TRANSTECH

Parallel Systems

MatchBox User Manual

Ref: MBX M 711

Document reference number MBX M 711.

Transtech Parallel Systems Corp
20 Thornwood Drive
Ithaca
NY 14850-1263
USA

tel: 607 257 6502
fax: 607 257 3980

transtech@transtech.com
http://www.transtech.com

Transtech Parallel Systems Ltd
17-19 Manor Court Yard
Hughenden Avenue
High Wycombe
Bucks, HP13 5RE
United Kingdom

tel: +44 (0) 1494 464303
fax: +44 (0) 1494 463686

support@transtech.co.uk
http://www.transtech.co.uk

Table of Contents

Chapter 1

Introduction

The Transtech MatchBox is primarily designed to interface the SCSI port of a workstation to Transtech PARAstation systems. However, the MatchBox can equally be used to interface to a wide range of other Inmos Transputer based systems.

On multi-user workstations, up to four users can concurrently access the PARAstation resources per MatchBox. In addition to these four host links, a fifth configuration link is implemented which allows the network configuration of the PARAstation to be set.

When using a Transtech PARAstation system, this manual should be read in conjunction with the PARAstation User Manual.

Please note that a password is required to use the MatchBox with a UNIX host. See section 3.2 on page 12 for details. Passwords can be obtained from your local Transtech office.

1.1 Manual Conventions

Commands, filenames e.t.c are printed in `courier font`. Command lines which are to be typed at the Unix prompt are printed thus:

```
% command
```

except for commands which are required to be run by root or superuser which are printed thus:

```
# command
```

The contents of files or the output of programs is printed thus:

```
First line
Second line
```

Where relative filenames or directory names are quoted (i.e. filenames not starting with a '`/`') then these should be interpreted as being relative to the software installation directory. See the relevant software installation section for details of the installation directory.

Chapter 2

Hardware Installation

The Transtech MatchBox needs to be connected to:

- The power supply.
- The host workstation SCSI bus.
- The Transtech PARAstation.

The MatchBox connectors and switches are shown in figure 1, which is a rear view of the MatchBox.

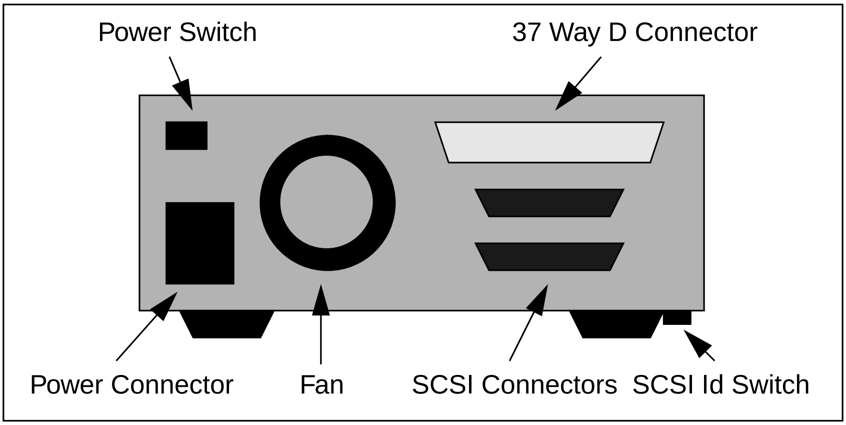

Figure 1. MatchBox Back Panel

2.1 Power Connection

Mains power at the correct voltage needs to be supplied to the workstation, the MatchBox and the PARAstation. The MatchBox takes AC voltages in the range 90V to 260V.

For reliable operation, ensure that the workstation, the MatchBox, the PARAstation and any other peripherals are all powered from the same mains circuit.

2.2 SCSI Connection

The MatchBox is connected to the computer workstation by the SCSI bus. This requires you to:

- Choose and set the MatchBox SCSI id.
- Connect the MatchBox to the host workstation SCSI port and other SCSI devices (if any).
- Terminate the SCSI bus.

Shutdown and switch off the workstation and any connected SCSI peripherals before connecting the MatchBox.

Most UNIX workstations have a built in SCSI port accessible from the rear. A typical workstation SCSI setup is illustrated in figure 2 where the workstation has an internal disk and the MatchBox is the only external SCSI peripheral. Alternatively, it is possible to have more than one SCSI adapter installed in the workstation and have the MatchBox on a different SCSI bus from other SCSI peripherals.

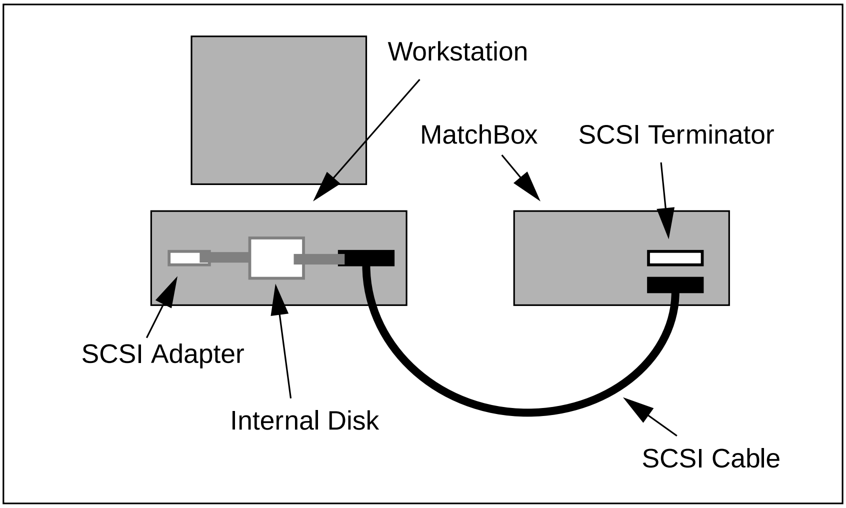

Figure 2. SCSI Connections

2.2.1 SCSI Cabling and Termination

Most SCSI peripherals, including the MatchBox, have two SCSI connectors. This allows multiple SCSI peripherals to be connected to one workstation. It does not matter which MatchBox SCSI connector is used to connect to the workstation and which is used to connect to other SCSI peripherals.

Problems can occur if the SCSI cables are of poor quality or if their total length is too big. The shorter the SCSI bus cables the better. The SCSI-2 specification requires that the total cable length (including internal cables) is no greater than 6m. As a rule of thumb, do not allow external cabling to exceed 3m.

The SCSI bus must be terminated using a suitable terminator at both ends. The end internal to the workstation will normally be terminated on the SCSI adapter. If the MatchBox is the last external SCSI peripheral, fit the supplied terminator to the unused MatchBox SCSI connector. If a SCSI peripheral is used which has internal SCSI bus termination, then this must be the last device on the SCSI bus.

2.2.2 MatchBox SCSI Id

The SCSI id of the MatchBox is set using a switch on the underside of the box as shown in figure 3.

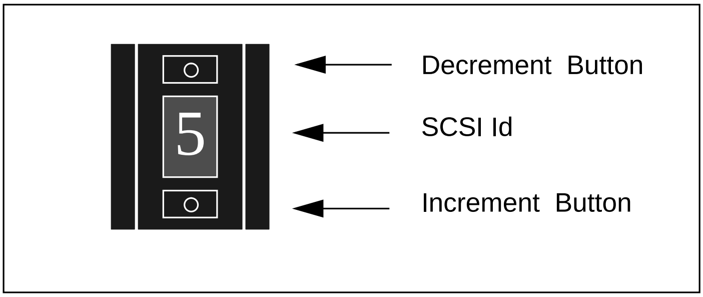

Figure 3. SCSI Id Switch

A ballpoint pen or other suitable object can be used to increase or decrease the value by one until the required value is set as required. You need to use an unused value between 0 and 7. A SCSI id of 5 is often suitable.

Each internal and external peripheral connected to the SCSI bus (including the workstation SCSI adapter) has a unique id between 0 and 7. The SCSI bus will fail to operate if more than one device has the same SCSI id. To choose an id for the MatchBox you need to know the SCSI ids of all of the other devices already on the SCSI bus.

