Archimedes TDS Server (D700D)

User Guide

Gnome Computers Limited
25A Huntingdon Street
St. Neots
Cambridgeshire, PE19 1BG

Tel. (0480) 406164

Issue 2

1. About this document

This document describes the Archimedes implementation of the Inmos Transputer Development System (TDS). The system described here is the 'D' release of the TDS, known as IMSD700D or just D700D. The Archimedes TDS is based on the version of the TDS for PC compatible machines and has been designed to behave in the same way as that version wherever possible. The primary reference document for the system should therefore be the book titled **`Transputer Development System`** which is published by Prentice Hall. This document describes the differences between the two systems.

2. Hardware Requirements

An Archimedes 305, 310 or one of the 4-series machines is required to run the TDS. The machine should be fitted with a hard disc, an expansion backplane and one of the Gnome Computers transputer add-on boards. The system will work with a T4 (T400, T414, T425) or T8 (T800, T801, T805) transputer with 2MB or more memory. Very limited operation is possible with only 1MB of memory but this is not recommended for routine use.

The Archimedes should be running either the Arthur 1.2 or RISC OS operating systems.

3. Further Reading

Refer to the User Guide of the Archimedes for details of the operating system and to the User Guide of the transputer add-in board for installation instructions.

4. Installation

The system is supplied on seven 3.5" discs. During installation, these should be placed in drive 0 of the Archimedes. Each disc contains a command file which installs the files on that disc. The discs may be installed in any order. Installation is done at the command line level so you should leave the desktop environment if necessary.

The system is installed in a directory **`tds2`** in the root directory of the drive on which the system is to reside. Normally this will be drive 4 and the installation

commands take this as the default. To install on another drive you must set up a command line variable **drive** as follows.

```
*set drive 5
```

This would cause installation to take place on drive 5. Note that the asterisk is the command prompt and only the underlined characters need be typed, followed by the RETURN key. To install a disc place it in drive 0 and type the following.

```
:0.install
```

You should now see files being copied onto the hard disc. When the copying is complete you should repeat it for the remaining discs. To save typing, each installation command file sets function key 1 to **:0.install** so that you just need press **F1** between discs after the first one has been installed. The whole system occupies nearly 5 MB of disc space.

In addition to the TDS system files, four other files are copied to the **library** directory of the chosen drive. These files are **server14** (the TDS server program), **tprod** (a test program), **FPEmulator** (the floating point emulator) and **keys** (described later). You should ensure that the FPEmulator module is loaded before running the TDS and also that the library directory will be scanned for commands typed at the command line. This can be achieved as follows.

```
rmload :4.library.FPEmulator
lib :4.library
```

These commands may be typed directly or, more conveniently, placed in a **!boot** file which is executed when the machine starts up. Refer to the Archimedes manual for details.

5. Confidence Testing

In order to check that the transputer hardware is functioning correctly, the command **tprod** is provided. Typing this at the command line should result in 5 lines of output similar to that shown below.


```
*tprod
Searching for podule ... Found "TRAM Motherboard"
                              (serial 24) in slot 0
Resetting podule     ... OK
Resetting transputer ... Error line is inactive
Peeking transputer   ... Found 32 bit transputer,
                              link speed 10/20 MHz
Sizing memory        ... Found 2097152 bytes
```

If less than 5 lines appear or if the memory size is different to that which is expected you should check that the hardware is correctly configured.

6. Getting started

The TDS is best run from within a directory specifically created to hold occam programs. If you have done the installation and are just starting to use the system, the next thing to do is to create such a directory. For example:

```
cdir $.occam
dir $.occam
```

Within this directory, various other directories will be created by the server as the TDS runs. The directory **top** contains top level files, the directory **tsr** contains occam source, etc. One directory which will normally have to be created manually is the toolkit directory (**tkt**) which contains toolkit files. To create this directory and copy the default toolkit file into it type the following:

```
cdir tkt
copy $.tds2.system.tkt.toplevel tkt.toplevel ~cq
```

Finally, the TDS server can be started as follows:

```
server14
```

This starts the server which will boot the transputer and enter the TDS. If a new top level file is required this should be introduced on the command line as follows.

```
server14 -t newtop.top
```

If you are new to the TDS, there is a tutorial directory which teaches the basic concepts of the TDS. To use this you should type the following.

```
dir $.tds2.tutor
server14
```

7. Differences between TDS on PC and Archimedes

The major difference between the PC and Archimedes implementations of the TDS occurs in the handling of filenames. In the vast majority of situations, the differences are hidden from the user. For the odd occasion where a problem arises the details of filename mapping are given in Appendix I.

A very useful feature of the Archimedes implementation of the TDS is that the mouse may be used to assist in moving around the folding editor. The mouse cursor is an arrow which moves around the screen as the mouse is moved. This has no effect on the editor's text cursor until a mouse button is pressed. When this occurs, the text cursor is moved to join up with the mouse cursor. If the left mouse button was pressed, an ENTER FOLD keystroke is now generated. If the right button was pressed, an EXIT FOLD keystroke is generated. If the centre button was pressed no keystroke is generated unless the text cursor is on the first or last line of the display in which case either a PAGE DOWN or a PAGE UP keystroke is generated.

A further difference is that the Archimedes screen has 32 rows as opposed to the 25 of a PC.

At present neither the TDS Suspend facility nor the use of the RS423 port are supported. It is intended that future releases of the server software will correct this deficiency.

The server can be interrupted at any time by typing CTRL/C. This will normally result in a message prompting to either reboot the transputer or leave the server. If the transputer has crashed it is possible that no message will appear. In this case, typing CTRL/C again will shut down the server to allow the system to be restarted.

The server requires that the floating point emulator program for the Archimedes be installed.

