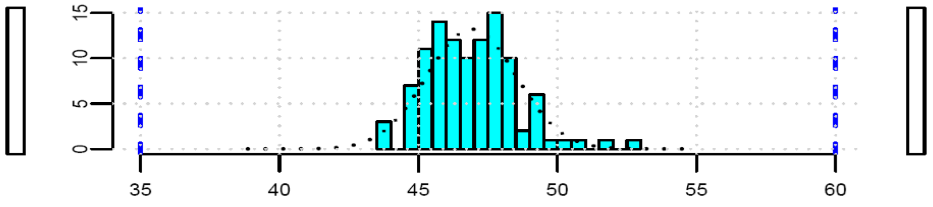
R-based ATE Data Analysis Resources

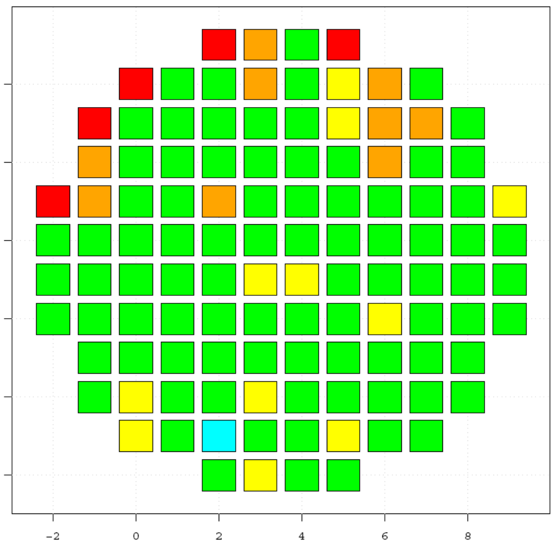
(RADAR)

Documentation

Version 0.6.9

November 2, 2015





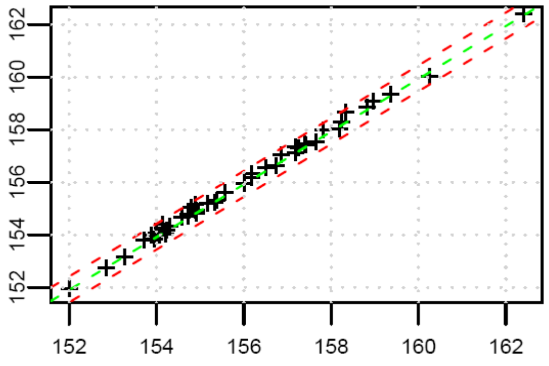


Table of Contents

1 Introduction

Welcome to the RADAR documentation! What is RADAR? An acronym for **R**-based **A**TE **D**ata **A**nalysis **R**esources. It is an open source (GPL) software tool for generating histograms, statistics, wafermaps and more from STDF data files and other related ATE (Automated Test Equipment) generated data formats. To check for the latest version of RADAR or this document, visit http://sites.google.com/site/stdfradar (this site migrated from google pages to google sites on 1 Dec 09) As the acronym suggests, it is written in "R".

1.1 What is R?

R is a statistical computing language/environment. There are quite a few reasons for the selection of R as the language to use when RADAR was started.

- Cost – R is open source and hence free. Some other statistical computing environments can cost in the thousands to tens of thousands of dollars.
- Performance – In benchmark exercises between various statistical packages, R proved to be among the fastest. (www.sciviews.org/benchmark) [this reference is getting a bit old...]
- Platform Support – R is available for a multitude of computer platforms including Linux, Mac, Unix, Windows and others.
- Interpretive Language – This gives the simplicity of not having to compile functions and also gives a level of transparency to the RADAR package. It is easier to verify that there is no malicious code when everything is ASCII source.
- Features – R has a lot of the required functionality already built in, including aspects like:
    - Various statistical commands, including pretty much even the most obscure or advanced for the hard core statistical guru.
    - Support for GUI applications via Tk/Tcl
    - Support for parsing binary and ascii files, including gzipped files, readily enabling one to work on .stdf.gz files directly
    - Various plotting functions
    - Interactive command line for "hands-on" data analysis and manipulation

1.2 What is RADAR?

RADAR is a group of scripts written in R for processing various data formats and converting them to a common RTDF (Radar Test Data Format) format, as well as manipulating data in RTDF format or extracting plots, statistics, or wafermaps from the RTDF data. The primary reason for creating an RTDF format is that the STDF format is not optimal for processing. It is optimized more for facilitating the tester generation of the data. The RTDF file is structured in a way that is easier and much faster for R to process. A good introduction to RADAR is to step through the first tutorial found later in this document, Introductory Tutorial Using The Example STDF. Of course, to be able to do this, you would first have to install RADAR (and R) which is covered in the next section.

1.2.1 What is RTDF?

RTDF (Radar Test Data Format) is a standard Rdata file, which is the binary format that R uses to save workspaces. The .rtdf extension rather than the .Rdata extension is to indicate that the workspace objects in the file are the specific ones that the RADAR scripts work with. The 4 key objects and the other optional objects are as follows:

- ParametersFrame

  This frame contains the per test information, such as test number, test name, scaler for units, units, lower limit, upper limit, plot lower limit, and plot upper limit.

- DevicesFrame

  This frame contains the per device information, such as part id, temperature, x coordinate, y coordinate, wafer index, soft bin, hard bin, test time, and site.

- LotInfoFrame

  This frame contains the per lot information, such as lot id, sublot id, start time, program name, tester type, tester id, and handler

- ResultsMatrix

  This is a pure numerical matrix with the dimensions of Parameters x Devices. It contains all the measurements aka results.

and optionally:

- SbinInfoFrame

  This frame contains the soft bin information, such as the soft bin number, soft bin count, soft bin type (pass or fail), and the soft bin name

- HbinInfoFrame

  This frame contains the hard bin information, such as the hard bin number, hard bin count, hard bin type (pass or fail), and the hard bin name

- WaferInfoFrame

  This frame contains the wafer orientation information if it is available. The fields are: wafer size, die height, die width, wafer units, wafer flat direction, x coordinate for the center of the wafer, y coordinate for the center of the wafer, direction for x going positive, and direction for y going positive. This information is more often than not missing from the incoming data, so the WaferMap script hasn't yet been coded to use this.

- WafersFrame

  This frame contains the per wafer information, such as wafer id, start time, finish time, part count, and good count.

- TSRFrame

  This frame contains the information from the TSR records of the STDF file, the test synopsis records: test number, test name, test type, execution count, and fail count.

- TestFlagMatrix

  This matrix is the same size as the ResultsMatrix, and contains the pass/fail flag status for each test as extracted from the STDF file.

  - 0 = pass
  - 1 = pass alternate limits

- 2 = test failed
- 4 = test flag not valid

If you want to get a better feel for what an RTDF file is, check out the tutorial Looking inside an RTDF file or look at the ConvertCsv section.

1.2.2 STDF Files, RTDF Files and Conditions

One feature on the "like to have" list for STDF files is conditions. For a particular device run, or within a device run, it would be useful to be able to record the conditions that apply for the subsequent tests. One option has been to use DTR records, but some ATEs do not support this approach in that they don't necessarily or consistently put the DTR records into the STDF data stream in the same locations relative to the PTR/FTR/MPR records as they are in the test program. A more universal proposal is to use special PTR or MPR records that the data analysis tools can trigger on. RADAR supports this scheme in the following way: If the "do_conditions" flag is set to TRUE in the ConvertStdf script, then any PTR's or MPR's encountered that have test names that begin with "CONDITION=" are treated as conditions rather than tests. The conditions are then added to the DevicesFrame of the RTDF file. Examples would be tests like "CONDITION=temp" and the value would be something like -40, 25, or +85C, or tests of "CONDITION=Vdda" and "CONDITION=Vddd", with the values being combinations of 3.0V, 3.3V or 3.6V.

An alternate approach to tracking conditions in a test program is to encode conditions into the testnames. An example would be to have most tests done at nominal voltage, and for tests that are repeated at minimum supply voltage, append "_Vmin" to the testname, and similarly, "_Vmax" for tests done at maximum supply voltage. There is a RADAR script available for then extracting this data into multiple separate files, SplitBySubstr.

1.3 What is TkRADAR?

TkRadar is a Graphical User Interface (GUI) based on the Tk/Tcl toolkit that provides a pointy-clicky way of running the various RADAR scripts. It is comprised of windows for executing the various RADAR scripts, and a main window that allows you to launch windows for the various RADAR scripts. The philosophy has been to make most if not all scripts command line, and then create a GUI wrapper. This enables the writing of automation scripts.

One shortcoming of the GUI is that the commands sent to the console do not get included in the console history. As a work around, TkRADAR has the option to generate a TkRadar.log file which provides a timestamp of when a command was executed, as well as the command itself. Also, by default, GUI commands themselves don't show up in the console. Again there is an option to have the GUI print one or more lines of the command to the console when a command is executed. These options are controlled by setting the appropriate variables in your .Rprofile file. The following snippet of the .Rprofile shows the 2 relevant variables:

```
# to change default values, uncomment and adjust the values accordingly:
# ==================================================================
# -- TkRadar_logfile: if "", no log file written,
# -- else write timestamp and RADAR script call to filename specified
# tclvalue(.Radar$.TkRadar.env$TkRadar_logfile) <- "TkRadar.log"

# -- TkRadar_verbose:
# -- if 0, just print RADAR script name to console window
# -- if >0, print up to the first n lines of command to console window
# -- if <0, print whole command, (can be quite long) to console window
tclvalue(.Radar$.TkRadar.env$TkRadar_verbose) <- -1        # default is 1
```

These settings can also be set on a per product (aka Settings file) basis using the Settings Edit Gui which you can invoke by clicking on the "Edit" button in the row just below the Settings File: field in the main TkRADAR window.

2 Installation and Starting RADAR

RADAR is a set of scripts written in the R language, so if you do not have R installed on your computer, then this is the first step. For RADAR version 0.6.3, version 2.8.0 or newer version of R is required. (RADAR version 0.6.2 worked with older versions of R, but the new TkRadarDefaultsGui window uses a Tcl/Tk command that was only introduced into the 2.8.0 version of R. For Linux Ubuntu distributions, this version of R is included in the 9.04 release ("Jaunty").

