

### HIGHLIGHTS

- High Performance
- Robust and RFC Compliant
- Multicasting and IGMPv2 for Low-Overhead Streaming Audio and Video
- Optional SNMPv3 support with Integrated MIB-II Agent
- APIPA for Automatic IP Address Assignment on Local Networks
- Optional Embedded Web Server
- Compact and Configurable
- Built-in Protocol Decoder
- Works with or without an RTOS

Blunk Microsystems provides system software, device drivers, and board support packages to the embedded systems market, both off-the-shelf products and custom work done under contract.



BLUNK  
Microsystems

TargetTCP is a fast, reliable, re-entrant TCP/IP implementation developed and supported by Blunk Microsystems that has been extensively tested and demonstrated interoperable with Windows, UNIX, and other TCP/IP protocol stacks.

RFC-compliant with a full protocol suite. Supports TCP, UDP, IP, ICMP, Multicasting, IGMPv2, ARP, APIPA, Ethernet, PPP (with CHAP, CHAT, and PAP), PPPoE, SMTP, NAT, and Wi-Fi. Includes servers for FTP, RARP, Telnet, and TFTP. Includes clients for DNS, DHCP, RARP, SNMPv1 (optional SNMPv3), and TFTP.

Standard Berkeley sockets application program interface with enhancements:

- `connect()`, `recv()` and `send()` timeouts can be set by `setsockopt()`.
- `MSG_WAITALL` flag makes `recv()` block until all requested bytes have been received.
- `setsockfunc()` installs a socket callback function, supporting event driven programming and allowing a single task to service both socket and other events.
- `setsockopt()` `IP_TOS` and `IP_TTL` options set the IP header type-of-service and time-to-live values on a per socket basis.

**High Performance.** No data copies are performed within the stack. Matching of IP fragment lists, ARP address records, and TCP sockets is done with a fast hashing function. The TCP and UDP checksum calculation is done in assembly language (for PowerPC, ColdFire, and 68K). Access to the assembly language “add with carry” instruction enables the calculation to be done with fast 32-bit accesses.

Supports RISC processors without requiring an extra data copy within the Ethernet driver to align packets. IP, TCP, and UDP headers are 4-byte aligned within the stack, allowing

fast access to the 32-bit fields in these structures. Application data remains unaligned, eliminating an extra copy required by Berkeley-derived implementations.

Using the Berkeley sockets API, only one copy of application data is performed. This applies to inbound and outbound transfers using both UDP datagrams and TCP streams. A zero-copy API is provided which eliminates the copy associated with the sockets API.

Supports unnumbered serial links (RFC1716) to avoid wasted IP addresses and artificial subnets on targets with multiple PPP interfaces.

Supports TCP out-of-band data and urgent data mark. Both socket callback functions and `select()` can be used to notify applications when a connected peer has entered urgent mode.

**Easy to administer.** The DHCP client may be configured to use gateway and DNS server addresses supplied by the server. The RARP client may be configured to use the RARP server as a default gateway. PPP connections can be configured to be a default gateway and to request a DNS server address from the remote peer.

**Integrated.** The Telnet server is integrated with a command line monitor that provides a full set of built-in commands, access to the TargetOS%o command line monitor, and easy extensibility. The FTP server uses TargetTCP's zero-copy interface and will serve files using any POSIX compatible file system. The TFTP client is integrated with the TargetOS runtime library, allowing access to remote files via `fopen()`, and with the TargetOS loader, which accepts both ELF and S-record formats.

**Includes an SMTP client.** Send email from your embedded target using only the SMTP server name or IP address, the sender and recipient email addresses, a subject string, and a pointer to the message text. ▶

## CONTACT INFORMATION

- Visit our web site:  
[www.blunkmicro.com](http://www.blunkmicro.com)
- Customer Support:  
**(408) 323-9833**
- Technical Support:  
**(408) 323-1758**
- Fax:  
**(408) 323-1757**
- E-mail:  
[sales@blunkmicro.com](mailto:sales@blunkmicro.com)
- Address:  
**6576 Leyland Park Drive  
San Jose, CA 95120  
USA**

## AVAILABLE COMPONENTS

### TargetOS

Real-time, deterministic, multi-tasking, priority-based, preemptive kernel. Includes Standard C library.