2.2.2.1 Sun SPARCstations

On Sun SPARCstations there is a monitor command `probe-scsi` which can be used to determine the ids of peripherals connected to the SCSI bus:

1. If any external SCSI peripherals are used, ensure that they are properly connected to the Sun and the SCSI bus is terminated.
2. Switch on the Sun monitor and external SCSI peripherals.
3. Switch on the Sun workstation.
4. When the Sun logo appears and the monitor tests start, hold down the `L1` key and press the `A` key. This stops the tests and brings up the monitor prompt.
5. If the monitor prompt is '>' then type `n` and press `Return`.
6. At the `ok` prompt type `probe-scsi` and press `Return`.

A list of SCSI target devices like the following is produced:

```
Target 1
 Unit 0  Disk     MAXTOR LXT-231S     4.18
```

This shows that there is a disk with SCSI id 1 with one logical unit. In fact it is a Maxtor LXT-231S disk (revision 4.18) which is installed internally in the Sun.

The Sun workstation SCSI adapter is 7 by default, thus in this case a suitable choice for the MatchBox SCSI id is 5.

When a MatchBox is correctly connected to the host SCSI bus, for example with SCSI id 5, the `probe-scsi` command will additionally display messages similar to:

```
Target 5
 Unit 0  Processor     TRANST  MatchBox     1.00
 Unit 1  Processor     TRANST  MatchBox     1.00
 Unit 2  Processor     TRANST  MatchBox     1.00
 Unit 3  Processor     TRANST  MatchBox     1.00
 Unit 4  Processor     TRANST  MatchBox     1.00
 Unit 5  Processor     TRANST  MatchBox     1.00
Unit 6  Processor     TRANST  MatchBox     1.00
Unit 7  Processor     TRANST  MatchBox     1.00
```

showing the eight MatchBox logical units.

At this point you can safely switch off the Sun.

2.3 PARAstation Connection

The Transtech MatchBox is connected to the PARAstation by a cable which connects to 37 way D connector on both systems as shown in figure 4.

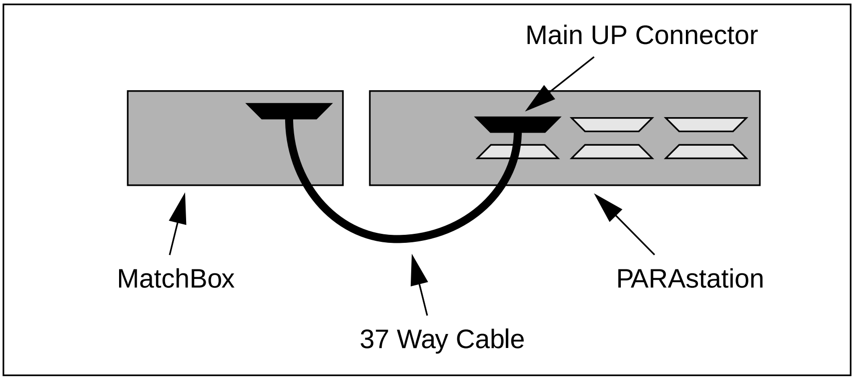

Figure 4. PARAstation Connection

2.3.1 D Connector

The connection between the MatchBox and the PARAstation, or to other external Transputer systems, is made via the MatchBox 37 way D connector. The pin-out for this is shown in figure 5.

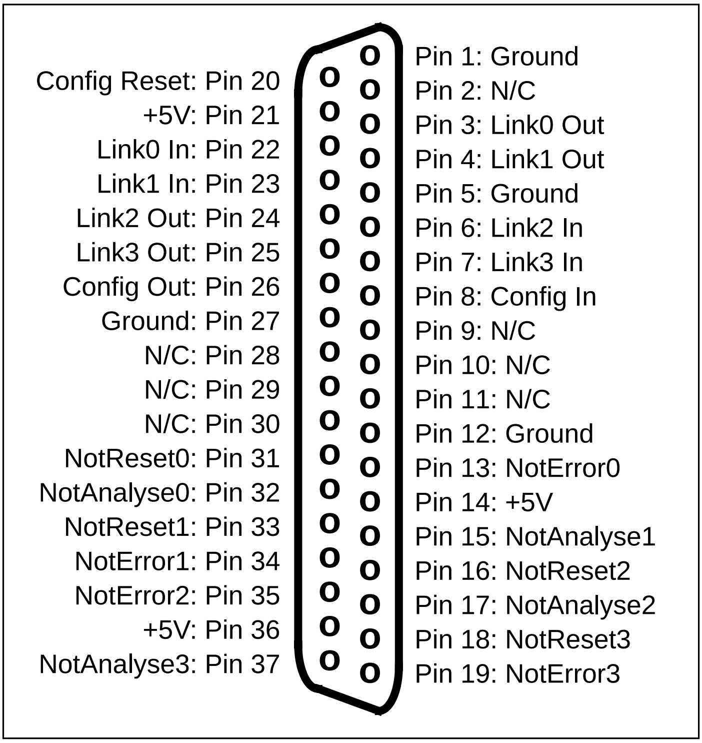

Figure 5. 37 Way D Connector

A Break-out board is supplied which plugs into the 37 way D connector and has a number of link headers which can be used to connect the MatchBox to Transputer networks using standard Inmos link cables. The connectors available on the Break-out board are shown in figure 6.

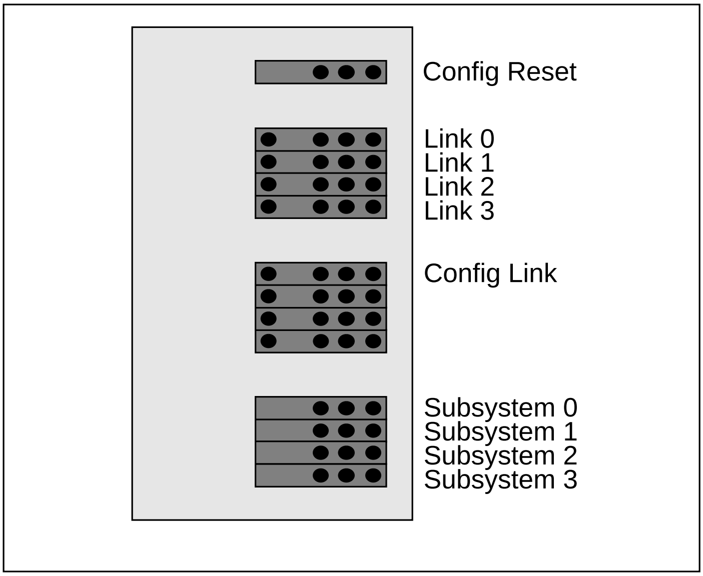

Figure 6. Break-out Board Connections

Chapter 3
Software Installation

Instructions are given here for installing the MatchBox software on Sun SPARCstations running Solaris 2.

A password is required to use the Transtech MatchBox connected to UNIX hosts. See section 3.2 for details.

It is recommended that the MatchBox is connected to the host workstation before installing the MatchBox software.

3.1 Sun SPARCstation

The MatchBox software for Sun SPARCstation systems running Solaris 2 is supplied on CD-ROM - the contents of which are installed using the `pkgadd` command. Packages that are installed using the `pkgadd` command can be removed using the `pkgrm` command.

Most of the software is installed by default in the directory `/opt/transtech`. This software can in fact be installed anywhere in the file system, for instance on a networked file server disk, but it is recommended that a symbolic link is made to the installation directory from `/opt/transtech` if the software is not installed there.

To install the MatchBox software on Sun SPARCstation systems running Solaris 2:

1. Log into the Sun as root.
2. Put the supplied CD-ROM into the CD-ROM drive.
3. Use `volcheck` to mount the CD-ROM:

```
# volcheck cdrom
```

4. Move to the `solaris2` directory on the CD-ROM:
   ```
   # cd /cdrom/cdrom0/solaris2
   ```
5. Add the Solaris 2 packages:
   ```
   # pkgadd -d ‘pwd‘ all
   ```
   When prompted for an installation directory, press Return for the default (`/opt/transtech`) or type in an alternative installation directory.
6. Eject the CD-ROM from the drive:
   ```
   # eject cdrom
   ```

3.2 Password

A password file `/opt/transtech/tdpasswd` is required on Solaris 2 systems. This file contains one or more passwords - one for each host computer with a connected MatchBox.

Passwords can be obtained from your local Transtech office by quoting the host id of the workstation as displayed by the `hostid` command.