2.1 Installing R

You can find the R installation package on the R project web page, http://www.r-project.org/ or depending on the flavour of Linux you are using, it may already exist in the standard repositories. Below are instructions for 2 of the many supported operation systems:

2.1.1 For Windows

For a windows based system, you will download the latest install package. In December, 2008, the package was R-2.8.0-win32.exe and had a file size of about 30MByte. Once downloaded, just click on the .exe file to start the install wizard and go with the install defaults.

2.1.2 For Ubuntu (Linux)

The R package is a standard Ubuntu repository package available through synaptic. In synaptic, search for "r statistical" and then scroll down to r-base, or do a quick search for "r-base"

2.2 Installing RADAR

The RADAR scripts are not in an official R package format, so installation is not done like a typical R package. An install script using NSIS has been created for Microsoft windows There is also the more manual (and transparent) approach of downloading the compressed folder and following these instructions to get the package installed and the desktop shortcut created.

The installer.exe.zip and/or the .tar.gz versions of the RADAR package can be downloaded from the RADAR web page (http://sites.google.com/site/stdfradar)

2.2.1 Windows win_installer.exe

One quirk of using Google Sites is that is does not allow download files to have .exe extensions. This is the reason that the installer has been compressed into a ".exe.zip" state on the webpage. You would download the .exe.zip file, uncompress it, and run the installer script.

The script will install the folder of RADAR scripts, setup the .Rprofile file, and create a desktop shortcut.

2.2.1.1 NSIS background

The windows installer was created using NSIS (http://nsis.sourceforge.net/Main_Page) . Three related files have been included in the RADAR package folder:

- RADAR_0v6p7_win_installer_v1.nsi – this is the script that generates the .exe file
- NSIS_instructions.txt – these are notes to myself on what I did to install NSIS and what additional plug-ins I used.

- nsis.Rprofile – this is a copy of the .Rprofile, with some NSIS variable substitution, so that the installer can modify this file at install time to tailor some paths and values to the particular install.

2.2.2 Manual install for all OS'es (RADAR_package_....tar.gz)

To manually install the RADAR package, download the .tar.gz version, then uncompress this folder to whatever location you wish. (In the windows example below, the package was put in a folder called RADAR that is in "My Documents", ie. My Documents\RADAR\RADAR_package_xxx. In the linux example, it was put in ~/RADAR/RADAR_package_xxx) Along with the various R scripts in the RADAR_package_xxx folder, there will also be an example .Rprofile file. Copy this file to the directory where you intend to do your RADAR data analysis work. (normally one directory up, in the RADAR folder) Now, edit this copied .Rprofile file with whatever text editor you wish. [NOTE: If using Windows notepad, you may need to open notepad first and then browse to .Rprofile to open it] The file should look something like:

```
my_dir <- getwd()
setwd("C:/Documents and Settings/David/My Documents/RADAR/RADAR_package_0v6p4")
source("Radar.R")

# to change default values, uncomment and adjust the values accordingly:
#tclvalue(.Radar$.TkRadar.env$default_min_plots_per_page) <- 8
#tclvalue(.Radar$.TkRadar.env$default_use_csv_formulas) <- 1
#tclvalue(.Radar$.TkRadar.env$default_use_OOCalc_csv) <- 1
#tclvalue(.Radar$.TkRadar.env$default_add_normal_curve) <- 1

setwd(my_dir)
rm(my_dir)

TkRadar()
```

You will need to modify the path on the second line (the line starting with "setwd("..." to reflect the location where you installed the RADAR scripts. If on a Linux system, the line would be something like `setwd("/home/david/RADAR/RADAR_package_0v6p4")`. NOTE: On Windows systems, if you cut and paste the path in to this file, you will need to change the back slashes to forward slashes!

2.2.2.1 Desktop Shortcut in Windows

If you are working in a Windows environment, you can now create a desktop shortcut for launching RADAR. You can copy the shortcut found in the same folder as the RADAR scripts and the .Rprofile file to your desktop. You should then update the properties to reflect your installation, the "Target" should reflect the correct path to the Rgui.exe file, and the "Start in" should reflect the location where you decided to place your .Rprofile file, ie the folder you want to work in. They would be something like:

Target:	"C:\Program Files\R\R-2.8.0\bin\Rgui.exe" --sdi --no-save
Start in:	"C:\Documents and Settings\David\My Documents\RADAR"

2.2.2.2 Desktop Shortcut in Ubuntu (Linux)

You can create a desktop shortcut, or copy the example RADAR.desktop file found in the directory that contains all the RADAR scripts to your desktop. It should look something like:

```
#!/usr/bin/env xdg-open

[Desktop Entry]
Version=1.0
Name=RADAR
Type=Application
Terminal=false
Exec=gnome-terminal -e /home/david/RADAR/RADAR.sh
Icon=/home/david/RADAR/RADAR_package_0v6p7/Rlogo4.png
```

The Exec= references a shell script that you need to create, ideally in your RADAR working directory. Alternatively, you can copy the example one from the directory with the RADAR scripts and edit it to reflect your installation. The Icon= references the path to the icon, which will be whatever directory you decided to put the RADAR scripts The RADAR.sh script is as follows:

```
#!/bin/bash
cd /home/david/RADAR
/usr/bin/R --no-save
```

The second line should be the path to your RADAR working directory and the 3rd line should reflect the path to the R command.

2.2.3 Using .Rprofile to customize RADAR

As well as loading the scripts into R and starting the RADAR GUI, the .Rprofile file can also be used to modify some of the default behaviours of TkRADAR. Shown below is an example .Rprofile file:

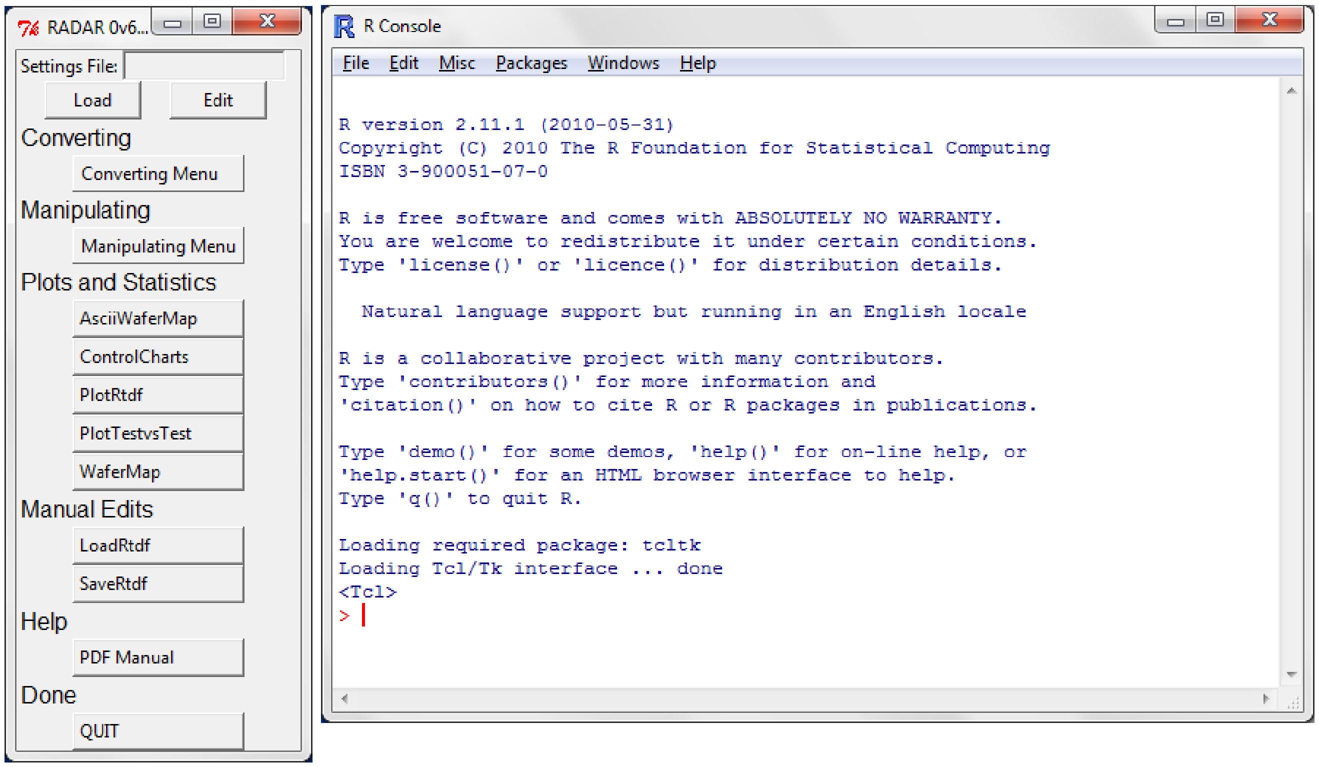
```
##########################################################
#
# .Rprofile for RADAR version 0.6.3  oct/09
#
##########################################################
my_dir <- getwd()
setwd("C:/Documents and Settings/David/My Documents/RADAR/RADAR_package_0v6p3")
source("Radar.R")

tclvalue(.Radar$.TkRadar.env$Orig_dir) <- my_dir
setwd(my_dir)
rm(my_dir)

# to change default values, uncomment and adjust the values accordingly:
# ==================================================================
# -- TkRadar_logfile: if "", no log file written,
# -- else write timestamp and RADAR script call to filename specified
# tclvalue(.Radar$.TkRadar.env$TkRadar_logfile) <- "TkRadar.log"

# -- TkRadar_verbose:
# -- if 0, just print RADAR script name to console window
# -- if >0, print up to the first n lines of command to console window
# -- if <0, print whole command, (can be quite long) to console window
tclvalue(.Radar$.TkRadar.env$TkRadar_verbose) <- -1       # default is 1

# ControlChartsGui defaults
# -----------------------
#tclvalue(.Radar$.TkRadar.env$default_do_western_electric) <- 0
#tclvalue(.Radar$.TkRadar.env$default_control_start_n) <- 1
```

```
#tclvalue(.Radar$.TkRadar.env$default_control_count_n) <- 50
#tclvalue(.Radar$.TkRadar.env$default_control_plots_per_page) <- 7
#tclvalue(.Radar$.TkRadar.env$default_control_landscape) <- 0

# ConvertStdfGui defaults
# -----------------------
#tclvalue(.Radar$.TkRadar.env$default_do_summary) <- 1
#tclvalue(.Radar$.TkRadar.env$default_just_fail_tests_summary) <- 1
#tclvalue(.Radar$.TkRadar.env$default_do_conditions) <- 0

# PlotRtdfGui defaults
# --------------------
tclvalue(.Radar$.TkRadar.env$default_min_plots_per_page) <- 8    # default is 6
#tclvalue(.Radar$.TkRadar.env$default_use_csv_formulas) <- 1
#tclvalue(.Radar$.TkRadar.env$default_use_OOCalc_csv) <- 1
tclvalue(.Radar$.TkRadar.env$default_add_normal_curve) <- 1      # default is 0
#tclvalue(.Radar$.TkRadar.env$default_do_robust_stats) <- 0

# WaferMapGui defaults
# --------------------
#tclvalue(.Radar$.TkRadar.env$default_wmap_xleft) <- 0
#tclvalue(.Radar$.TkRadar.env$default_wmap_ydown) <- 0
#tclvalue(.Radar$.TkRadar.env$default_x_coord_alpha) <- 0
#tclvalue(.Radar$.TkRadar.env$default_panel) <- 0

TkRadar()

#####################################################
```

