command (OCX to SCR or RCR).

4.2 **Data path**

DMA output is into a four word DOF that serves as a buffer between the I/O bus and the transmission system. At the receiving end there is a 1k word buffer called DIF that can contain a complete packet. Input from this buffer to the computer is done via DMA. A feature of the DIF is that the read pointer can be reset so that re-reading of the packet is possible.
4.3 **Interrupts**

There are several interrupt-causing status bits which generate an SI, on condition that they are not masked. Masking and unmasking is performed with SIAR and RIAR, which are accessed with an OCX command. Selective clearing of these status bits by an OCX command guarantees unambiguous interrupt treatment.

4.4 **Timers**

Every module has a timer (STOC and RTOC) that starts running after being loaded with a value. At time-out it generates an interrupt. A timer is loaded and read with an ODX and IDX instruction, respectively, after being selected with an OCX command.

5. **REMOTE FILL**

A specially coded control word, when decoded in the receiver produces a FILL pulse. The fill pulses from every receiver are input to the fill hardware connected to the A bus of the 5053 (see Fig. 1).

The fill hardware works as follows (see Fig. 6): a fill pulse, if not masked, sets its corresponding latch. The highest priority latch being set gets encoded and produces a four-bit number that can be interpreted as the group and the three most significant bits of the device address of the corresponding receiver. The fill pulse causes the fill logic to go through the standard fill sequence forcing the CPU to start reading bytes of data from a read only memory called P.ROM, whose address was picked up during the fill operation. After the P.ROM has been read into core, the CPU uses the code to set up a transfer over the link that caused the fill.

As this can be any of the 15 links in a system, the code from the P.ROM has to be adapted to contain the right TA and TC pointers, group and device addresses. This is done by reserving special values (Hex. 40-47) in the P.ROM that are replaced in real time by the appropriate values for the filling link.

Figure 7 gives the table for this conversion and Fig. 6 shows how the hardware is implemented to achieve this: the translation ROM (T.ROM) produces the code for the I/O instructions in group C or D as well as the
relative TA and TC pointers. The function multiplexer allows for
insertion of the device addresses of the particular sender and receiver
(these addresses are consecutive).

It is possible to fill from the console, but then only one default
hardwired link address can be used.

6. ACKNOWLEDGEMENTS

The design and testing of the hardware has been very skillfully done
by R. Pieters and S. Brobecker. Furthermore, I would like to compliment
our electronic workshop for the quality of the wirewrap service they pro-
vided for our prototypes. Last, but not least, we are indebted to
D. Wiskott who implemented a set of programs which made the writing of
wiring lists for the semi-automatic wirewrap machine, if not a pleasure,
at least an acceptable job.

* * *

REFERENCES

   Publication No. 210-102000-000.

   Publication No. 270-100102-001.

   Network Project Note 25.

   Network Project Note 37.
ABBREVIATIONS

**SENSE**

- STOC  : Sender Time-Out Counter
- SSR   : Sender Status Register
- SWCR  : Sender Word-Count Register
- COF   : Control Output Fifo (Buffer)
- DOF   : Data Output Fifo (Buffer)
- SCR   : Sender Command Register
- SIAR  : Sender Interrupt Allow Register

**RECEIVER**

- RTOC  : Receiver Time-Out Counter
- RSR   : Receiver Status Register
- RWCR  : Receiver Word-Count Register
- CIF   : Control Input Fifo (Buffer)
- DIF   : Data Input Fifo (Buffer)
- RCR   : Receiver Command Register
- RIAR  : Receiver Interrupt Allow Register

**MODCOMP**

- XDMP  : External Direct Memory Processor
- TA    : Transfer Address
- TC    : Transfer Count
- SI    : Service Interrupt
- DI    : Data Interrupt
- OCX   : Output Command
- ODX   : Output Data
- IDX   : Input Data
- ISX   : Input Status