3.3 Environment

A number of environment variables should be set:

`IBOARDSIZE`	Memory size of the first Transputer board.
`IDEBUGSIZE`	Memory size of the first Transputer board used by `idebug`.
`ITERM`	The file containing terminal keyboard and screen codes.
`ISEARCH`	The list of directories on which the Iserver will search for certain files if the full pathname is not specified. Include the `tp/4libs` directory for use with the 4th generation Inmos ANSI C Toolsets Dx414.
`ISERVER`	Full pathname of the Iserver program. Relative to the installation directory this is `bin/iserver`. Iserver is described in Chapter 5.
`ICONDB`	Full pathname of the connection database. See section 4.1 for details.
`ASERVDB`	Full pathname of the Aserver database file.

`TRANSPUTER`	The host link to use. See Chapter 4 for details.
`MANPATH`	Search path for the Unix `man` command. Include the full pathname of the `man` installation directory to access the on-line manual pages.

The command search path should include the `bin` directory. If Inmos Toolsets are being used then this should occur before any Inmos Toolset directory is quoted.

The following lines are an example extract of the file `.login` which can be used to set these environment variables for users of the C shell under Solaris 2:

```
setenv TPS /opt/transtech
setenv IBOARDSIZE ‘#400000’
setenv IDEBUGSIZE ‘#400000’
setenv ITERM /usr/inmos/d4414/iterms/sun.itm
setenv ISEARCH “/usr/inmos/d4414/libs/ $TPS/tp/4libs/”
setenv ISERVER $TPS/bin/iserver
setenv ICONDB $TPS/lib/tsp.db
setenv ASERVDB $TPS/tp/4libs/aservdb
setenv TRANSPUTER c0t5l0
setenv MANPATH /usr/man:/usr/openwin/man:$TPS/man
set path = ( $path $TPS/bin /usr/inmos/d4414/tools \
 /usr/inmos/d4405/tools )
```

3.4 Connection Database

Certain utilities such as the host server program `iserver` use a connection database file to determine what resources are available to it. See section 4.1 for details.

A connection database file `lib/tsp.db` is supplied which can be used with a MatchBox at any SCSI id on SCSI bus 0. This file will need to be altered if a MatchBox is connected to a SCSI bus other than 0.

3.5 Rebooting

The host workstation should be switched off whenever a MatchBox is connected to the SCSI bus, disconnected from the SCSI bus, or its SCSI id is changed.

When rebooting a Sun workstation running Solaris 2, use the `-r` boot flag to tell the host that it should reconfigure itself. With the Sun switched off:

1. Switch on the MatchBox (if connected) and any other SCSI peripherals.
2. Switch on the Sun.
3. When the Sun logo appears, hold down the `L1` key and press `A.` This stops the Sun from booting and enters monitor mode.
4. If the monitor prompt is '>' then type n and press `Return` to get the `ok` prompt.
5. Check that the Sun can see the MatchBox (if connected) or any other SCSI peripheral:

   ```
   ok probe-scsi
   ```
6. Boot the Sun with the `-r` flag. Exactly what command to type depends on which Sun system is being used and how the operating system is installed. On early SPARCstations that boot from disk, use something like:

   ```
   ok boot sd(0,1,0) -r
   ```

   On later SPARCstations that boot from disk the command will be something like:

   ```
   ok boot disk -r
   ```

Chapter 4

Host Link Names

The MatchBox host link to be used is specified using a host link name such as `c0t5l0`, which corresponds to SCSI controller 0, SCSI target 5 (the MatchBox SCSI id), link 0. Thus, for a MatchBox with SCSI id 5, the names of the host links 0 to 3 are `c0t5l0`, `c0t5l1`, `c0t5l2` and `c0t5l3`.

4.1 Connection Database

The connection database lists the capabilities (resources) available to certain tools such as the host server utility Iserver. The actual resource that the tool is to use is specified by the `TRANSPUTER` environment variable or command line arguments such as the Iserver `sl` option. The name of the connection database file to use is the value of the environment variable `ICONDB`.

The connection database file `lib/tsp.db` defines capabilities for MatchBoxes at any SCSI id on SCSI bus 0. The capability names are chosen to have the same name as the host link as defined above. An extract from this file is:

```
# SCSI id 5

|c0t5l0     |T      |localhost |c0t5l0 |tsp   ||||
|c0t5l1     |T      |localhost |c0t5l1 |tsp   ||||
|c0t5l2     |T      |localhost |c0t5l2 |tsp   ||||
|c0t5l3     |T      |localhost |c0t5l3 |tsp   ||||
```

The non-comment lines in this file contain the fields:

`Capability`	The resource name passed, for instance, to Iserver using the `sl` option or the environment variable `TRANSPUTER`.
`IsWorking`	Set to `T` if the resource is available. Set to `F` if the resource is not available.
`Machine`	The name of the host on which the resource exists. For a MatchBox connected to the local workstation, this is set to `localhost`.
`Linkname`	The device name. See above for the device naming convention used.
`Linkdev`	The type of device. Set to `tsp` to indicate the use of a Transtech SCSI Processor system such as the MatchBox.
`Mmsfile`	Reserved for future use.
`Mmslink`	Reserved for future use.
`Description`	Descriptive comment.

Thus, using the above connection database, to use Iserver on host link 0 to load and run the Transputer executable `run.btl` use the command on UNIX systems:

```
% iserver -sl c0t5l0 -sb run.btl
```

Alternatively, set the environment variable `TRANSPUTER` to the required capability name. For instance, on UNIX systems:

```
% setenv TRANSPUTER c0t5l0
% iserver -sb run.btl
```

Chapter 5
Iserver

Iserver is a host server program which loads and runs Transputer and PowerPC programs on Transtech systems allowing access to host services such as the keyboard, screen, and filesystem.

The resource that Iserver is to use is specified by the environment variable `TRANSPUTER` or the `sl` option. The host link name corresponding to this is found from the connection database. See section 4.1 on page 15 for details of the connection database file.

5.1 Source

The source of Iserver is supplied in subdirectories of the `source/iserver` directory. It is based on the Inmos version 1.50v of Iserver with some changes including:

- Support for the MatchBox is added using calls to the Transtech SCSI Processor Library in the file `linkios/tsplink.c`.
- The variable `OPScommsmode` defined in `linkops/linkops.c` is made global and an external reference to it is added to `linkops/linkops.h`. This is used in `linkios/tsplink.c` to optimise Iserver transactions.
- Support for PC systems running DOS with no network connection is added.

5.2 Iserver Libraries

The following files are supplied in the `tp/4libs` directory to be used with the Inmos ANSI C Toolset Dx414 in conjunction with Iserver:

`genio.h`	Header file for `genio.lib`.

`genio.lib`	Generic I/O library. Used to improve I/O performance of Transputer and i860 programs using Iserver.
`hostmux.lku`	Iserver host link multiplexor process.
`iocache.lku`	Disk cacheing process.

The `genio` library and the `hostmux` and `iocache` processes are documented in the form of Unix `man` pages in the following section.

5.3 Reference Section

iserver

```
NAME
     iserver - Inmos host server program

SYNOPSIS
     Unix:

     iserver [ -sb filename ] [ -si ] [ -se ] [ -sl linkname ] [
     -sr ] [ -sa ] [ -sc filename ] [ -sp n ] [ -ss ] [ -sm ] [ -
     sk n ] [ -sz[1|2] ] [ [ -st ] arguments...  ]

     DOS:

     iserver [ /sb filename ] [ /si ] [ /se ] [ /sl linkname ]  [
     /sr  ]  [ /sa ] [ /sc filename ] [ /sp n ] [ /ss ] [ /sm ] [
     /sk n ] [ /sz[1|2] ] [ [ /st ] arguments...

DESCRIPTION
     iserver is the Inmos host server program.   It  is  used  to
     load  programs  onto Transtech systems and allow them access
     to the host workstation’s keyboard, screen, file system  and
     other services.

     The list of available systems (resources) which iserver  can
     use  is  kept in a file called the connection database.  The
     environment variable ICONDB should be set to the pathname of
     this  file.  The resource that iserver should attempt to use
     is specified by the sl option or  the  environment  variable
     TRANSPUTER.

     The session manager provides a mechanism to allow users con-
     tinuous  access to a resource.  It has a simple command line
     interface allowing users to specify  and  use  the  required
     resource.   This interface can be customised by a configura-
     tion file specified by the environment variable ISESSION.

OPTIONS
     On Unix systems iserver options are preceded by ‘-’.  On DOS
     systems iserver options are preceded by ‘/’.

     sb filename
          Boot the named file (same as -sr -ss -si -sc filename).

     si   Verbose mode.
     se   Test the error flag.

     sl linkname
          Use the named resource.

     sr   Reset the root transputer.

     sa   Analyse and peek the root transputer.

     sc filename
          Copy the named file to the link.

     sp n Set peek size to n Kchars.

     ss   Serve the link.

     sm   Enter the session shell.

     sk interval
          Retry connects every interval (seconds).

     sz[1|2]
          Very verbose debug mode (logs all transactions).

     st   Pass all of the following arguments to the booted  pro-
          gram.

     Options and or  arguments  not  recognised  by  iserver  are
     passed to the booted program.

ENVIRONMENT
     The following environment variables are used by iserver:

     TRANSPUTER
          Specifies the resource to used.  May be  overridden  by
          the sl option.

     ICONDB
          Pathname of the connection database file.

     ISESSION
          Pathname of the  session  manager  configuration  file.
          Default is session.cfg.
```