For various RADAR command GUIs, there are defaults for the various input parameters. If you find you would like the default to be something else, and the parameter you wish to change is in the above file, then you are in luck. Uncomment the line for the associated variable (ie remove the "#" in front of the relevant tclvalue line) and change the value. The most common change is for people using Excel in windows instead of OpenOfficeCalc for their spreadsheet. For the PlotRtdfGui defaults, they uncomment the "use_OOCalc_csv" line and set the default to 0. This generates the comma separated variables (CSV) file in a syntax that Excel expects rather than the OpenOffice syntax when running PlotRtdf. For some defaults, you might want to have different default values depending on the project/product you are working with (ie. Wafer notch orientation in WaferMap). This can be accomplished with different settings files. See the Settings section for more details.

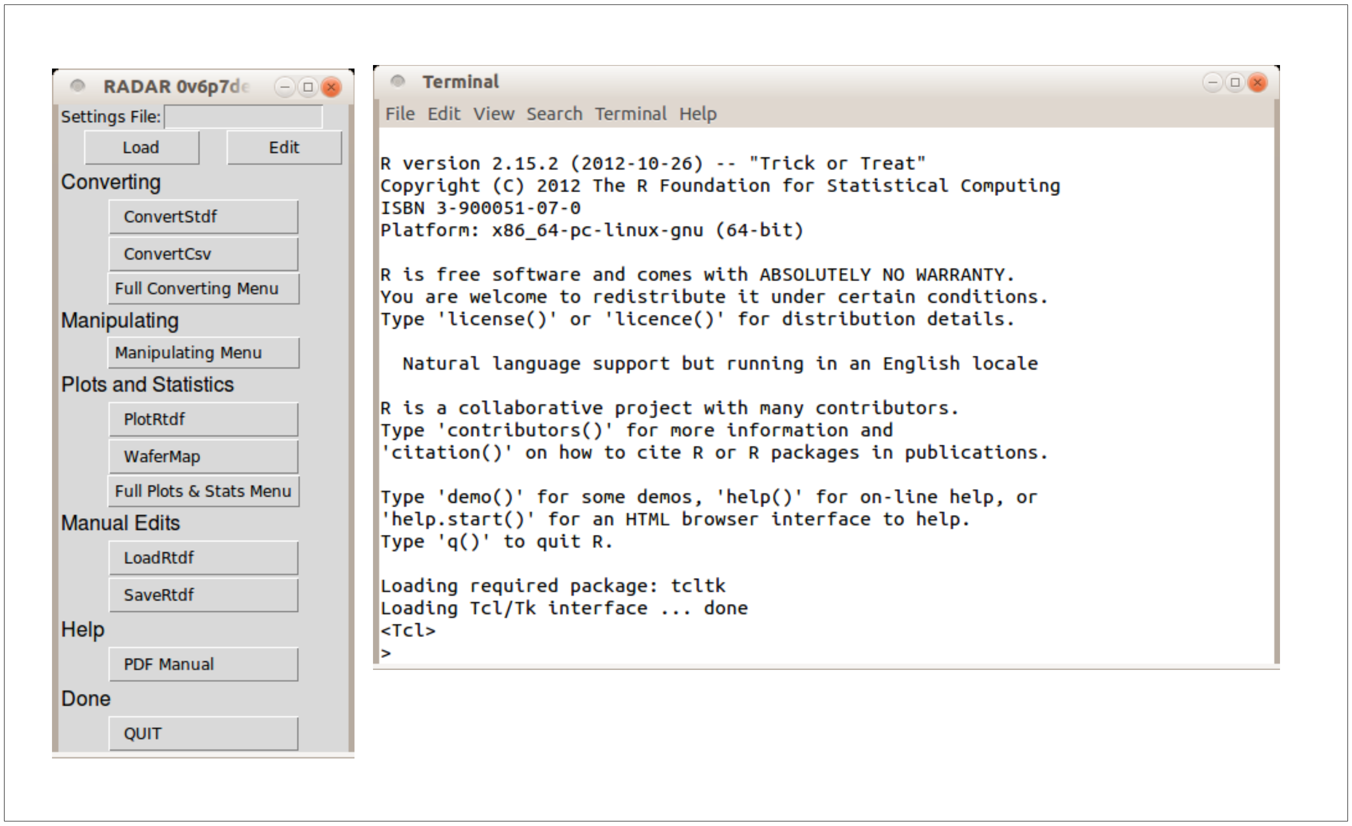
2.3 Starting TkRADAR

If you have done the full installation, then you should now have a shortcut on your desktop labelled RADAR. If so, then just double click this to launch the RADAR Gui. You should now see a pair of windows, the R console and the TkRADAR window.



```
R version 2.11.1 (2010-05-31)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

Loading required package: tcltk
Loading Tcl/Tk interface ... done
<Tcl>
>
```

When you execute TkRADAR commands, they run in the R Console window. The Ubuntu 12.04 windows would look like below:



```
R version 2.15.2 (2012-10-26) -- "Trick or Treat"
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

Loading required package: tcltk
Loading Tcl/Tk interface ... done
<Tcl>
>
```

3 RADAR Scripts

RADAR is composed of quite a few scripts/functions. In the TkRADAR window, these scripts have been organized into the following sections:

– Settings

  This section allows you to create, modify, or load settings files. These files define default directory paths and can also contain defaults variable values to be used by the various scripts. See the Settings section for more details.

– Converting

  Converting includes the scripts that are used to get data into "rtdf", the native data format used in RADAR. It also includes a few scripts that are bidirectional, ones that can convert both into and from "rtdf" format, a script for converting rtdf data into a deducer friendly format, as well as one script that converts STDF to STDF. See the section Converting for more details.

– Manipulating

  Manipulating includes scripts that are used for merging, separating or screening the data. See the section Manipulating for more details.

– Plots and Statistics

  These scripts generate various graphical outputs from the RTDF file or files, including control charts, histograms, normalized probability plots, XY plots and wafermaps. For more details see the Plots and Statistics sections below.

– Manual Edits

  There are 2 scripts here, LoadRtdf and SaveRtdf. LoadRtdf will load an RTDF file into the R workspace. SaveRtdf will save any RTDF objects in the workspace into the specified filename. For more details see the LoadRtdf and SaveRtdf function description sections below.

– Help

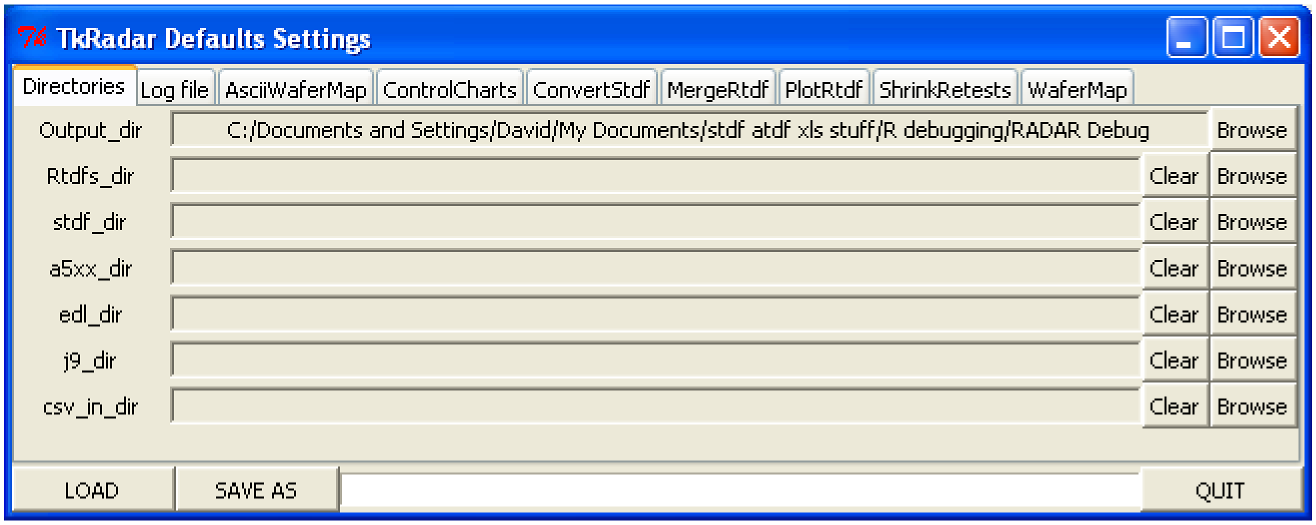
  This section does not invoke any RADAR scripts, but launches this pdf document.

3.1 Settings

At the top of the TkRADAR window is a box showing the most recent settings file that has been loaded, and two buttons, "Load" and "Edit". The "Load" button brings up a menu of the available settings files. The "Edit" button brings up the "TkRadar Defaults Settings" window. This window shows the current values of all the default variables, sorted into tabs based on Directories, Log file, and various Radar scripts such as ControlCharts, ConvertStdf, PlotRtdf, and WaferMap. From this window, you can load, edit, and save settings files.

For most of the tabs, there is an initial column of "Set" checkboxes. If a particular "Set" checkbox is checked, then that default variable will be included in the setting file when you press "SAVE AS".

