XNET Formats for Internet Protocol Version 4
Jack Haverty
Bolt Beranek and Newman Inc.
October 1, 1980

This IEN is intended to capture in print the formats used currently in the version 4 XNET protocol; most of the data is courtesy of Ray Tomlinson.

Version 4 XNET is identical with version 2.5 XNET with the exceptions listed below. The version 2.5 format is described in RFC 643. It should be noted that the manner in which the protocol is used by a user program (such as the PDP10 XNET program), and by the various target-machine XNET servers, is not defined herein. In particular, there are several problems and heuristics in dealing with the operation of the protocol in the internet environment, where individual packets may be duplicated, lost, and reordered.

Changes from the version 2 formats include the following:
1) XNET header and data is embedded in a IN V4 packet instead of a V2.6 packet.
2) Packet format changed to add Port, Sequence number, and Checksum fields.
3) Change in synchronous reply codes.
4) Addition of ACK bit to opcode field.
5) Positive acknowledgement of all messages.
Packet Format

```
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| Port            | Sequence        | MSB             | LSB             | Checksum        |
| Port            | Sequence        | MSB             | LSB             | Checksum        |
|                | PID             | ICNT|ACK|      | Opcode          |
|                | Argument 1      | MSB             | LSB             | Argument 2      | MSB             |
|                | Argumet 2      | MSB             | LSB             |                |
|                | Data            |                 |                 |                |
```

The IN protocol is set to the XNET protocol number (17 octal).

Host to target opcodes

- **NOP** 0: No operation
- **DEBUG** 1: Start debugging a process or address space.
- **ENDBUG** 2: End debugging a process or address space.
- **HALT** 3: Halt the process.
- **DPOSIT** 4: Deposit in memory.
- **RESUME** 5: Resume execution of a process.
- **EXAM** 6: Examine memory.
- **DSV** 7: Deposit state vector (r0-r5,sp,pc,ps).
- **SETBPT** 10: Set breakpoint.
- **REMBPT** 11: Remove breakpoint.
- **ONESTP** 12: Single step process (using trace trap).
- **PROC** 13: Proceed from breakpoint.
- **CREAP** 14: Create a new process (or address space).
- **DSTROY** 15: Destroy (delete) a process or address space.
- **XIOREP** 16: Reply to XIO output (not used anymore).
- **XINREP** 17: Reply to XIO input.
- **DEFALL** 20: Define and allocate memory to an address space.
- **SAP** 21: Start all processes.
- **SAVDSK** 22: Save on disk.
- **GETDSK** 23: Get from disk.
- **ENTRST** 24: Enter address space into restart table.
Opcodes from target to host machine.

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HALTED</td>
<td>77</td>
<td>Process halted (FREET with arguments of 0).</td>
</tr>
<tr>
<td>TRAPPED</td>
<td>76</td>
<td>Process trapped due to error.</td>
</tr>
<tr>
<td>TTRAP</td>
<td>75</td>
<td>Trace trap.</td>
</tr>
<tr>
<td>BPT</td>
<td>74</td>
<td>Breakpoint hit.</td>
</tr>
<tr>
<td>XIOIN</td>
<td>73</td>
<td>XIO input request.</td>
</tr>
<tr>
<td>XIOOUT</td>
<td>72</td>
<td>XIO output request.</td>
</tr>
</tbody>
</table>

Checksum

The checksum is the same as that for the IN header; ones complement of ones complement sum of words in the packet from Port field to last data word inclusive. In case of an odd number of data bytes, an additional byte of zeroes is assumed for checksum purposes.

Port number

The port number is a unique number relative to the host machine which appears in every packet for a particular debugging session. It is suggested that this number be derived from the time of day so that each session will be unique over a long period of time.

Sequence number

The first packet of a session (first use of a particular port number) is numbered 0. Subsequent packets increment by 1 modulo 2**16. Packets initiated by the target machine (opcodes
72-77) are also numbered starting from 0. The target machine is allowed to execute packets out of order but must never execute a packet twice unless the effect is harmless. For example, an examine packet should be re-executed so that the data may be returned to the sender. Deposit or resume should not be re-executed. The host machine is responsible for correct ordering of critical functions. For example, it must not send a RESUME command until all prior deposits have been acknowledged.

Acknowledgements

Each packet must be acknowledged by the receiver. An acknowledgement consists of the original header plus any requested data (e.g. EXAM) with the ACK bit set. Acknowledgements are not cumulative; an acknowledgement acknowledges only the one packet with the matching sequence number. If the target debugger is incapable of performing the requested function, it should set the CNT (can't) bit instead of the ACK bit. Both bits may be set meaning that the function is available but the data required is no longer available. This might be the result of a duplicate packet.