hostmux

```
NAME
     hostmux - host server channel multiplexor

SYNOPSIS
     process (
       stacksize = 16K,
       heapsize = 64K,
       interface (
          input   from_host,
          output  to_host,
          input     in[size],
          output    out[size],
          int  n=size
          )
       ) mux;

     use “hostmux.lku” for mux;
```

DESCRIPTION

The host server channel multiplexor is used where more than one process needs access to host I/O facilities. The multiplexor takes any number of pairs of server channels, and combines them into a single pair that is connected to the host.

The host connections can be connected another multiplexor, giving rise to tree-shaped structures, which may be built up to any level of complexity.

A process accessing the host is referred to as a “client” of the multiplexor. Each client is connected to one “in” channel, and to the “out” channel with the same array index.

The multiplexor receives requests on any “in” channel, passes the request to the host, waits for the reply, and passes the reply to the corresponding out channel. Termination messages (which cause the iserver to terminate) receive special treatment: a termination message is not passed to the host until a total of “n” termination messages are received from clients. Hence the iserver will not terminate until all clients have terminated.

Client channels should not be left unconnected: if they are the iserver will never terminate, because the multiplexor will expect a termination message from the unconnected channel.

The multiplexor is not limited to a fixed size of iserver packet. If a packet is encountered that exceeds the current buffer size, it attempts to allocate a new buffer from the heap.

The multiplexor inputs requests from the clients using a fair ALT. This means that even if there is a continuous stream of requests from one client, requests from any other client are guaranteed to be serviced within a finite time.

EXAMPLE

```
/* loader.cfs
 *
 * This example shows the use of hostmux to
 * provide host services to run860 processes
 * on three processors.
 */

#include “i860tset.cfh”

#include “hardware.cfs”

process ( stacksize=64K, heapsize=64K );

/* process declarations */


val boot_1 “[0].860 “;
val len_1  size(boot_1);

val boot_2 “[1].860 “;
val len_2  size(boot_2);

val boot_3 “[2].860 “;
val len_3  size(boot_3);


process   ( interface ( input HostInput, output HostOutput,
               int flags = NORUN,
               char boot_file[len_1]=boot_1) ) driver_1;

process   ( interface ( input HostInput, output HostOutput,
               int flags = NORUN,
               char boot_file[len_2]=boot_2) ) driver_2;

process   ( interface ( input HostInput, output HostOutput,
               int flags = NORUN,
               char boot_file[len_3]=boot_3) ) driver_3;

process ( interface ( input HostInput,  output HostOutput,
                      input Input[2],   output Output[2],
                        int Size = 2))            mult_1;

process ( interface ( input HostInput,  output HostOutput,
                      input Input[2],   output Output[2],

input   HostInput;
output  HostOutput;

connect mult_1.HostInput to      HostInput;
connect mult_1.HostOutput to     HostOutput;

connect mult_1.Input[0]  to      mult_2.HostOutput;
connect mult_1.Output[0] to      mult_2.HostInput;

connect mult_1.Input[1]  to      driver_1.HostOutput;
connect mult_1.Output[1] to      driver_1.HostInput;

connect mult_2.Input[0]  to      driver_3.HostOutput;
connect mult_2.Output[0] to      driver_3.HostInput;

connect mult_2.Input[1]  to      driver_2.HostOutput;
connect mult_2.Output[1] to      driver_2.HostInput;

/* placement */

use “hostmux.lku” for mult_1;
use “hostmux.lku” for mult_2;
use “run860.lku” for driver_1;
use “run860.lku” for driver_2;
use “run860.lku” for driver_3;

place driver_1 on TTM100_1;
place driver_2 on TTM100_2;
place driver_3 on TTM100_3;

place mult_1 on TTM100_1;
place mult_2 on TTM100_2;

place  HostInput    on host;
```

```
     place  HostOutput   on host;
```

FILES

```
     hostmux.lku
```

SEE ALSO

```
     iserver(1)
```

iocache

NAME

    iocache - host file I/O accelerator

SYNOPSIS

```
     /* io_type */

     val TMB16       16384;
     val TB400       65536;
     val PARAMID     65536;
     val MCP         131072;
     val BIG_IO     0;
     val STD_IO     0;


     /* packet size */
     val DEFAULT_SIZE 0;


     /* cache block size */
     val DEFAULT_BLOCK 16384


     process (
       stacksize = 10K,
       heapsize = 1M,
       interface (
             input   from_host,
             output  to_host,
             input   from_client[n],
             output  to_client[n],
             int    num_chans = n,
             int     io_type = type,
             int     packet_size = size,
             int     block_size = b_size
          )
       ) cache;

     use “iocache.lku” for cache;
```

DESCRIPTION

    The file cache process improves I/O performance by cacheing recently accessed data. The size of I/O blocks is converted up or down to the optimum, depending on the host server and interface hardware being used.

Where many processes are reading the same file concurrently (such as when a group of i860s are booting the same program), I/O speed can be dramatically increased. The problem of the iserver running out of file descriptors is also avoided, as the cache only opens each file only once.

The configuration parameters are as follows: “from_host” and “to_host” are the normal iserver channels to the host. “from_client” and “to_client” are arrays of iserver channels to the clients, or application processes. Note that iocache multiplexes the host channels of several processes in a manner similar to hostmux(2).

“io_type” specifies the type of I/O optimization that is performed. It is an integer as defined in the above value declarations. “packet_size” is used to determine the size of iserver packets when the io_type is BIG_IO - it is ignored for all other I/O types. To obtain I/O compatible with a standard iserver, both io_type and packet_size should be set to zero.

“block_size” determines the size of buffer that is allocated for each file opened by a client. A value of 16384 (16K) is recommended. The space available for buffers is controlled by the heapsize given to the process. In general, as much heap as possible should be given to this process.

If the host is a PC running DOS, then text mode files opened by Transputer processes are not cached. This does not apply to files opened by i860s.

The standard input and output are not cached.

When the same file is opened more than once (eg, by different client processes) the cached data can be shared between processes. For this to happen, both the file name and the open mode must match exactly.

FILES

iocache.lku

SEE ALSO

iserver(1) hostmux(2)

genio

NAME

genio, genio_open, genio_close, genio_read, genio_write, genio_lseek, genio_control, - optimized file I/O library

SYNOPSIS

```
#include <genio.h>

int genio_open( char *name, int mode );

int genio_close( int fd );

int genio_read( int fd, char *buf, int n );

int genio_write( int fd, char *buf, int n );

int genio_lseek( int fd, long int offset, int origin );

void genio_control( int iotype, int param );
```

DESCRIPTION

The generic I/O library implements faster versions of the Inmos C Toolset low level file I/O functions. Several techniques are included that give optimal results on different kinds of hardware.

This is done by allowing a larger iserver packet size (up to 4096 bytes) to be used than normal (512 bytes), or, in the case of the TMB16 motherboard, by DMA direct to the BIOS disk controller routines.

All operations on files opened using genio_open must be performed using the genio library functions and not using the default library functions. Similarly the genio library functions should not be used on files opened using the normal open function.

On multi-processor systems, the server channels should be multiplexed using hostmux.lku and not the occam procedure so.multiplexor. This routine should also be used instead of so.buffer.