At the bottom of the window are the "LOAD", "SAVE AS", and "QUIT" buttons. The "QUIT" button closes the window. The "LOAD" button brings up a menu of settings files you have already created and allows you to select and load one of these. The "SAVE AS" button and the string entry box beside it will create a settings file using the name you have entered in the string entry box, and save the various directory and default settings as currently set in the "TkRadar Defaults Settings" window



Output_dir	C:/Documents and Settings/David/My Documents/stdf atdf xls stuff/R debugging/RADAR Debug
Rtdfs_dir	
stdf_dir	
a5xx_dir	
edl_dir	
j9_dir	
csv_in_dir	

The various tabs in detail:

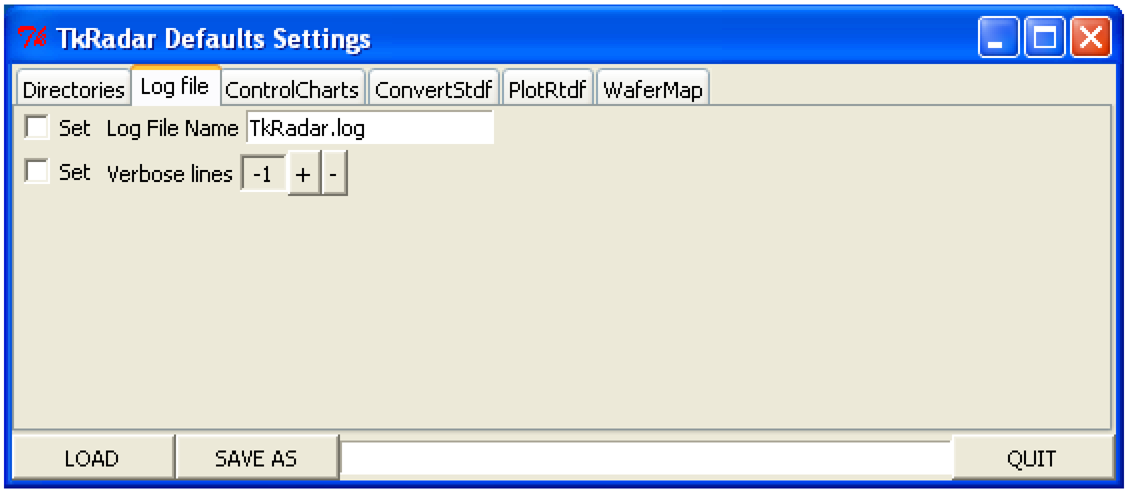
- Directories

  This tab shows the current default folders/directories to be used when browsing for files of the specific type by the various RADAR scripts. If a folder name is set to "", it will default to the Output_dir.

- Log file

  This tab allows you control the logging behaviour of TkRADAR. By default, all the commands executed by TkRADAR are written to a file called TkRadar.log along with a timestamp. You can change the name of the log file, or if you set it to an empty string, logging will be disabled.

  If the "Set" flag is set for this line, then this default value will be included in the settings file you create when you hit the "SAVE AS" button at the bottom of the window. The Verbose lines setting determines how many lines of each command will appear in the R console window when you execute a TkRADAR command. Some commands can get to 10 or more lines. If you want to see all the lines, set the value to -1.

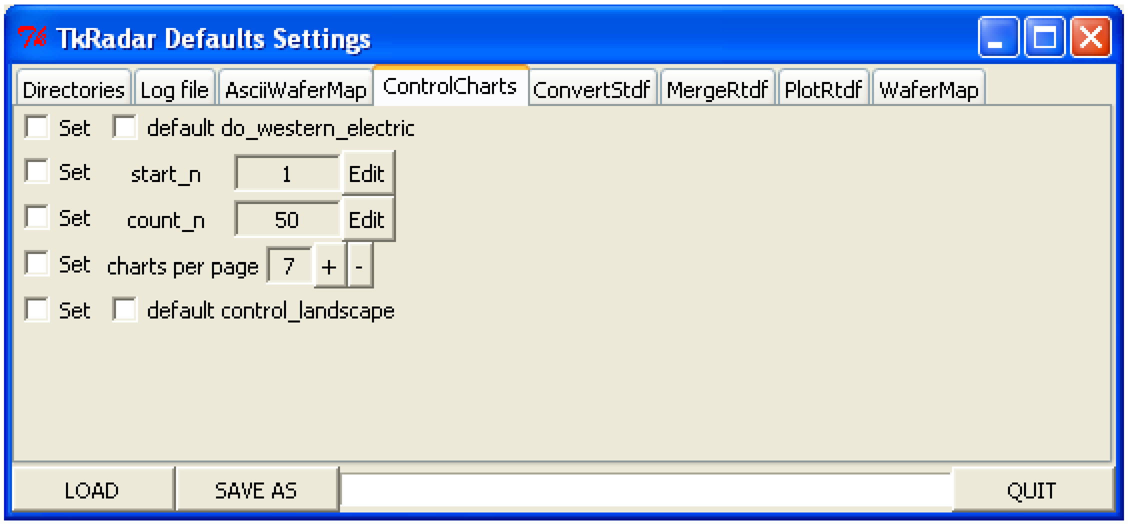


Set Log File Name TkRadar.log

Set Verbose lines -1

- ControlCharts and the rest of the tabs

These tabs contain the defaults variables associated with the TkRADAR script of the same name as the specific tab. The example below is for the ControlCharts tab. The settings are all settings that can be found in the associated script window. For a detailed description of these settings, see the associated window documentation later in this document. As described above in the Log file section, for each default, there is a "Set" flag that can be set. If the flag is set, then this variable will be explicitly stored in the settings file. This allows you to the flexibility to allow specific settings to be controlled either by the .Rprofile file, or overwritten by specific settings files. You may find that you like to use the .Rprofile file to set the majority of the defaults, but that you want product specific settings files for defining the X/Y directions for wafer maps.



☐ Set ☐ default do_western_electric
☐ Set start_n 1
☐ Set count_n 50
☐ Set charts per page 7
☐ Set ☐ default control_landscape

3.2 Converting

There are several scripts that have been organized into the Converting section:

– Convert9470CSV

  This script takes .csv files generated by the HP9470 series of test systems (NOT to be confused with the completely different HP9490/HP94000 systems) and converts them into the rtdf format. See the Convert9470CSVsection for more detail

– ConvetA5xx

  This script takes Teradyne A500, A580, A595 (aka Catalyst) and similar .data ascii files and converts them into the rtdf format. This is useful for those occasions when the stdf file is for some reason not available. See the ConvertA5xx section for more detail.

– ConvertCsv

  This script is primarily used to dump the rtdf file into a format that can be readily pulled into a spreadsheet. This script is bidirectional, in that it can generate a comma separated variable (csv) file that spreadsheets can easily process, or it can take a similarly formatted csv file and generate an rtdf file. This gives users the option to do some of their data manipulation in a spreadsheet if they are more comfortable with that approach. It also gives a defined intermediate ascii file format for users that have data in a format that is not yet directly supported by RADAR. If they don't feel comfortable writing a converter in R, then using Perl or Python or whatever to put the data in this format and calling ConvertCsv may be a pragmatic solution. See the ConvertCsv section for more detail.

- ConvertEagleCSV

  This script takes Eagle tester .log files (csv formatted files) and converts them into rtdf format. For more details see the ConvertEagleCSV section. There is also a ConvertETS script that takes non-csv formatted ascii datalogs. Does the ETS500 generate the csv format and the ETS300 the plain ascii format?

- ConvertEDL

  This script takes HP/Agilent/Verigy 93000 EDL files (the ascii version of the native "Event Datalog" data format) and converts them into rtdf format. This is primarily used when the stdf file is not complete, which occurs if the test program contains test methods with more than 256 tests and the tester is running HPUX. On the newer systems running Linux, this stdf issue has been resolved. (Not really, in Smartest 7.1, limit is 1024 after which 'bad' things happen). For further details, see the ConvertEDL section.

- ConvertETS

  This script takes Eagle tester ascii .log files (NOT csv formatted files) and converts them into rtdf format. For more details, see the ConvertETS section. There is also a ConvertEagleCSV script that takes csv formatted ascii datalogs generated by Eagle test systems.

- ConvertFrugal

  This script takes Frugal datalog files (ascii) and converts them into rtdf format. For more details, see ConvertFrugal

- ConvertHP9490

  This script takes HP9490 or HP94000 .dat files (ascii) and converts them into rtdf format. For more details, see ConvertHP9490

- ConvertJ9

  ConvertJ9 will parse Teradyne J9xx generated text datalogs and convert them into the RADAR native format, rtdf. For more details, see ConvertJ9

- ConvertKDF

  ConvertKDF will parse a Keithley data format file and optionally its associated Keithley limits file and convert them into the RADAR native format, rtdf. For more details, see ConvertKDF

- ConvertParameters

  This script can either generate a CSV version of the ParametersFrame from an RTDF file, or read in a CSV file and generate a sparse RTDF file containing just a ParametersFrame. The PlotRtdf function can use this alternate RTDF file for determining which tests to plot and to which limits. Run this to generate the CSV file, then edit the CSV file to adjust limits (both test limits and plot limits) and remove tests you don't care to see, then rerun the script to convert back to a new RTDF file. For more details, see ConvertParameters.

- ConvertStdf

  This along with PlotRtdf are the 2 main workhorse scripts of the RADAR package. This script will convert STDF files into RTDF files. It tries to adapt to the varying interpretations of the STDF files by the various ATE vendors and test systems so that it "just works". It has been used successfully on a few different systems including Teradyne A530, Teradyne Catalyst, Teradyne J971, LTX Fusion/HF, LTX Fusion/Cx (Unix), LTX Fusion/Mx (Linux), HP/Agilent/Verigy/Advantest 93K, Advantest T2000, and Credence Quartet. For more details, see ConvertStdf.

- ExpandMPRs

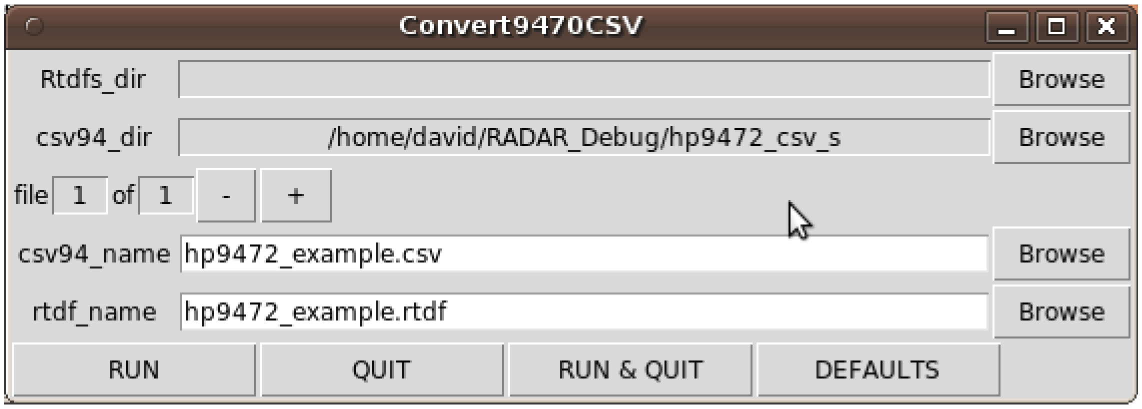
  This script was added to enable another STDF processing tool. That tool was not able to process 93K STDF files directly due to the MPR records. This script flattens out the MPR records into multiple PTR records and creates a new, somewhat larger, STDF file that can now be run in the other tool. For specific details on the script, see ExpandMPRs.