Table 1
Fig. 1 - CERNET Hardware Configuration
Block diagram
Fig. 3 – SERIAL TRANSMISSION UNIT
Fig. 4 — INFORMATION FLOW
<table>
<thead>
<tr>
<th>P. ROM OUTPUT</th>
<th>GROUP</th>
<th>T. ROM ADDRESS</th>
<th>T. ROM OUTPUT</th>
<th>FUNCTION MUX.</th>
<th>RESULT</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>C</td>
<td>0</td>
<td>FO</td>
<td>A&lt;sub&gt;0&lt;/sub&gt;</td>
<td>FA&lt;sub&gt;0&lt;/sub&gt;</td>
<td>Lower byte I/O instruction</td>
</tr>
<tr>
<td>41</td>
<td>C</td>
<td>1</td>
<td>FO</td>
<td>A&lt;sub&gt;1&lt;/sub&gt;</td>
<td>FA&lt;sub&gt;1&lt;/sub&gt;</td>
<td>&quot; &quot; &quot; &quot;</td>
</tr>
<tr>
<td>42</td>
<td>C</td>
<td>2</td>
<td>00</td>
<td>A&lt;sub&gt;2&lt;/sub&gt;</td>
<td>0A&lt;sub&gt;2&lt;/sub&gt;</td>
<td>Lower byte TC</td>
</tr>
<tr>
<td>43</td>
<td>C</td>
<td>3</td>
<td>20</td>
<td>A&lt;sub&gt;3&lt;/sub&gt;</td>
<td>2A&lt;sub&gt;3&lt;/sub&gt;</td>
<td>&quot; &quot; TA</td>
</tr>
<tr>
<td>44</td>
<td>C</td>
<td>4</td>
<td>42</td>
<td>2</td>
<td>42</td>
<td>Upper byte OCC</td>
</tr>
<tr>
<td>45</td>
<td>C</td>
<td>5</td>
<td>46</td>
<td>6</td>
<td>46</td>
<td>&quot; &quot; ODC</td>
</tr>
<tr>
<td>46</td>
<td>C</td>
<td>6</td>
<td>4A</td>
<td>A</td>
<td>4A</td>
<td>&quot; &quot; ISC</td>
</tr>
<tr>
<td>47</td>
<td>C</td>
<td>7</td>
<td>4E</td>
<td>E</td>
<td>4E</td>
<td>&quot; &quot; IDC</td>
</tr>
<tr>
<td>40</td>
<td>D</td>
<td>8</td>
<td>FO</td>
<td>A&lt;sub&gt;0&lt;/sub&gt;</td>
<td>FA&lt;sub&gt;0&lt;/sub&gt;</td>
<td>Lower byte I/O instruction</td>
</tr>
<tr>
<td>41</td>
<td>D</td>
<td>9</td>
<td>FO</td>
<td>A&lt;sub&gt;1&lt;/sub&gt;</td>
<td>FA&lt;sub&gt;1&lt;/sub&gt;</td>
<td>&quot; &quot; &quot; &quot;</td>
</tr>
<tr>
<td>42</td>
<td>D</td>
<td>A</td>
<td>10</td>
<td>A&lt;sub&gt;2&lt;/sub&gt;</td>
<td>1A&lt;sub&gt;2&lt;/sub&gt;</td>
<td>Lower byte TC</td>
</tr>
<tr>
<td>43</td>
<td>D</td>
<td>B</td>
<td>30</td>
<td>A&lt;sub&gt;3&lt;/sub&gt;</td>
<td>3A&lt;sub&gt;3&lt;/sub&gt;</td>
<td>&quot; &quot; TA</td>
</tr>
<tr>
<td>44</td>
<td>D</td>
<td>C</td>
<td>43</td>
<td>3</td>
<td>43</td>
<td>Upper byte OCD</td>
</tr>
<tr>
<td>45</td>
<td>D</td>
<td>D</td>
<td>47</td>
<td>7</td>
<td>47</td>
<td>&quot; &quot; ODD</td>
</tr>
<tr>
<td>46</td>
<td>D</td>
<td>E</td>
<td>4B</td>
<td>B</td>
<td>4B</td>
<td>&quot; &quot; ISD</td>
</tr>
<tr>
<td>47</td>
<td>D</td>
<td>F</td>
<td>4F</td>
<td>F</td>
<td>4F</td>
<td>&quot; &quot; IDD</td>
</tr>
</tbody>
</table>

Note: I/O instructions use R15. Sender Address: A<sub>0</sub>. Receiver Address: A<sub>1</sub>

FIGURE 7