7.1 Default Command Line Parameters File

When the server program starts up it gets its command line arguments from two places. It first reads them from the first line of the argument file **`$.tds2.args`**. This file need not exist but in practice its presence is normally required. Further arguments are then read from the command line which invoked the server. The argument file may conveniently contain arguments which are normally given while additional or overriding arguments may be given on the command line. The content of the argument file as supplied with the system is as follows:

```
-b $.tds2.system.tdsload.b4 -s #200000
                        -f $.tds2.system.tds2.xsc
```

7.2 Access to Graphics Facilities

It is possible to access the Archimedes VDU drivers directly in order to use the graphics operations, change the screen mode, etc. This is done by placing the keyboard in raw mode by sending a TT_KEY_RAW operation. In this mode all characters passed to the TT_OUT_STRING operation go directly to the Archimedes VDU driver. This state of affairs is reversed by placing the keyboard back in cooked mode with the TT_KEY_COOKED operation. A sample program which accesses the Archimedes graphics facilities is listed in Appendix II.

7.3 Key File

Each transputer board supplied by Gnome Computers has a unique indentification code built in to it. In order for the TDS to run with a particular board, a matching software key must be present in a file known as the key file and normally filed as **`$.library.keys`**. The key is a 32 bit integer which must be specified as a hexadecimal number in the key file. This file should contain only hex numbers, spaces and newlines. It may contain several numbers allowing the TDS Server to operate with any one of a number of transputer boards installed in the machine.

The allocation of keys is made by Gnome Computers and one is normally supplied with each copy of the TDS Server.

Appendix I - File name transformations

Because the version of the TDS used on the Archimedes is PC based, various transformations have to be applied to filenames to make them compatible with the Archimedes filing system. In practice, these will generally be transparent to the user but are described below for when something goes wrong! The server has a debugging mode which prints filenames when a file is opened.

In general, it is best to make all file name references either to the current directory or to the TDS system directory. If this is done, programs should be readily portable between PC and Archimedes. If explicit references to Archimedes directories are made the program will have to be modified to run on a PC.

The problems are as follows:

(1) The Archimedes allows a smaller number of entries per directory than MSDOS

(2) The Archimedes has no concept of a filename extension (eg the '.BAT' in 'AUTOEXEC.BAT')

(3) The Archimedes directory separator is '.' whereas that of MSDOS is '\'

The Archimedes TDS server performs certain translations on filenames in order to get around these problems. The translation begins by looking at the beginning of the filename and the following translations applied.

```
A:      ->   :0.
A:\     ->   :0.
B:      ->   :1.
B:\     ->   :1.
C:      ->   :4.
C:\     ->   :4.
D:      ->   :5.
D:\     ->   :5.
\       ->   $.
```

For example


```
c:dir\file.ext          ->  :4.dir\file.ext
\tds2\iolibs\file.tsr   ->  $.tds2\iolibs\file.tsr
toplevel.top            ->  toplevel.top
```

The next transformation replaces all instances of '\' with '.'.

```
:4.dir\file.ext         ->  :4.dir.file.ext
$.tds2\iolibs\file.tsr  ->  $.tds2.iolibs.file.tsr
toplevel.top            ->  toplevel.top
```

The third transformation involves the extension. If the extension is a 'recognised' extension, the extension and preceding filename component are swapped. Thus:

```
:4.dir.file.ext         ->  :4.dir.file.ext
$.tds2.iolibs.file.tsr  ->  $.tds2.iolibs.tsr.file
toplevel.top            ->  top.toplevel
```

Recognised extensions are as follows:

```
tcm tsr dcd cs1 dds ddb cp1 cu1 ce1
dlk csc cpr cut cex dck dbn dmp tci tai
top b4 tsf xsc tkt
```

When a file is being written on the Archimedes, the server ensures that all necessary directories are created before the file is opened for writing. When such a file is deleted, any directories which were created when it was first written remain intact.

Server debug mode

The debugging mode causes the server to print two filenames and the access mode whenever a file is opened. The first is the name the transputer system is attempting to access, the second is the transformed name that the server generates. The access mode will be r, w, or u for read, write or update. The success or failure of the open operation is also printed.

The server is placed into this mode by an extra parameter on the command line. This is introduced with the -d key which is accompanied by a numeric argument. 0 generates no debug information, 1 prints filenames and 2 prints filenames and then waits for a key press before continuing. Thus a command line might look like:

```
server14 -d 1
```


Appendix II - Simple graphics demonstration

```
#USE userhdr

INT i, char:

PROC plot (VAL INT k, x, y)          – emulate BASIC plot
  [6] BYTE buf:
  SEQ
    buf [0] := BYTE 25
    buf [1] := BYTE (k /\ 255)
    buf [2] := BYTE (x /\ 255)
    buf [3] := BYTE ((x >> 8) /\ 255)
    buf [4] := BYTE (y /\ 255)
    buf [5] := BYTE ((y >> 8) /\ 255)
    screen ! tt.out.string
    screen ! 6
    screen ! buf
:

PROC mode (VAL INT mode)             – sets screen mode
  [2] BYTE buf:
  SEQ
    buf [0] := BYTE 22
    buf [1] := BYTE (mode /\ 255)
    screen ! tt.out.string
    screen ! 2
    screen ! buf
:

SEQ
  screen ! tt.key.raw                – set keyboard raw mode
  keyboard ? char                    – get response byte

  mode (0)                           – set screen mode 0

  SEQ i = 0 FOR 20                   – draw some ellipses
```

```
    INT j:
    SEQ
      j := i * 20
      plot (4, 640, 512)
      plot (4, 215 + j, 512)
      plot (#C5, 640, 512 + j)

  screen ! tt.key.cooked            – set keyboard cooked
  keyboard ? char                   – get response byte

  keyboard ? char                   – wait for a key
```