- Rtdf2Deducer

  This script takes the R objects in and RTDF file and combines them into a single object that is more Deducer friendly. For specific details, see Rtdf2Deducer.

3.2.1 Convert9470CSV

This script reads in .csv files that were generated on the HP9472 or similar test systems and puts the data into RTDF format.


Convert9470CSV

Rtdfs_dir		Browse
csv94_dir	/home/david/RADAR_Debug/hp9472_csv_s	Browse
file 1 of 1	- +	
csv94_name	hp9472_example.csv	Browse
rtdf_name	hp9472_example.rtdf	Browse

The various fields in detail:

- Rtdfs_dir

  This field indicates where the RTDF file will be created. This field is greyed out. You can not enter a path here directly, but if you use the browse button, this field will be updated to the location you selected in the browsing.

- csv94_dir

  This field indicates where the .csv file will be found. This field is greyed out. You can not enter a path here directly, but if you use the browse button, or if you use the browse button for the csv94_name, this field will be automatically updated. If the field is empty, then the directory field above will be used as the path to the data file.

- file 1 of 1 -/+

  This window supports converting multiple files with a single click of the RUN button. If you have more than 1 file, this will allow you to scroll through the list of csv94_name's and associated rtdf_name's. To select more than 1 file, click the csv94_name Browse button and either use the control key to select multiple files, or the shift key to select multiple sequential files in the browser window.

- csv94_name

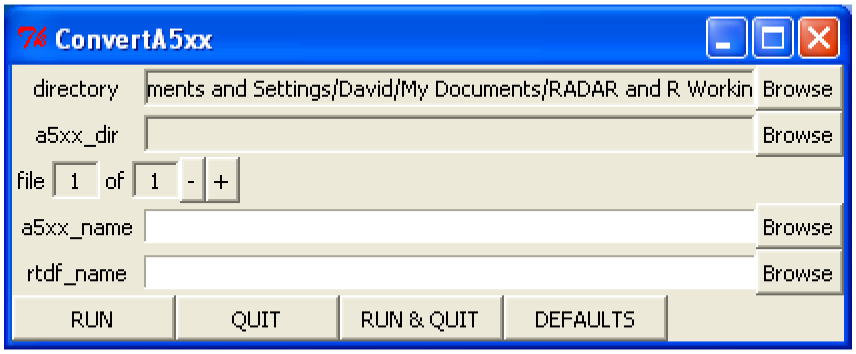
  This is the name of the data file you wish to convert into an RTDF file.

- rtdf_name

  This is the name of the RTDF file that you wish to create. NOTE: When you enter a name in the csv94_name field, the rtdf_name field will automatically update to reflect this name but with a .rtdf extension.

3.2.2 ConvertA5xx

This script reads in Teradyne A500/A580/Catalyst ".data" ascii files and puts the data into RTDF format.



directory: ments and Settings/David/My Documents/RADAR and R Workin

file 1 of 1

The various fields in detail:

- directory

  This field indicates where the RTDF file will be created. This field is greyed out. You can not enter a path here directly, but if you use the browse button, this field will be updated to the location you selected in the browsing.

- a5xx_dir

  This field indicates where the .data or .data.gz file will be found. This field is greyed out. You can not enter a path here directly, but if you use the browse button, or if you use the browse button for the a5xx_name, this field will be automatically updated. If the field is empty, then the directory field above will be used as the path to the data file.

- file 1 of 1 +/-

  This window supports converting multiple files with a single click of the RUN button. If you have more than 1 file, this will allow you to scroll through the list of a5xx_name's and associated rtdf_name's. To select more than 1 file, click the a5xx_name Browse button and either use the control key to select multiple files, or the shift key to select multiple sequential files in the browser window.

- a5xx_name

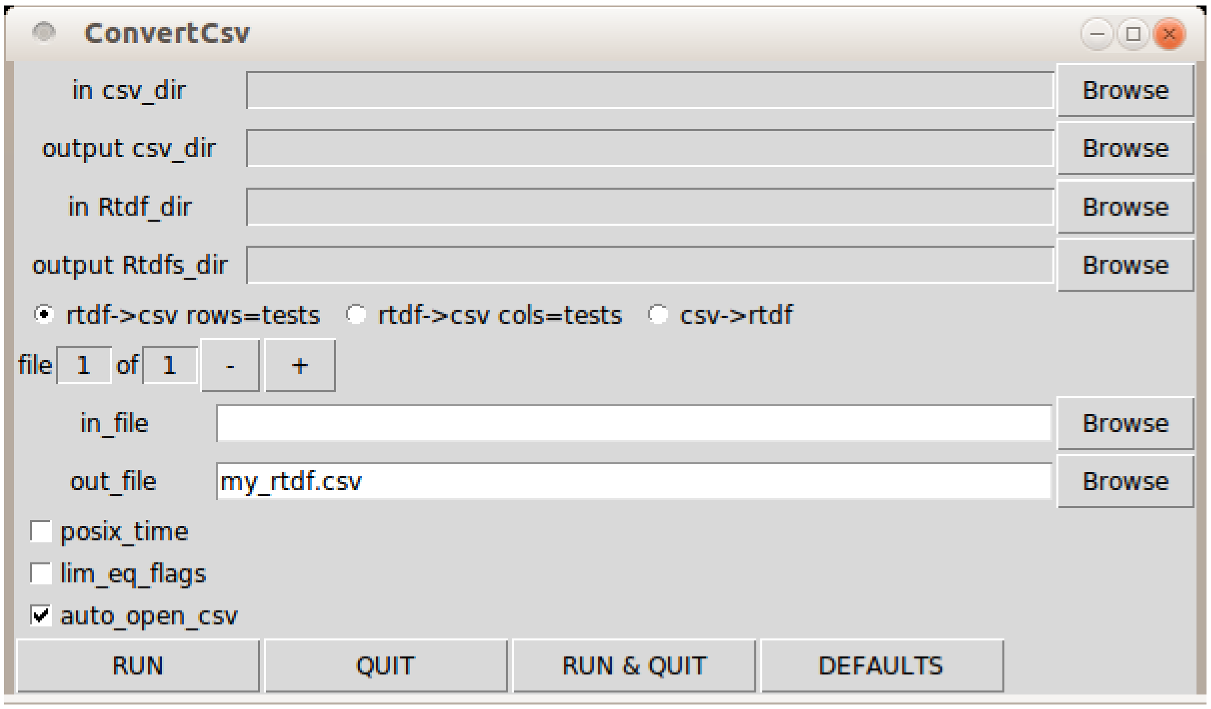
This is the name of the data file you wish to convert into an RTDF file. NOTE: the file can be gzipped and the conversion script will work just fine. ie .data or .data.gz files both work.

- rtdf_name

  This is the name of the RTDF file that you wish to create. NOTE: When you enter a name in the a5xx_name field, the rtdf_name field will automatically update to reflect this name but with a .rtdf extension.

3.2.3 ConvertCsv

This script will translate an RTDF file into a CSV format, or conversely, take a properly formatted CSV file and convert it into an RTDF file.



ConvertCsv

in csv_dir:

output csv_dir:

in Rtdf_dir:

output Rtdfs_dir:

[✓ rtdf->csv rows=tests] rtdf->csv cols=tests csv->rtdf

file 1 of 1

in_file:

out_file: my_rtdf.csv

☐ posix_time

☐ lim_eq_flags

☑ auto_open_csv

The various fields in detail:

- in csv_dir – if converting CSV to RTDF, the directory where the CSV file(s) will be found
- output csv_dir – if converting RTDF to CSV, the directory where the CSV file(s) will be written
- in Rtdf_dir – if converting RTDF to CSV, the directory where the RTDF file(s) will be found
- output Rtdfs_dir – if converting CSV to RTDF, the directory where the RTDF file(s) will be written
- radio buttons for conversion direction
- file 1 of 1 +/-

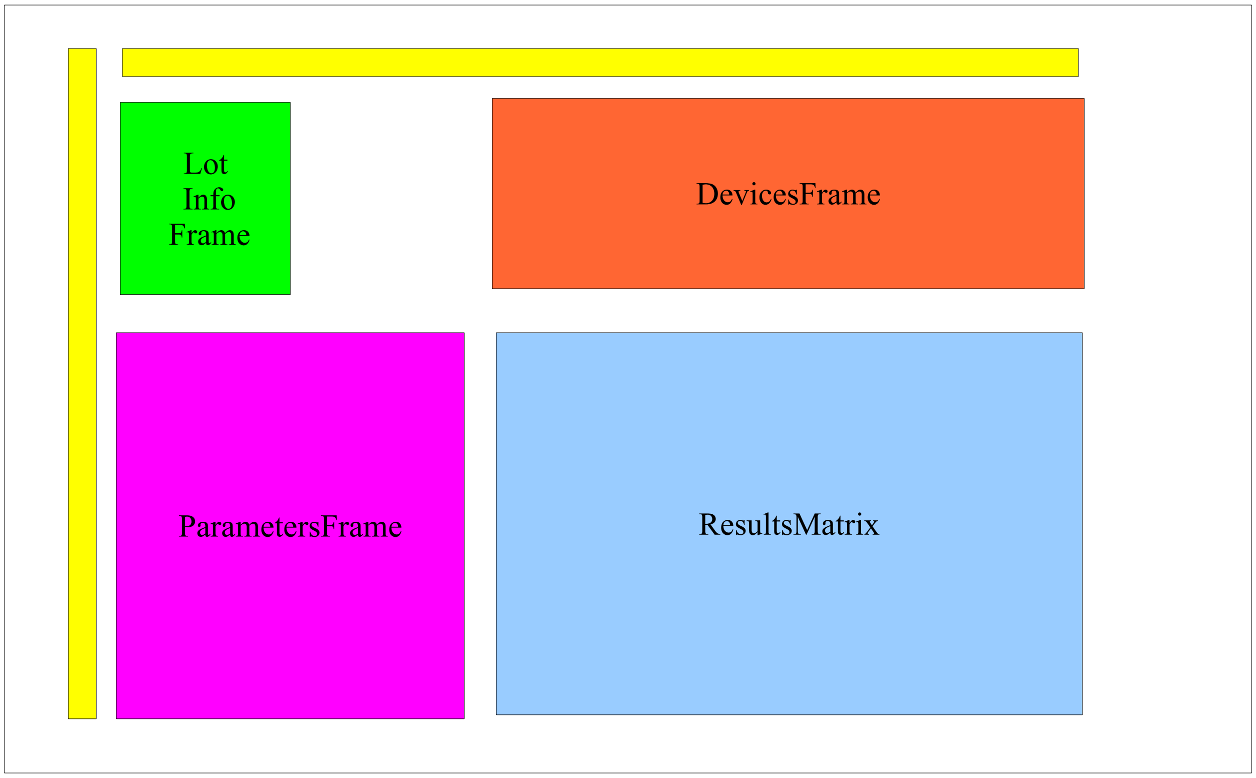
  This window supports converting multiple files with a single click of the RUN button. If you have more than 1 file, this will allow you to scroll through the list of in_file's and associated out_file's. To select more than 1 file, click the in_file Browse button and either use the control key to select multiple files, or the shift key to select multiple sequential files in the browser window.

