

## DT Source Overview

Hi Folks,

This paper provides an overview of what each DT source file contains. As is my usual way, you'll find DT is designed using a modular approach. For the most part, the code is quite clean and documented to the level where most programmers can understand its' code flow without too much difficulty.

There are a number of Unix Makefile's for the different systems DT builds on. One day I hope to switch over to the GNU *configure* approach, but I haven't taken the time to do so (yet). For example, Makefile.linux is for Intel/Alpha Linux builds, while Makefile.win32 is for Windows builds.

You'll also find various *pattern\_\** files, generated from the RRD Test Disc, we use to ship with each CD-ROM (I don't think this is true anymore).

The documentation files are there as well, WhatNew\* text files, and HTML files in the html/ directory. The documentation is written in Unix *troff* and requires a program named *devps* available on the UEG production systems for the conversion to Postscript. At some point, I'll be converting this document to Adobe Portable Document Format (.pdf), which is universally accepted on most operating systems. [ *yep, I just ordered my Adobe Acrobat 4.0 for Windows, so this will be one of my last Word docs!* ]

Source File	Description
dt.h	This file contains the common definitions, structures, and function prototypes. It also contains the most of the conditionalization necessary for each operating system.
dt.c	This is the mainline module where everything starts. It contains the parsing functions, mainline test loops, option sanity checking, and signal handlers.
dtaioc.c	This module contains the POSIX Asynchronous I/O functions. These are currently supported on Tru64 Unix and SCO UnixWare only.
dteei.c	The module contains the Tru64 Unix specific EEI support. It contains functions for reporting EEI information, as well as tape reset support.
dtfifo.c	This module provides support for Unix FIFO's (a.k.a. named pipes).
dtgen.c	This module contains the generic I/O functions.
dtinfo.c	This module provides support for platform specific device name information. The functions here are those responsible for setting up the device type and device size.
dtmmap.c	This module provides support for testing memory mapped files.
dtprocs.c	This module contains functions for multiple processes ( <i>procs=</i> and <i>slices=</i> ).
dtread.c	This module contains the generic read/verify functions.
dtstats.c	This module is responsible for reporting per pass and total statistics.
dttape.c	This module contains the various magtape I/O operations for each platform.
dttty.c	This module contains support functions for testing serial lines (speed, parity, etc).
dtusage.c	This module contains the built-in help reported via the " <i>dt help</i> " command.
dtutil.c	This module contains the various utility or support functions used by all files. Here, you will find functions to initialize buffers, perform data comparison, and report errors.
dtwrite.c	This module contains the generic write functions.

I don't suspect you'll actually need to support DT, as I'll still be doing this as its' author. But on the off chance that I'm too busy or not responsive enough, feel free to dive into the source. The Windows port will not be a simple one, since 1) POSIX API's use 32-bit entities (need 64-bit values), 2) native mode API's must be used for asynchronous I/O, and 3) native mode tape API's need implemented.

Personally, the simplest way to learn the code flow, is use SlickEdit or EMACS with appropriate tags files, and pop around to the various functions called.

If necessary, I guess I could do a code walkthrough, if this level of details is desired.