The open, close, read, write and lseek routines are used in the same way as the corresponding Inmos routines, taking exactly the same parameters.

By default, the library performs standard Iserver I/O with 512 byte packets. genio_control is used to specify what type of I/O optimizations should be used. It can be called to switch between modes whenever desired: I/O will be performed using the mode selected when the file was opened.

genio_control takes two arguments: an I/O type and a secondary parameter. The parameter is ignored, unless otherwise specified. The following I/O types can be used:

STD_IO	Standard mode (default)
TMB16_IO	Optimize I/O for the TMB16 (NB the calling process must have it’s host I/O links connected directly to the TMB16 interface, NOT through any multiplexors)
TB400_IO	Use enlarged iserver packets optimized for the TB400.
MCP_IO	Use enlarged iserver packets optimized for MCP1000 and MCP500.
BIG_IO	Use enlarged iserver packets of the specified size. The second argument specifies the size of packet that may be used in bytes.

FILES

genio.h genio.lib

SEE ALSO

hostmux(2)

RESTRICTIONS

File descriptors used by routines in the genio library cannot be used with the standard file I/O routines and vice-versa.

The server channels from Transputer processes using the genio routines cannot be multiplexed using so.multiplexor or buffered using so.buffer.

If TMB16-optimized I/O is enabled, the calling process must have its host channels connected DIRECTLY to the TMB16 interface link. No multiplexors or buffers are allowed between the library and the host link. The hostmux process can handle enlarged iserver packets.

Chapter 6

Check Utilities

The check utilities are a set of tools for testing networks of Transputer nodes. The utilities are:

`check`	Probe a Transputer network.
`ckmon`	Transputer hex monitor.
`ftest`	Perform Transputer functional tests.
`load`	Load Transputer node in network.
`mtest`	Perform Transputer memory tests.

Some of the utilities take the output of `check` as their input. For instance, to test the Transputer memory of a network of processors use:

```
% check | mtest -l
```

Similarly, to perform functional tests of the Transputers in a network of processors use:

```
% check | ftest
```

These utilities are documented in the form of Unix `man` pages in the following section.

6.1 Reference

check

```
NAME
     check - Test a Transputer network

SYNOPSIS
     Unix:

     check [ -c4 ] [ -cl ] [ -cr ] [ -cs ] [ -cfb filename ] [ -h
     ] [ -i ] [ -l name ] [ -m filename ] [ -n ] [ -r ] [ -v ] [
     -x ] [ < filename ]

     DOS:

     check [ /c4 ] [ /cl ] [ /cr ] [ /cs ] [ /cfb filename ] [ /h
     ]  [ /i ] [ /l name ] [ /m filename ] [ /n ] [ /r ] [ /v ] [
     /x ] [ < filename ]

DESCRIPTION
     check is a utility which tests a network of Transputer  pro-
     cessors.   It outputs a list of the processors found and the
     connections between them.

     The output of check can be  used  as  the  input  of  mtest,
     ftest,  or  load.  Alternatively, the output from a previous
     run of check can be piped into subsequent runs.

     The host link connection to use is specified by the l option
     or  the  TRANSPUTER  environment  variable  using the syntax
     cmtnlo where m is the SCSI controller number  (or  SCSI  bus
     number),  n  is  the  SCSI target Id, and o is the host link
     number.

OPTIONS
     In Unix, options are preceded by `-`.  In DOS,  options  are
     preceded by ‘/’.

     c4   Read the state of all C004s found.

     cl   Read the state of C004s, long form.

     cr   Reset all C004s found.

     cs   Set all C004s in file piped into check.
```

```
cfb filename
     Use filename as a configuration binary file.

h    Print help page.

i    Information - tells you whats happening.

l name
     Use  this  link,  else  use  TRANSPUTER  environment
     variable.

m filename
     Use filename as a toolset map file.

n    Do not reset the root transputer.

r    Reset the root transputer subsystem.

v    Leave network in virgin reset state.

x    Ignores any file piped in to check.
```

ckmon

```
NAME
     ckmon - Transputer hex monitor

SYNOPSIS
     Unix:

     check | ckmon -0 [ -h ] [ -l name ] [ -a ]

     ckmon -f filename -n [ -h ] [ -l name ] [ -a ]

     DOS:

     check | ckmon /0 [ /h ] [ /l name ] [ /a ]

     ckmon /f filename /n [ /h ] [ /l name ] [ /a ]

DESCRIPTION
     ckmon is a Transputer monitor program.  The first Transputer
     in  the  network  can  be  monitored  by using the 0 option.
     Other Transputers can be monitored using the n option  where
     n  is  the number of a processor as defined in the output of
     check in the file filename as specified using the f option.

     The host link connection to use is specified by the l option
     or  the  TRANSPUTER  environment  variable  using the syntax
     cmtnlo where m is the SCSI controller number  (or  SCSI  bus
     number),  n  is  the  SCSI target Id, and o is the host link
     number.

OPTIONS
     In Unix, options are preceded by ‘-’.  In DOS,  options  are
     preceded by ‘/’.

     0    Monitors root processor.

     n    Monitors processor n.

     f filename
          Use check output filename.

     h    Help page.

     l name
          Use the named link.
```

```
a    Assert Subsystem Analyse.
```

ftest

NAME

ftest - Test processors in a Transputer network

SYNOPSIS

Unix:

```
check | ftest [ -t2 ] [ -t4 ] [ -t8 ] [ -l ]
```

DOS:

```
check | ftest [ /t2 ] [ /t4 ] [ /t8 ] [ /l ]
```

DESCRIPTION

ftest is a utility used in conjunction with check which tests processors in a network of Transputers.

The host link connection to use is specified by the TRANSPUTER environment variable using the syntax cmtnlo where m is the SCSI controller number (or SCSI bus number), n is the SCSI target Id, and o is the host link number.

OPTIONS

In Unix, options are preceded by ‘-’. In DOS, options are preceded by ‘/’.

t2	Test M212s, T2s, T225s only.
t4	Test T414s, T425s only.
t8	Test T800s only.
l	Log progress of testing.

load

```
NAME
     load - Loads files onto a Transputer in a network

SYNOPSIS
     Unix:

     check | load [ -n File1 File2 ] [ -f filename ] [ -i n ] [ -
     l ] [ -h ]

     DOS:

     check | load [ /n File1 File2 ] [ /f filename ] [ /i n  ]  [
     /l ] [ /h ]

DESCRIPTION
     load is a utility used in conjunction with check which loads
     a file or files onto a processor in a Transputer network.

     The host link connection to use is specified  by  the  TRAN-
     SPUTER  environment variable using the syntax cmtnlo where m
     is the SCSI controller number (or SCSI bus number), n is the
     SCSI target Id, and o is the host link number.

OPTIONS
     In Unix, options are preceded by ‘-’.  In DOS,  options  are
     preceded by ‘/’.

     n File1 File2
          Load Transputer n with a number of files in sequence.

     f filename
          Use check command file.

     i n  Set default iserver path to Transputer n.

     l    Log progress of load.

     h    Display this help page.
```

mtest

```
NAME
     mtest - Test memory of Transputers in a network

SYNOPSIS
     Unix:

     check | mtest [ -c ] [ -e Kb ] [ -i iter ] [ -l ] [ -t tp  ]
     [ -t2 ] [ -t4 ] [ -q ] [ -x ] [ -0 ] [ -h ]

     DOS:

     check | mtest [ /c ] [ /e Kb ] [ /i iter ] [ /l ] [ /t tp  ]
     [ /t2 ] [ /t4 ] [ /q ] [ /x ] [ /0 ] [ /h ]

DESCRIPTION
     mtest is a utility used  in  conjunction  with  check  which
     tests the memory of processors in a network of Transputers.

     The host link connection to use is specified  by  the  TRAN-
     SPUTER  environment variable using the syntax cmtnlo where m
     is the SCSI controller number (or SCSI bus number), n is the
     SCSI target Id, and o is the host link number.

OPTIONS
     In Unix, options are preceded by ‘-’.  In DOS,  options  are
     preceded by ‘/’.

     c    Include T2s with C004s on links (default - no).

     E Kb Sets ceiling in Kbytes to which memory is tested.

     i iter
          Number of iterations.

     l    Log progress of testing.

     t tp Test processor tp only.

     t2   Test T2s only.

     t4   Test T4s and T8s only.

     q    Quick memory sizing option.

     x    Extra information on why memory search stopped.
```

`0`	`Do not include root processor in tests.`
`h`	`Display help page.`

Chapter 7

Aserver

The Inmos Aserver based tools (such as `rspy` and the Inquest debugger) can be used with the MatchBox. Set the environment variable `ASERVDB` to point to the supplied Aserver database file `tp/4libs/aservdb`. Please see the Inmos documentation for details of the format of the Aserver database file. The value of the environment variable `TRANSPUTER` is used in conjunction with the Aserver database file to determine the host interface resource to use to access the Transputer network.