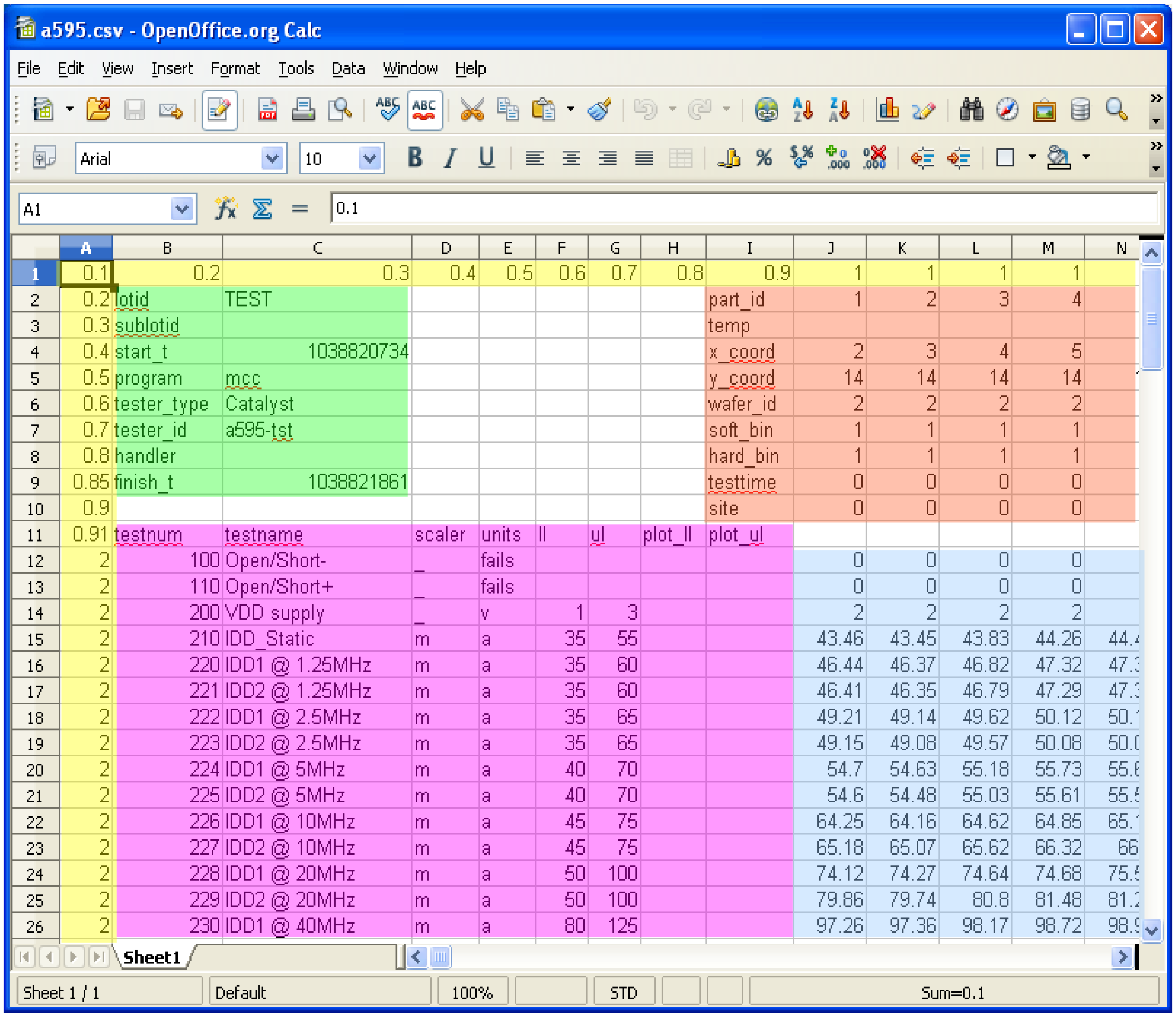
- in_file
- out_file
- posix_time – the start_t and finish_t fields will be in seconds from 1970. If this is unchecked, the time will be converted to an ascii string, ie 2013-09-21 20:15:45 EDT
- lim_eq_flags – support for the flag that indicates if the limit is < or <= has been added. If you want these extra columns, you would check this box. If you have scripts built around the original ConvertCSV output, you can leave this box unchecked to get the original format.
- auto_open_csv – If this flag is set, once the script has finished generating the CSV file, The default application for CSV files will be invoked to open this file.

3.2.3.1 ConvertCsv CSV file

An example CSV version of the RTDF file is shown below. The ConvertCsv function also supports the transform of this table, ie rows become columns, columns become rows. If you are using OpenOffice 2.0, or a version of Excel prior to Office2007, you have a restriction of 256 rows. Depending on your data set, you may find you need to use the transform of the table to get it to fit in the older versions of the spreadsheets.

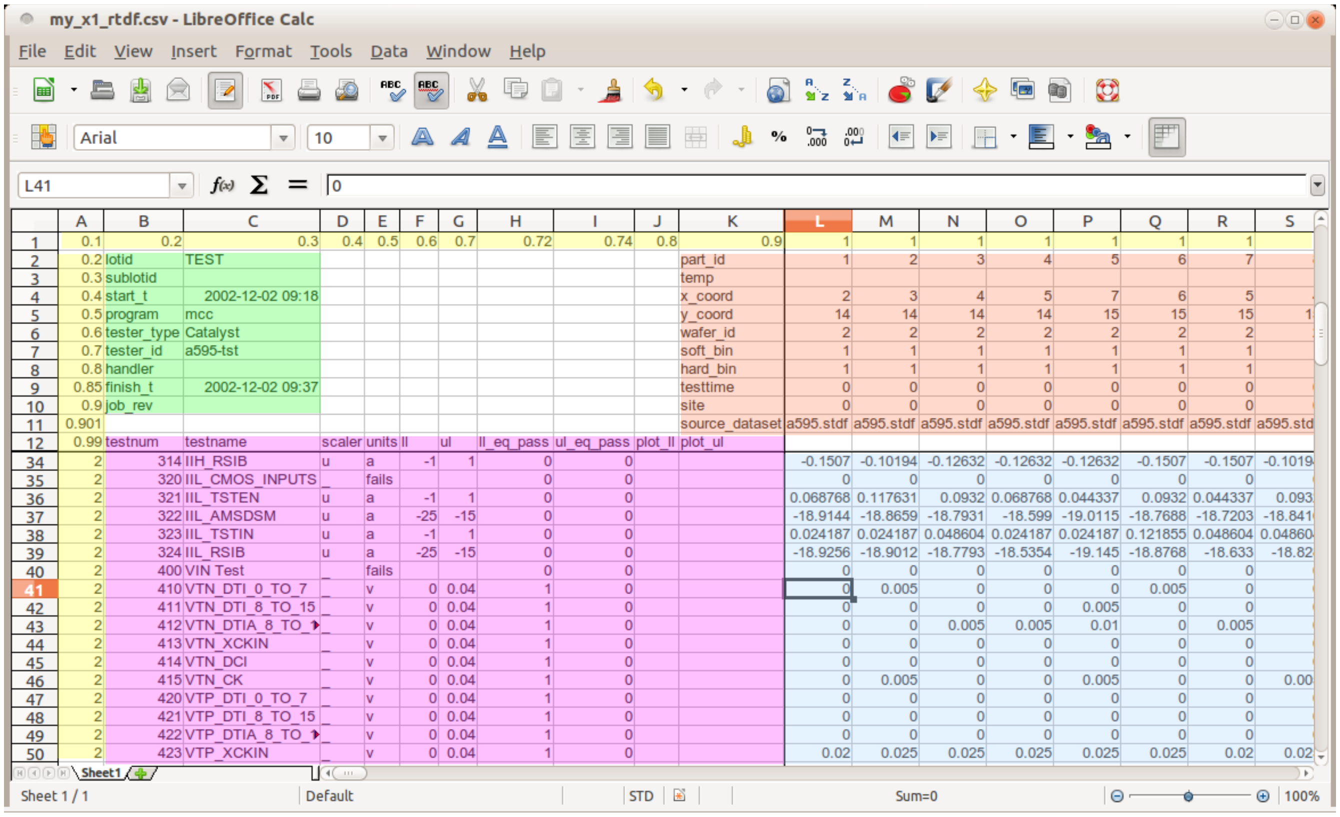
The first row and column are used to separate the actual measurements from the headers, per device information and the per test information when sorting the table in the spreadsheet. If you sort by these first, then just the data rows or columns will be sorted, which is probably what you wanted. The diagram below shows which portion of the CSV file correspond to which of the 4 objects of the RTDF file; LotInfoFrame, DevicesFrame, ParametersFrame, and ResultsMatrix. For files that have extra conditions in the DevicesFrame, this block is bigger, and the ParametersFrame and ResultsMatrix end up being shifted down the same number of rows as the number of extra conditions.




a595.csv - OpenOffice.org Calc
File Edit View Insert Format Tools Data Window Help
Arial
10
A1
0.1
Sheet1
Sheet 1 / 1
Default
100%
STD
Sum=0.1

