SIO2BT NETWORKING

v2.2

TCP/IP Networking over SIO

SIO protocol basics

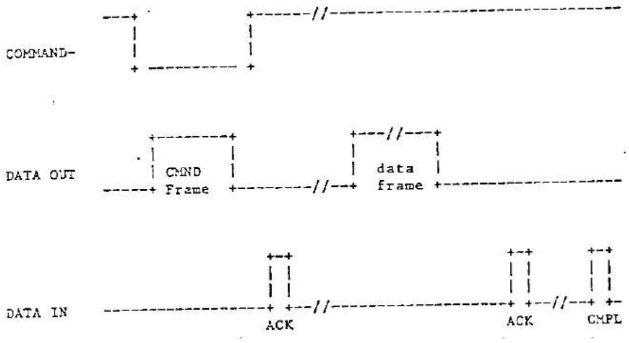
The bus protocol specifies that all commands must originate from the computer, and that peripherals present data on the bus only when commanded to do so.

Every bus operation goes to completion before another bus operation is initiated (no overlap).
An error detected at any point in the bus operation will abort the entire sequence.

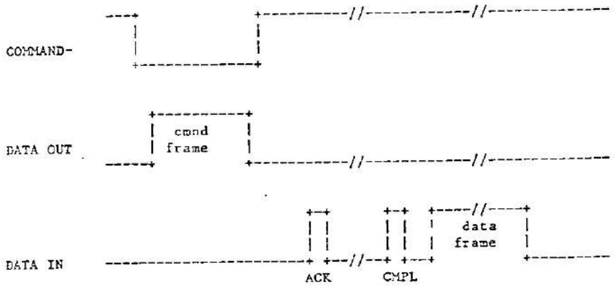
A bus operation consists of the following elements:
- Command Frame (From Computer)
- Acknowledge Frame (From Peripheral)
- Optional Data Frame (To or From Computer)
- Complete Frame (From Peripheral)

The serial bus protocol provides for three types of commands (as seen from ATARI side):

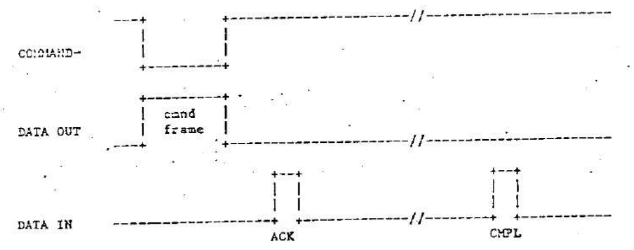
A) Data Send



B) Data Receive



C) Immediate (No Data – Command only)



Device control block

DDEVIC [$0300 (768)]
Device ID

DUNIT [$0301 (769)]
Device number, if more than one

DCOMND [$0302 (770)]
Command

DSTATS [$0303 (771)]
Before the SIO call, this byte tells whether the operation is "Data Receive", "Data Send" or that there is no data transfer associated with the command. After the call this byte will hold the status (error/no error code) of the operation.

DSTATS format before command

```
 7 6 5 4 3 2 1 0
-----------------
|S|R| not used  |
-----------------
```

If both S and R are 0, there is no data transfer.

DBUFLO [$0304 (772)]
DBUFHI [$0305 (773)]
Points to the data buffer for either input or output.

DTIMLO [$0306 (774)]
Timeout value (response time limit) in 64/60ths of a second to be set by handler or program.

DBYTLO [$0308 (776)]
DBYTHI [$0309 (777)]
Number of bytes to be transferred, set by handler or program. This parameter is not required if the DSTATS specifies no data transfer.

DAUX1 [$030A (778)]
DAUX2 [$030B (779)]
These parameters are sent to the device as part of the command frame.

After the control block parameters are set, a JSR is made to the SIO entry vector, SIOV, at $E459 (58457).

Any networking operation on the serial bus begins with a five byte command frame:

$46 DDEVIC (device id of the Networking Device)
$xx DCOMND
$xx DAUX1
$xx DAUX2
$xx checksum

The first four bytes of the command frame come from the device control block. The checksum is the sum of the other four bytes with the carry added back after each addition.

If both R and S of the DSTATS are 0, no data is sent to, or expected from the peripheral, after a command frame is sent. However, the device is expected to send an ACK byte ($41) after the command frame is sent. If the command frame is invalid, an NAK byte ($4E) should be sent.
If the operation is “Data Send” (S = 1) the computer will send a data frame after it receives the ACK of the command frame. It then expects an ACK after the data frame is sent.
If the operation is “Data Receive” (R = 1) the computer expects a data frame from the peripheral after the ACK.
A "complete" code ($43) should always be sent to the computer when the operation is finished. If the operation is not completed for some reason, the peripheral should send an “error” code ($45) instead of "complete".