The `rspy` utility can be used to determine the link connectivity of the network. For example:

```
% rspy
  # Part-rt  Link0 Link1 Link2 Link3
  0 T805-25   HOST   ...   ...   ...
```

Chapter 8

Configuration

The Transtech PARAstation link connection scheme can be altered by use of link switches which are set by use of the MatchBox configuration link using the `tspconf` command.

A number of pre-defined configuration files are supplied in the `parastn` directory for use with `tspconf` to setup various PARAstation configurations. The Inmos Module Motherboard Software (MMS) can be used to create custom configuration files. See the PARAstation User Manual for details.

Note that due to the way in which the PARAstation configuration system is implemented then `tspconf` can only be used with host link 0 devices with names such as `c0t5l0`.

8.1 Manual Page

tspconf

```
NAME
     tspconf - Transtech SCSI Processor configuration program

SYNOPSIS
     tspconf [ -h ] [ -v ] [ -l name ] [ file ]

DESCRIPTION
     tspconf is a utility used to load a file down the SCSI  Pro-
     cessor  configuration  link.  Typically this is used to con-
     figure a Transtech PARAstation connected to the host  works-
     tation using a Transtech MatchBox.

     The syntax of the name parameter is cmtnl0 where  m  is  the
     SCSI  controller  number  (or  SCSI bus number) and n is the
     SCSI target Id.  The logical unit or  host  link  number  is
     zero.

     If no host link name is specified, the value of the environ-
     ment variable TRANSPUTER is used to specify the host link to
     use.

OPTIONS
     -h    Print help message.

     -v    Verbose mode.

     -l name
          Host link name.

     file Name of file to boot.
```

Chapter 9

TSP Library

The Transtech SCSI Processor library (TSPLIB) is a library of C functions which can be used by workstation programmers to use MatchBox facilities without the need for detailed knowledge of the use of SCSI. The function argument syntax is common to all workstations that support the MatchBox.

The Solaris 2 version of the TSP library, used on Sun SPARCstations, is `lib/libtsp.a` and uses the header file `include/tsplib.h`. C programs using TSPLIB can be compiled and linked as follows:

```
% cc -c -o prog.o prog.c -I$TPS/include
% cc -o prog prog.o -L$TPS/lib -ltsp
```

The TSP library functions are documented in the following section.

tsp_analyse

```
NAME
     tsp_analyse - Reset a host link connection with analyse

SYNOPSIS
     #include <tsplib.h>

     int tsp_analyse( int fd );

ARGUMENTS
     fd   Descriptor returned by tsp_open().

DESCRIPTION
     A call to tsp_analyse() resets the link  and  (with  analyse
     asserted)  the connected devices of the host link connection
     specified by fd.
```

```
RETURN VALUES
     Returns zero on success or -1 on error.
```

tsp_close

```
NAME
     tsp_close - Close a host link connection

SYNOPSIS
     #include <tsplib.h>

     int tsp_close( int fd );

ARGUMENTS
     fd   Descriptor returned by tsp_open().

DESCRIPTION
     A call to tsp_close() closes a connection  to  a  host  link
     that was created by a call to tsp_open().

RETURN VALUES
     Returns zero on success or -1 on error.
```

tsp_error

```
NAME
     tsp_error - Return host link connection error status

SYNOPSIS
     #include <tsplib.h>

     int tsp_error( int fd );

ARGUMENTS
     fd   Descriptor returned by tsp_open().

DESCRIPTION
     A call to tsp_error() returns the state of the error line of
     the host link connection specified by fd.

RETURN VALUES
     Returns zero or one indicating the state of the  error  line
     on success or -1 on error.
```

tsp_mknod

```
NAME
     tsp_mknod - Create TSP device character special files

SYNOPSIS
     #include <tsplib.h>

     int tsp_mknod( char *device )

ARGUMENTS
     device
          Name of host link for which character special files are
          to be created.

DESCRIPTION
     Creates the character special files which  are  required  on
     some  systems to access a Transtech SCSI Processor host link
     using tsp_open().

     The host link is specified by the device parameter which has
     the  format  cmtnlo  where m is the SCSI controller (or bus)
     used, n is the target SCSI id, and o is the link number (0 -
     3).  For example, c0t5l0 specifies link 0 on the target with
     id 5 on SCSI bus 0.

     Note that root privileges are required on  some  systems  to
     use this function.

RETURN VALUES
     Returns zero on success or -1 on error.
```

tsp_open

```
NAME
     tsp_open - Open a host link connection

SYNOPSIS
     #include <tsplib.h>

     int tsp_open( char *device );

ARGUMENTS
     device
          Name of host link to open.

DESCRIPTION
     A connection to  a  host  link  is  created  by  a  call  to
     tsp_open().

     The link to connect to is specified by the device  parameter
     which  has  the format cmtnlo where m is the SCSI controller
     (or bus) used, n is the target SCSI id, and o  is  the  link
     number (0 - 3).  For example, c0t5l0 specifies link 0 on the
     target with id 5 on SCSI bus 0.

RETURN VALUES
     Returns a non-negative descriptor on success or -1 on error.
```

tsp_protocol

```
NAME
     tsp_protocol - Set a host link read protocol

SYNOPSIS
     #include <tsplib.h>

     int tsp_protocol( int fd, int protocol, int block_size );

ARGUMENTS
     fd   Descriptor returned by tsp_open().

     protocol
          Link protocol.

     block_size
          Block size.

     The supported values of protocol are:

     TSP_RAW_PROTOCOL
          Raw byte mode protocol.

     TSP_ISERVER_PROTOCOL
          Iserver mode protocol.

     TSP_BLOCK_PROTOCOL
          Fixed block size protocol.

     The maximum value of block_size is 4096.

DESCRIPTION
     A call to tsp_protocol() sets the  link  protocol  used  for
     host link reads using tsp_read().  This allows the target to
     attempt to perform reads before  they  are  requested  using
     tsp_read() thus improving link bandwidth.

     With raw byte mode protocol, read request lengths correspond
     to  the  number  of  bytes of data to receive.  The value of
     block_size is ignored.

     With iserver protocol, read request lengths must be  set  to
     the value of block_size and correspond to the maximum number
     of bytes to receive in the form of an iserver  packet  which
     consists  of  a  two  byte  count followed by count bytes of
     data.
```

In fixed block size protocol, all reads must of length block_size.

The link protocol is set to raw byte mode when the host link is reset by a call to tsp_reset(), tsp_analyse() or tsp_reset_config(). After being set by tsp_protocol() this protocol persists until the link is reset.