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1	1	1	1	
2	0.2	lotid	TEST						part_id	1	2	3	4	
3	0.3	sublotid							temp					
4	0.4	start_t	1038820734						x_coord	2	3	4	5	
5	0.5	program	mcc						y_coord	14	14	14	14	
6	0.6	tester_type	Catalyst						wafer_id	2	2	2	2	
7	0.7	tester_id	a595-tst						soft_bin	1	1	1	1	
8	0.8	handler							hard_bin	1	1	1	1	
9	0.85	finish_t	1038821861						testtime	0	0	0	0	
10	0.9								site	0	0	0	0	
11	0.91	testnum	testname	scaler	units	ll	ul	plot_ll	plot_ul					
12	2	100	Open/Short-	_	fails					0	0	0	0	
13	2	110	Open/Short+	_	fails					0	0	0	0	
14	2	200	VDD supply	_	v	1	3			2	2	2	2	
15	2	210	IDD_Static	m	a	35	55			43.46	43.45	43.83	44.26	44.4
16	2	220	IDD1 @ 1.25MHz	m	a	35	60			46.44	46.37	46.82	47.32	47.3
17	2	221	IDD2 @ 1.25MHz	m	a	35	60			46.41	46.35	46.79	47.29	47.3
18	2	222	IDD1 @ 2.5MHz	m	a	35	65			49.21	49.14	49.62	50.12	50.1
19	2	223	IDD2 @ 2.5MHz	m	a	35	65			49.15	49.08	49.57	50.08	50.0
20	2	224	IDD1 @ 5MHz	m	a	40	70			54.7	54.63	55.18	55.73	55.6
21	2	225	IDD2 @ 5MHz	m	a	40	70			54.6	54.48	55.03	55.61	55.5
22	2	226	IDD1 @ 10MHz	m	a	45	75			64.25	64.16	64.62	64.85	65.1
23	2	227	IDD2 @ 10MHz	m	a	45	75			65.18	65.07	65.62	66.32	66
24	2	228	IDD1 @ 20MHz	m	a	50	100			74.12	74.27	74.64	74.68	75.5
25	2	229	IDD2 @ 20MHz	m	a	50	100			79.86	79.74	80.8	81.48	81.2
26	2	230	IDD1 @ 40MHz	m	a	80	125			97.26	97.36	98.17	98.72	98.9

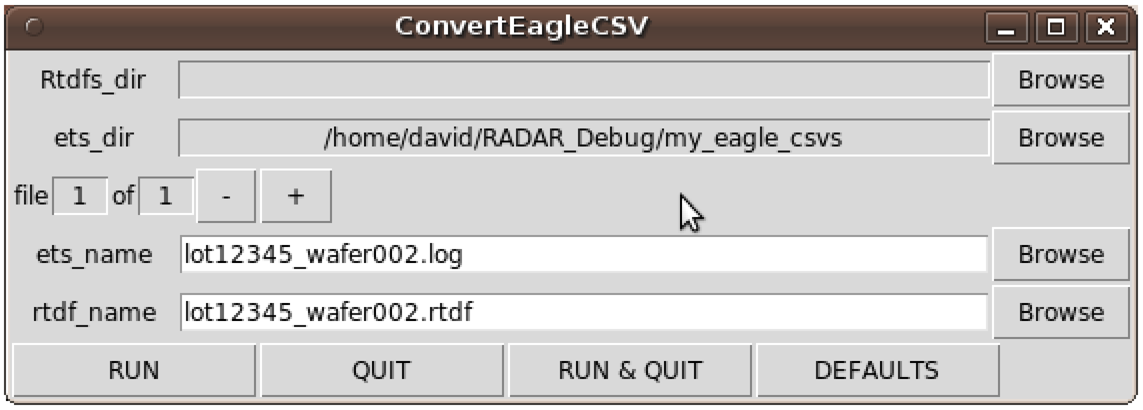
Another, newer example is below, showing the additional (optional) ParametersFrame limits flags, the addition of the job_rev field to the LotInfoFrame, the start_t and finish_t fields of the LotInfoFrame in ascii format, and the source_dataset field in the DevicesFrame:


my_x1_rtdf.csv - LibreOffice Calc
File Edit View Insert Format Tools Data Window Help
Arial
10
L41
0
Sheet1
Sheet 1 / 1
Default
STD
Sum=0
100%

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.72	0.74	0.8	0.9	1	1	1	1	1	1	1	
2	0.2	lotid	TEST								part_id	1	2	3	4	5	6	7	
3	0.3	sublotid									temp								
4	0.4	start_t	2002-12-02 09:18								x_coord	2	3	4	5	7	6	5	
5	0.5	program	mcc								y_coord	14	14	14	14	15	15	15	1
6	0.6	tester_type	Catalyst								wafer_id	2	2	2	2	2	2	2	
7	0.7	tester_id	a595-tst								soft_bin	1	1	1	1	1	1	1	
8	0.8	handler									hard_bin	1	1	1	1	1	1	1	
9	0.85	finish_t	2002-12-02 09:37								testtime	0	0	0	0	0	0	0	
10	0.9	job_rev									site	0	0	0	0	0	0	0	
11	0.901										source_dataset	a595.stdf	a595.stdf	a595.stdf	a595.stdf	a595.stdf	a595.stdf	a595.stdf	a595.std
12	0.99	testnum	testname	scaler	units	ll	ul	ll_eq_pass	ul_eq_pass	plot_ll	plot_ul								
34	2	314	IIH_RSIB	u	a	-1	1	0	0			-0.1507	-0.10194	-0.12632	-0.12632	-0.12632	-0.1507	-0.1507	-0.1019
35	2	320	IIL_CMOS_INPUTS	_	fails			0	0			0	0	0	0	0	0	0	
36	2	321	IIL_TSTEN	u	a	-1	1	0	0			0.068768	0.117631	0.0932	0.068768	0.044337	0.0932	0.044337	0.093
37	2	322	IIL_AMSDSM	u	a	-25	-15	0	0			-18.9144	-18.8659	-18.7931	-18.599	-19.0115	-18.7688	-18.7203	-18.841
38	2	323	IIL_TSTIN	u	a	-1	1	0	0			0.024187	0.024187	0.048604	0.024187	0.024187	0.121855	0.048604	0.04860
39	2	324	IIL_RSIB	u	a	-25	-15	0	0			-18.9256	-18.9012	-18.7793	-18.5354	-19.145	-18.8768	-18.633	-18.82
40	2	400	VIN Test	_	fails			0	0			0	0	0	0	0	0	0	
41	2	410	VTN_DTI_0_TO_7	_	v	0	0.04	1	0			0	0.005	0	0	0	0.005	0	
42	2	411	VTN_DTI_8_TO_15	_	v	0	0.04	1	0			0	0	0	0	0.005	0	0	
43	2	412	VTN_DTIA_8_TO_1	_	v	0	0.04	1	0			0	0	0.005	0.005	0.01	0	0.005	
44	2	413	VTN_XCKIN	_	v	0	0.04	1	0			0	0	0	0	0	0	0	
45	2	414	VTN_DCI	_	v	0	0.04	1	0			0	0	0	0	0	0	0	
46	2	415	VTN_CK	_	v	0	0.04	1	0			0	0.005	0	0	0.005	0	0	0.00
47	2	420	VTP_DTI_0_TO_7	_	v	0	0.04	1	0			0	0	0	0	0	0	0	
48	2	421	VTP_DTI_8_TO_15	_	v	0	0.04	1	0			0	0	0	0	0	0	0	
49	2	422	VTP_DTIA_8_TO_1	_	v	0	0.04	1	0			0	0	0	0	0	0	0	
50	2	423	VTP_XCKIN	_	v	0	0.04	1	0			0.02	0.025	0.025	0.025	0.025	0.025	0.02	0.02

3.2.4 ConvertEagleCSV

This script is used to convert datalog files generated by Eagle testers that are in a CSV format into RTDF files.



Rtdfs_dir	
ets_dir	/home/david/RADAR_Debug/my_eagle_csvs
file	1 of 1
ets_name	lot12345_wafer002.log
rtdf_name	lot12345_wafer002.rtdf

The various fields in detail:

- Rtdfs_dir

  the folder/directory where the generated rtdf files will be written
- ets_dir

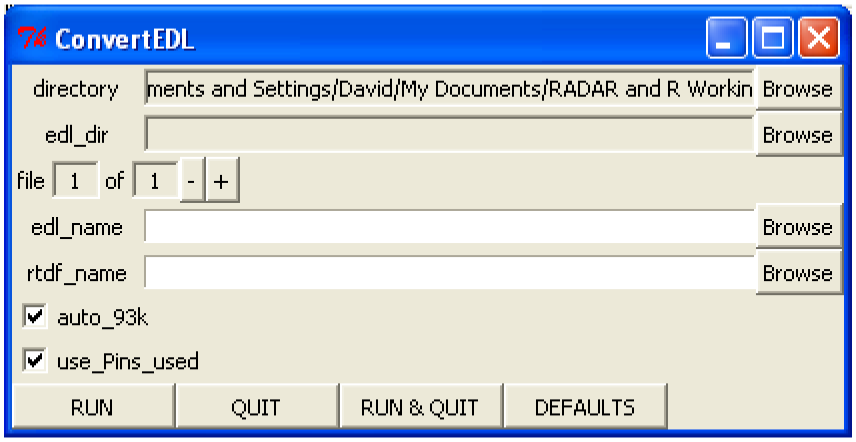
  the folder/directory to look in for the Eagle data files. If this is empty, it will use the same folder as where the rtdf files are being written
- file 1 of 1 -/+

  This window supports converting multiple files with a single click of the RUN button. If you have more than 1 file, this will allow you to scroll through the list of ets_name's and associated rtdf_name's. To select more than 1 file, click the ets_name Browse button and either use the control key to select multiple files, or the shift key to select multiple sequential files in the browser window.
- ets_name
- rtdf_name

  This is/are the name(s) of the rtdf file(s) that will be generated. When a new ets_name is entered, a matching rtdf_name is automatically filled in with the .log extension removed and a .rtdf extension added.

3.2.5 ConvertEDL

This script is used to convert HP/Agilent/Verigy ascii datalog files (ascii output from Event formatter) (sometimes referred to as EDL files or EDF files) into RTDF files.