### TargetFFS

POSIX compatible flash file system. Implements wear-leveling to prolong life of flash media. File system integrity is guaranteed across unexpected resets.

### TargetLAPB

ISO/IEC 7776 protocol stack. Supports exchanging data on point-to-point networks. Provides automatic flow control and data reliability.

Ideal for sending alerts when a limit is approached or a critical event occurs.

TargetTCP has a clearly documented network driver interface and supports concurrent use of multiple network interfaces. Ethernet and PPP drivers for the MCF5475, MCF5272, MPC860, DP83815, i82559, LAN91C111, and other controllers are available separately. Blunk Microsystems provides competitive bids on custom drivers.

**Built-in TCP/IP protocol decoder.** When enabled by a compile-time flag, a short summary of every received or transmitted packet is printed to stdout on an ongoing basis. Ideal for testing new drivers or verifying new configurations.

**Compact and configurable.** Applications typically require approximately 30 KB of code from the TargetTCP library and can be configured to use as little as 32 KB of RAM data.

Shipped with several sample applications: an SNMP time client, a chargen, discard, and echo client, a chargen, discard, and echo server, and an HTML client that cycles through a list of URLs, downloading each site's home page.

Developed and maintained using TargetOS, Blunk Microsystems' real-time operating system. Easily ported to other real-time kernels or to polling environments that do not use a kernel. A pSOS port is maintained for Philips' Nexperia™ processor.

The TargetTCP source code is 100% ANSI C (other than the optional assembly language checksum routine) and is tested using GCC and CodeWarrior™, the embedded IDE from Metrowerks.

Royalty free. Includes full source code, user's manual, sample applications, and one year of technical support. ■

### The TargetTCP Socket Interface:

```
int accept(int socket, void *addr,
int *addrlen);
Accepts connections for TCP servers.

int bind(int socket, void *addr,
int addrlen);
Attaches a local port number and IP address
to a socket.

int closesocket(int socket);
Closes the socket connection, if one exists,
and recovers socket resources.

int connect(int socket, const void
*addr, int addrlen);
Assigns a remote socket address and
actively initiates TCP connections.

int getpeername(int socket, void
*addr, int *addrlen);
Retrieves a socket's remote IP address and
port number.

int getsockname(int socket, void
*addr, int *addrlen);
Retrieves a socket's local port number and,
if connected, IP address.

int getsockopt(int socket, int
level, int optname, void *optval,
int *optlen);
Reads various options that control socket
or protocol behavior.

unsigned long htonl(unsigned long
hostlong);
Converts 32-bit integers from host byte
order to network byte order.

unsigned short htons(unsigned short
hostshort);
Converts 16-bit integers from host byte
order to network byte order.

ui32 inet_addr(const char ptr);
Converts an IP address from a dotted-decimal
string to a network byte ordered integer.
```

```
int ioctlsocket(int socket, int
request, void *arg);
Sets or gets a socket's mode.

int listen(int socket, int backlog);
Prepares TCP sockets to accept client connections.

unsigned long ntohl(unsigned long
netlong);
Converts 32-bit integers from network order
to host byte order.

unsigned short ntohs(unsigned short
netshort);
Converts 16-bit integers from network order
to host byte order.

int recv(int socket, void *buffer,
int buflen, int flags);
Receives data from a socket.

int recvfrom(int socket, void
*buffer, int buflen, int flags,
void *from, int *fromlen);
Receives data from a socket.

int select(int fdnum, fd_set *reads,
fd_set *writes, fd_set *excepts,
struct timeval *timeout);
Simultaneously monitors multiple sockets.

int send(int socket, void *buffer,
int buflen, int flags);
Sends data over a connected socket.

int sendto(int socket, void *buffer, int
buflen, int flags, void *to, int tolen);
Sends data over a socket.

int setsockopt(int socket, int level, int
optname, void *optval, int optlen);
Sets various options that control socket or
protocol behavior.

int shutdown(int socket, int how);
Shuts down a socket connection.

int socket(int family, int type, int
protocol);
Allocates a TCP or UDP socket.
```

## Licensing Terms

TargetTCP™ is royalty free. Purchasers are granted a non-exclusive license to use the provided source code at a single site. Licensees have the right to disseminate or resell the software in executable format only. The source code and derived object code may not be redistributed or resold.