RETURN VALUES

Returns zero on success or -1 on error.

tsp_read

```
NAME
     tsp_read - Read data from a host link

SYNOPSIS
     #include <tsplib.h>

     int tsp_read( int fd, void *data, size_t length,
      int timeout );

ARGUMENTS
     fd   Descriptor returned by tsp_open().

     data Pointer to memory into which data is read.

     length
          The maximum amount of data to be read.

     timeout
          Timeout in seconds.  Zero for no timeout.

DESCRIPTION
     A call to tsp_read() requests  the  target  to  read  up  to
     length bytes of data from the host link.

     By default a raw link protocol is used in which  the  target
     attempts to input length bytes of data (in 4096 byte blocks)
     until a timeout occurs.  If iserver link protocol  is  used,
     an  iserver packet of total length of up to length (or 4096)
     bytes containing a two byte count and count bytes of data is
     input.   In  block link protocol mode, an attempt is made to
     read a block of data of length length.

     The  link   protocol   and   block   size   is   set   using
     tsp_protocol().   In protocol modes other than raw mode, the
     length of data requested must be the same as the block  size
     specified by the function tsp_protocol().

     If a timeout occurs during a read, it is an error to  subse-
     quently  request  a  lesser  amount of data until either the
     original request is satisfied or  until  the  connection  is
     reset using tsp_reset().

RETURN VALUES
     Returns the number of bytes read (which  may  be  less  than
     that requested) on success or -1 on error.
```

tsp_read_config

```
NAME
     tsp_read_config - Read data from the configuration link

SYNOPSIS
     #include <tsplib.h>

     int tsp_read_config( int fd, void *data, size_t length,
      int timeout );

ARGUMENTS
     fd   Descriptor returned by tsp_open().

     data Pointer to memory into which data is read.

     length
          The maximum amount of data to be read.

     timeout
          Timeout in seconds.  Zero for no timeout.

DESCRIPTION
     A call to tsp_read_config() requests the target to  read  up
     to length bytes of data from the configuration link.

     The configuration link is not available for use by the  user
     on Paramid systems.

RETURN VALUES
     Returns the number of bytes read (which  may  be  less  than
     that requested) on success or -1 on error or timeout.

BUGS
     There is a built in timeout of five seconds (per  4096  byte
     block) for this command after which the number of bytes read
     is returned.  If a timeout occurs as specified  by  timeout,
     then  -1  is  returned and the number of bytes read from the
     configuration link is undefined.
```

tsp_reset

```
NAME
     tsp_reset - Reset a host link connection

SYNOPSIS
     #include <tsplib.h>

     int tsp_reset( int fd );

ARGUMENTS
     fd   Descriptor returned by tsp_open().

DESCRIPTION
     A call to tsp_reset() resets the link and connected  devices
     of the host link connection specified by fd.

RETURN VALUES
     Returns zero on success or -1 on error.
```

tsp_reset_config

```
NAME
     tsp_reset_config - Reset the configuration link

SYNOPSIS
     #include <tsplib.h>

     int tsp_reset_config( int fd );

ARGUMENTS
     fd   Descriptor returned by tsp_open().

DESCRIPTION
     A call to tsp_reset_config() resets the  configuration  link
     as well as any devices connected to the host link connection
     specified by fd.

     The configuration link is not available for use by the  user
     on Paramid systems.

RETURN VALUES
     Returns zero on success or -1 on error.
```

tsp_write

```
NAME
     tsp_write - Write data to a host link

SYNOPSIS
     #include <tsplib.h>

     int tsp_write( int fd, void *data, size_t length,
      int timeout );

ARGUMENTS
     fd   Descriptor returned by tsp_open().

     data Pointer to data to write.

     length
          The number of bytes to write.

     timeout
          Timeout in seconds.  Zero for no timeout.

DESCRIPTION
     A call to tsp_write() sends length bytes of data to be  out-
     put on a host link.

     If a write timeout occurs, it is an error to attempt further
     writes  until  the  connection  is  reset using tsp_reset(),
     tsp_analyse() or tsp_reset_config().

     Note that as write buffers are used  to  increase  the  host
     link  bandwidth,  then  tsp_write() may return without error
     before the data is actually output on the host link.  If the
     data cannot be output on the host link then subsequent calls
     to tsp_write() will return an error.

RETURN VALUES
     Returns the number of bytes written  on  success  or  -1  on
     timeout or other error.

BUGS
     There is a built in maximum timeout of five seconds.
```

tsp_write_config

```
NAME
     tsp_write_config - Write data to the configuration link

SYNOPSIS
     #include <tsplib.h>

     int tsp_write_config( int fd, void *data, size_t length,
      int timeout );

ARGUMENTS
     fd   Descriptor returned by tsp_open().

     data Pointer to data to write to the configuration link.

     length
          The amount of data to be written.

     timeout
          Timeout in seconds.  Zero for no timeout.

DESCRIPTION
     A call to tsp_write_config() causes the target to attempt to
     write length bytes of data to the configuration link.

     If a timeout occurs, it  is  an  error  to  attempt  further
     writes  to  the configuration link before resetting it using
     tsp_reset_config().

     The configuration link is not available for use by the  user
     on Paramid systems.

RETURN VALUES
     Returns the number of bytes written  on  success  or  -1  on
     error or timeout.

BUGS
     There is a built in timeout of five seconds (per  4096  byte
     block).
```

Appendix A
SCSI Reference

This appendix describes the SCSI commands supported by the Transtech MatchBox.

The MatchBox is a processor type SCSI device and has eight logical units. Units 0 to 3 correspond to host output (from host) links 0 to 3. Units 4 to 7 correspond to host input (to host) links 0 to 3.

A.1 Host Links

Data is sent to the output host links using `SEND` and received on the input host links using `RECEIVE`. Each logical unit can have one outstanding SCSI command at a time, thus, using disconnects, the MatchBox can be inputting and outputting on all four links concurrently.

Double buffering is used to maximise the link bandwidth - this means that a SCSI `SEND` command may complete before the data is actually sent down the output host link. Various protocols and the block size can be set on input host links so that double buffering can be performed on link input - data may be inputted on a host link before a SCSI `RECEIVE` command has been received.

Input protocol and block size are set and read using the vendor specific commands `WRITE FLAGS` and `READ FLAGS` and apply only to host input links. Host output links use Raw mode. The input protocols currently supported are:

Raw	Input the requested amount of bytes. The block size parameter is ignored.
Iserver	Input a two byte count and count bytes of data. The maximum packet size (count plus data) is set by the block size.

Block	Fixed length blocks.

The maximum value of the block size parameter is 4096 bytes.

A.2 Configuration Link

In addition to the host links the MatchBox has a configuration link on which data can be read or written by performing the vendor specific commands `READ CONFIG` and `WRITE CONFIG`. The configuration link can be reset using the vendor specific command `WRITE FLAGS`.

A.3 Subsystems

The four MatchBox subsystems correspond to the host input and output links and thus are common to logical units 0 and 4, 1 and 5, 2 and 6 and 3 and 7. These are manipulated by using the vendor specific commands `WRITE FLAGS` and `READ FLAGS`.

A.4 Miscellaneous

The MatchBox handles data in blocks of up to 4096 bytes. Commands that have timeouts associated with them are timed with respect to each block.

If a unit attention condition is current, for example after power on or reset, all commands except `INQUIRY` and `REQUEST SENSE` terminate immediately with `CHECK CONDITION` status after which the unit attention condition is cleared. An `INQUIRY` command can be performed returning `GOOD` status without clearing the unit attention condition. A `REQUEST SENSE` command can be performed returning `GOOD` status (if no error occurs) clearing the unit attention condition.

If an attempt is made to use a logical unit whilst it is processing another command then `BUSY` status is returned. This can happen if two initiators attempt to use a logical unit at the same time or if a previous command has been aborted and the MatchBox has not terminated processing it.

For the correct operation of the MatchBox, disconnects should always be enabled on commands which may not complete immediately.

A.5 SCSI Commands

The command descriptor blocks (CDBs) and data formats of SCSI commands supported by the MatchBox are listed in the following pages. Note that the Logical Unit Number field of the command descriptor block is only used if no `IDENTIFY` message is sent by the initiator on selection.

TEST UNIT READY

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (00h)							
1	Logical Unit Number							
2								
3								
4								
5								

A `TEST UNIT READY` command is used to test the readiness of a logical unit on a MatchBox to receive a command. A `GOOD` status is returned if the logical unit is ready, `BUSY` is returned if another command is in progress and `CHECK CONDITION` is returned if it is in a unit attention condition (e.g. after power on or reset). Further information is available by using the `REQUEST SENSE` command.

READ FLAGS

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (02h)							
1	Logical Unit Number							
2								
3								
4	Allocation Length							
5								

A `READ FLAGS` command is used to find the current state of the subsystem error and the link protocol flag. On success, up to five or Allocation Length bytes of data is returned to the Initiator:

Byte \ Bit	7	6	5	4	3	2	1	0
0						SE		
1	Protocol							
2	Block Size (MSB)							
3	Block Size							
4	Block Size (LSB)							

The data returned contains the following fields:

SE	Set to 1 if any nodes in the subsystem error line is set.
Protocol	The Protocol byte is set to the current link protocol as set by `WRITE FLAGS`.
Block Size	The Block Size is set to the current protocol block size as set by `WRITE FLAGS`.

REQUEST SENSE

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (03h)							
1	Logical Unit Number							
2								
3								
4	Allocation Length							
5								

A `REQUEST SENSE` command is used to return sense (error) information to the Initiator.