SIO data frame

```
byte 1     $xx\
               > data bytes
byte n     $xx/
byte n+1   $xx   checksum
```

Error Handling

SIO defines basic error handling: $4E - NAK byte and $45 - ERROR byte (see above).
In error case, the SIO procedure does a number of re-tries and eventually returns with an error code (DSTATS). This approach is good for handling SIO communication issues, but TCP/IP networking problems have to be handled at a higher level (simple automatic re-try at the SIO layer would not work). That's why TCP/IP errors are not reported with the ERROR byte.
All (valid) requests are always answered with the “COMPLETE” byte ($43), which ensures in the worst case, that the time spent in the SIO procedure will not exceed the time limit.
The application should query the status to detect TCP/IP issues (for example, when a read request returned 0 bytes read). Networking operations can be time consuming, so the DTIMLO value should be big enough, to avoid triggering timeouts in the SIO procedure.
The networking device has its own timeout values (they have to be smaller than DTIMLO and are adjustable in the SIO2BT app) to assure answering the SIO requests in a timely manner.

Networking Device

SIO DDEVIC: **$4E**
SIO DUNIT: **$01**
SIO DTIMLO (response time limit): **$E0**

SIO commands:

ID	Name	Type
$53	**GET STATUS**	**Data Receive**
AUX1: - AUX2: - The ATARI will always receive a block of data with size of 5 bytes. The first 4 bytes (corresponding to the connection ID 0-3) contain connection status information: bits 7-1: - reserved for future bit 0: 0 – OK, 1 – error (since the last GET STATUS) The last byte is the networking device status: bits 7-1: - networking device type (SIO2BT = 0) bit 0: 0 – no network connection, 1 – network connection available		

ID	Name	Type
$4F	**OPEN**	**Data Send**
AUX1: Protocol (6 most significant bits) + Connection ID (2 least significant bits) AUX2: Data Length (0 = 256 bytes) SIO2BT supports only raw socket TCP/IP connection (Protocol = 0). The host and port information is included in data frame sent with the command. Data length should be given in AUX2. The host name / IP address + port number should be ASCII coded and must not include the end of line character ($9B). The format is: “*host:port*”. Examples: “mydomain.net:5070” or “192.168.0.1:5000” The ATARI can OPEN up to 4 TCP/IP client connections. The networking device will try to open a connection to the server (with a given host name / IP address, which is listening on a given port).		

<table>
<tr><th>ID</th><th>Name</th><th>Type</th></tr>
<tr><td>**$52**</td><td>**READ**</td><td>**Data Receive**</td></tr>
<tr><td colspan="3">AUX1: Connection ID (2 least significant bits)
AUX2: Number of bytes to read over TCP/IP (1-255)

The networking device will try to read a number of bytes (AUX2) over a TCP/IP connection identified with AUX1. The READ call shall be considered as a not blocking call.
It is actually implemented as a blocking call with a timeout (adjustable via SIO2BT app settings).
If less bytes as requested (or even no bytes at all) are waiting in the buffer, the ATARI will receive only that many bytes.
The ATARI will always receive a data frame with the requested size **plus one (AUX2+1)**.
The last byte in the data frame is a number of bytes successfully read over TCP/IP.
When less bytes were read as requested, the rest will be filled with 0s.</td></tr>
</table>

<table>
<tr><th>ID</th><th>Name</th><th>Type</th></tr>
<tr><td>**$50**</td><td>**WRITE**</td><td>**Data Send**</td></tr>
<tr><td colspan="3">AUX1: Connection ID (2 least significant bits)
AUX2: Number of bytes to write over TCP/IP (1-255)

The networking device will try to write a number of bytes (AUX2) to a TCP/IP connection (AUX1).</td></tr>
</table>

<table>
<tr><th>ID</th><th>Name</th><th>Type</th></tr>
<tr><td>**$43**</td><td>**CLOSE**</td><td>**Immediate**</td></tr>
<tr><td colspan="3">AUX1: Connection ID (2 least significant bits)
AUX2: -

The networking device will close the TCP/IP connection identified with AUX1.</td></tr>
</table>

Communication sequence includes the following steps:

1. ATARI sends the GET STATUS command to find out if a network connection is available
2. ATARI sends the OPEN command
3. ATARI sends the GET STATUS command to find out if the OPEN command succeeded
4. ATARI sends the READ/WRITE and GET STATUS commands (in a loop)
5. ATARI sends the CLOSE command

Error handling:
Invalid AUX1/AUX2/Data Frames are handled with SIO error handling (NACK).
Networking problems are not reported this way (valid OPEN / READ / WRITE commands are always answered with a ACK/COMPLETE bytes).
In order to check if the last OPEN/READ/WRITE operation was really successful, the application needs to send GET STATUS and decide how to handle possible networking errors.