When an error occurs during the execution of a SCSI command, a status of `CHECK CONDITION` is returned. Further information about the error can be determined from the data returned by performing the `REQUEST SENSE` command immediately following the errant command.

Sense data is also set when a unit attention condition occurs, e.g. after power on or a SCSI bus or bus device reset.

The MatchBox returns up to eighteen or Allocation Length bytes of sense data:

Bit

Byte	7	6	5	4	3	2	1	0
0		Error Code (70h)						
1								
2						Sense Key		
3-6								
7	Additional Sense Length (0Ah)							
8-11								
12	Additional Sense Code							
13	Additional Sense Qualifier							
14-17								

See the SCSI-2 specification for details of the possible values of Sense Key, Additional Sense Code and Additional Sense Qualifier.

WRITE FLAGS

Bit

Byte	7	6	5	4	3	2	1	0
0	Operation Code (06h)							
1	Logical Unit Number							
2								
3								
4	Parameter Length							
5								

A `WRITE FLAGS` command is used to set subsystem lines, reset the configuration link, and set the link protocol. Five bytes are transferred from the Initiator:

Bit

Byte	7	6	5	4	3	2	1	0
0			CLR	LR			SA	SR
1	Protocol							
2	Block Size (MSB)							
3	Block Size							
4	Block Size (LSB)							

The data contains the following fields:

SR	The subsystem reset line is set to the value of SR.
SA	The subsystem analyse line is set to the value of SA.
LR	If set, the input or output link corresponding to the logical unit is reset.
CLR	If set, the configuration link is reset.
Protocol	Sets the current link protocol.
Block Size	Sets the current value of the link block size.

The valid values of the Protocol Field are:

0	Raw mode. This is the default mode.
1	Iserver mode.
2	Block mode.

The SR, SA, Protocol and Block Size fields retain their values until they are either explicitly changed by a subsequent `WRITE FLAGS` command or until a SCSI bus or bus device reset occurs. The LR and CLR bits are self-clearing.

RECEIVE

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (08h)							
1	Logical Unit Number							
2	Allocation Length (MSB)							
3	Allocation Length							
4	Allocation Length (LSB)							
5								

A `RECEIVE` command is used to request up to Allocation Length bytes of data from the input Transputer link. This command is valid only on logical units 4 to 7.

This command will terminate after one second if the requested data (or one 4096 byte block) is not received on the corresponding link. `CHECK CONDITION` status is returned. A subsequent `REQUEST SENSE` command returns sense data with Sense Key set to `ABORTED COMMAND` (8h), Additional Sense Code set to 08h and Additional Sense Code Qualifier set to 01h.

After such a timeout, it is an error for subsequent `RECEIVE` commands to request less data than is outstanding from previous `RECEIVE` commands.

The behaviour of the `RECEIVE` command depends on the Protocol and Block Size as set by `WRITE FLAGS`. In protocols other than raw mode, the value of Allocation Length must be the same as the Block Size as set by `WRITE FLAGS` otherwise the command terminates with `CHECK CONDITION` status.

In raw mode, an attempt is made to input Allocation Length bytes of data from the host link.

In Iserver mode, an attempt is made to input an iserver protocol packet (two byte count plus count data bytes) of up to Allocation Length bytes in total. Allocation Length must equal the protocol Block Size.

In block mode, an attempt to input a block of data of length Allocation Length on the host link is made. Allocation Length must equal the protocol Block Size.

Disconnects are used to minimise SCSI bus usage during this command. If disconnects are disabled and the request cannot be satisfied immediately then the command terminates as in the above timeout condition.

READ CONFIG

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (09h)							
1	Logical Unit Number							
2	Allocation Length (MSB)							
3	Allocation Length							
4	Allocation Length (LSB)							
5								

A `READ CONFIG` command is used to request up to Allocation Length bytes of data from the configuration link.

If not all of the requested data has been received within approximately five seconds, only the data received (if any) is transferred to the initiator. This is not an error.

Do not use this command unless disconnects are enabled.

SEND

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (0Ah)							
1	Logical Unit Number							
2	Transfer Length (MSB)							
3	Transfer Length							
4	Transfer Length (LSB)							
5								

A SEND command is used to send up to Transfer Length bytes of data to the output host link. This command is valid only on logical units 0 to 3.

Write buffering is used so that this command may terminate with GOOD status before the data is actually output on the link. If the data cannot be output on the host link then subsequent SEND commands will terminate with CHECK CONDITION status after approximately five seconds. It is then an error for subsequent SEND commands to be attempted until the link is reset using WRITE FLAGS.

Do not use this command unless disconnects are enabled.

WRITE CONFIG

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (0Ch)							
1	Logical Unit Number							
2	Allocation Length (MSB)							
3	Allocation Length							
4	Allocation Length (LSB)							
5								

A `WRITE CONFIG` command is used to send up to Transfer Length bytes of data to the configuration link.

If the data is not output on the configuration link within approximately five seconds, then the command terminates with `CHECK CONDITION` status. It is an error for subsequent `WRITE CONFIG` commands to be attempted until the configuration link is reset using `WRITE FLAGS`.

Do not use this command unless disconnects are enabled.

INQUIRY

Bit

Byte	7	6	5	4	3	2	1	0
0	Operation Code (12h)							
1	Logical Unit Number							
2								
3								
4	Allocation Length							
5								

An `INQUIRY` command is used to return information about the MatchBox logical unit to the initiator.

The MatchBox returns up to 36 or Allocation Length bytes of `INQUIRY` data:

Bit

Byte	7	6	5	4	3	2	1	0
0				Peripheral Device Type (03h)				
1 - 2								
3					Response Data Format (02h)			
4	Additional Length (20h)							
5 - 6								
7				Sync (1)				
8 - 15	Vendor Identification (`TRANST`)							
16 - 31	Product Identification (`MatchBox`)							
32 - 35	Product Revision Level							

See the SCSI-2 specification for details of the format of the data returned.

SEND DIAGNOSTIC

Bit

		7	6	5	4	3	2	1	0
Byte	0	Operation Code (1Dh)							
	1	Logical Unit Number					SelfTest		
	2								
	3								
	4								
	5								

The `SEND DIAGNOSTIC` command requests the MatchBox to perform diagnostic operations on itself.

The MatchBox responds to this with `GOOD` status if the SelfTest bit is set, otherwise it returns `CHECK CONDITION` status.

WRITE BUFFER

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (3Bh)							
1	Logical Unit Number					Mode		
2								
3	Buffer Offset (MSB)							
4	Buffer Offset							
5	Buffer Offset (LSB)							
6	Transfer Length (MSB)							
7	Transfer Length							
8	Transfer Length (LSB)							
9								

The `WRITE BUFFER` command writes data into the MatchBox target buffer and enables programming of the flash EPROM.

The target buffer is an array of memory on the MatchBox with the same size as the flash EPROM (i.e. 512K). Data can be written to this buffer and its contents can be written to the MatchBox flash EPROM.

Up to Transfer Length bytes of data are transferred to the MatchBox target buffer starting at offset Buffer Offset. The valid binary values of Mode have the following effect:

010b	Write to target buffer.
100b	Write to target buffer.
101b	Write to target buffer and write its contents to flash EPROM.

The MatchBox does not set a unit attention condition if these complete successfully.

READ BUFFER

Byte \ Bit	7	6	5	4	3	2	1	0
0	Operation Code (3Ch)							
1	Logical Unit Number					Mode		
2								
3	Buffer Offset (MSB)							
4	Buffer Offset							
5	Buffer Offset (LSB)							
6	Allocation Length (MSB)							
7	Allocation Length							
8	Allocation Length (LSB)							
9								

The READ BUFFER command reads data from the MatchBox target buffer or returns a description of the buffer.

The target buffer is an array of memory on the MatchBox with the same size as the flash EPROM (i.e. 512K).

With Mode set to 010b, up to Allocation Length bytes of data are transferred from the target buffer starting at offset Buffer Offset.

With Mode set to 011b, a descriptor of up to four or Allocation Length bytes is returned:

Byte \ Bit	7	6	5	4	3	2	1	0
0	Offset Boundary (0)							
1	Buffer Capacity (MSB) (08h)							
2	Buffer Capacity (00h)							
3	Buffer Capacity (LSB) (00h)							

Index