ConvertEDL

directory: ments and Settings/David/My Documents/RADAR and R Workin [Browse]

edl_dir: [Browse]

file 1 of 1 [-] [+]

edl_name: [Browse]

rtdf_name: [Browse]

[✓] auto_93k

[✓] use_Pins_used

The various fields in detail:

- directory
- edl_dir
- file 1 of 1 +/-

  This window supports converting multiple files with a single click of the RUN button. If you have more than 1 file, this will allow you to scroll through the list of edl_name's and associated rtdf_name's. To select more than 1 file, click the edl_name Browse button and either use the control key to select multiple files, or the shift key to select multiple sequential files in the browser window.

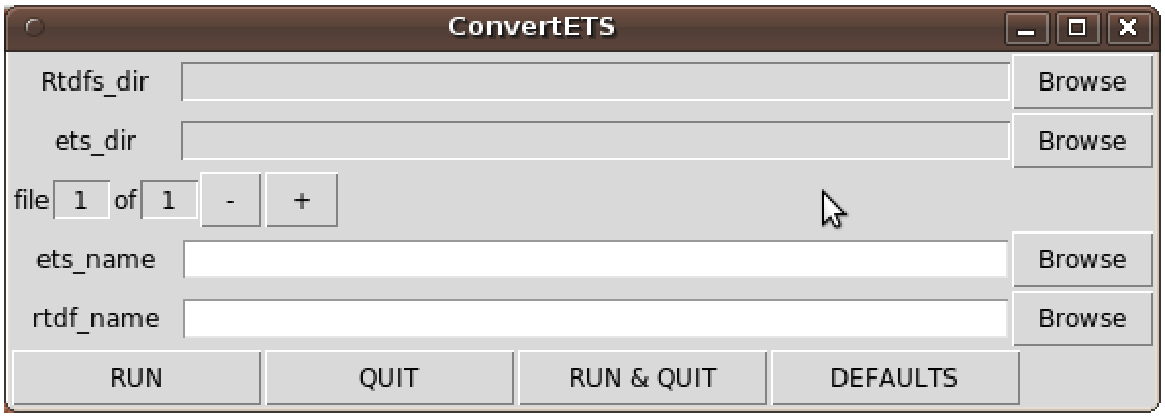
- edl_name
- rtdf_name
- auto_93k checkbox

  this enables the auto-correction features of the conversion script. In some cases, the 93K data file will have positive limits when they should be negative, or have a limit of 0 when there shouldn't be a limit. This feature searches for these occurrences and fixes them.
- use_Pins_used checkbox

  If the "Pin Results" section doesn't have pin names, setting this flag will cause the script to use the names from the "Pins used" line, otherwise it will create generic values no_pin1, no_pin2, no_pin3, ... to keep test names unique.

3.2.6 ConvertETS

This script is used to convert datalog files generated by Eagle testers that are in plain text format into RTDF files.



The various fields in detail:

- Rtdfs_dir

  the directory where the generated rtdf file(s) will be placed
- ets_dir

  the directory where the Eagle ascii datalog files are to be found
- file 1 of 1 +/-

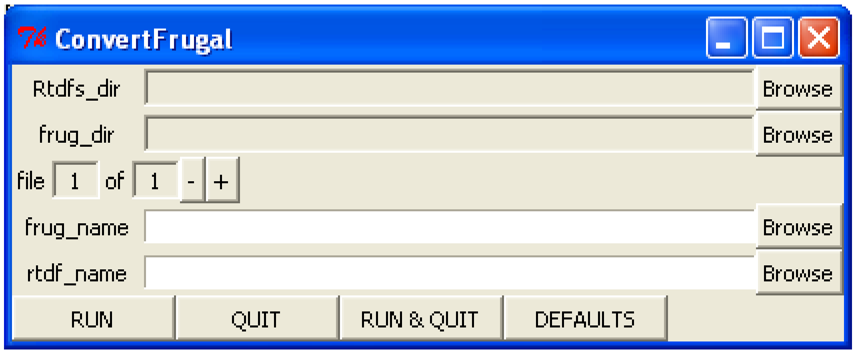
  if multiple files were selected, this allows you to step through the list
- ets_name

  the name of Eagle datalog file that will be processed
- rtdf_name

  the name to assign to the rtdf file that will be generated by processing the input datalog file

3.2.7 ConvertFrugal

This script is used to convert ascii datalog files from the Frugal series tester into RTDF files.



The various fields in detail:

- Rtdfs_dir

  the directory where the generated rtdf file(s) will be placed
- frug_dir

  the directory where the frugal datalog files are to be found
- file 1 of 1 +/-

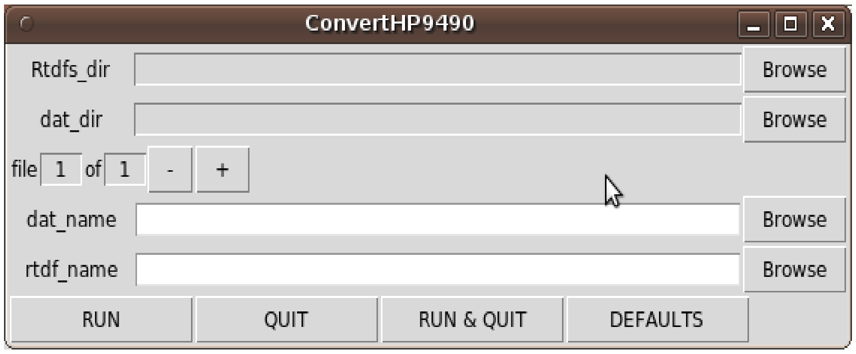
  if multiple files were selected, this allows you to step through the list
- frug_name

  the name of frugal datalog file that will be processed
- rtdf_name

  the name to assign to the rtdf file that will be generated by processing the input datalog file

3.2.8 ConvertHP9490

This script is used to convert HP9490/HP94000 series tester ascii datalog files into RTDF files.



The various fields in detail:

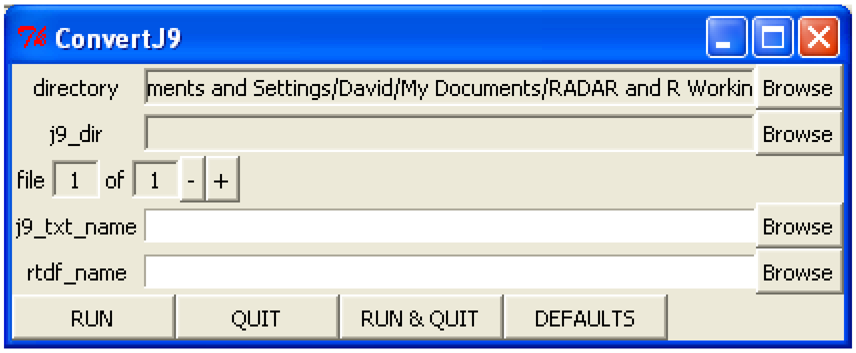
- Rtdfs_dir – the directory where the generated RTDF file will be placed.

- dat_dir - the directory where the HP9490 ascii datalog file(s) can be found
- file 1 of 1 +/-

  This window supports converting multiple files with a single click of the RUN button. If you have more than 1 file, this will allow you to scroll through the list of dat_name's and associated rtdf_name's. To select more than 1 file, click the dat_name Browse button and either use the control key to select multiple files, or the shift key to select multiple sequential files in the browser window.
- dat_name
- rtdf_name

3.2.9 ConvertJ9

This script is used to convert Teradyne J9 series tester ascii datalog files into RTDF files. The ParametersFrame testnames are the testnames in the ascii file with the associated testnumbers appended to them. This is to guarantee unique testnames.



directory: ments and Settings/David/My Documents/RADAR and R Workin

file 1 of 1

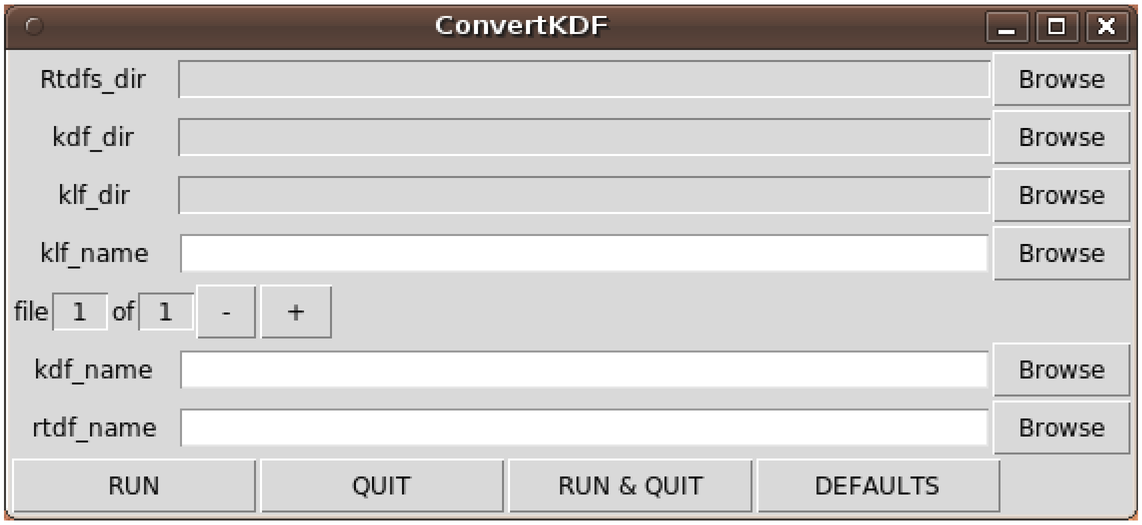
The various fields in detail:

- directory
- j9_dir
- file 1 of 1 +/-

  This window supports converting multiple files with a single click of the RUN button. If you have more than 1 file, this will allow you to scroll through the list of j9_txt_name's and associated rtdf_name's. To select more than 1 file, click the j9_txt_name Browse button and either use the control key to select multiple files, or the shift key to select multiple sequential files in the browser window.
- j9_txt_name
- rtdf_name

3.2.10 ConvertKDF

This script is used to generate RTDF formatted data from Keithley .kdf and .klf files.



The various fields in detail:

- Rtdfs_dir

  The directory that the RTDF file(s) will be created in if different than your working directory
- kdf_dir

  The directory to look in for the KDF (Keithley Data Format) file(s) to convert if different than your working directory.
- klf_dir

  If you have a KLF (Keithley Limits Format) file, the directory to look in for this file if different than your working directory
- klf_name

  If you have a limits file you would specify it here
- file 1 of 1 -/+

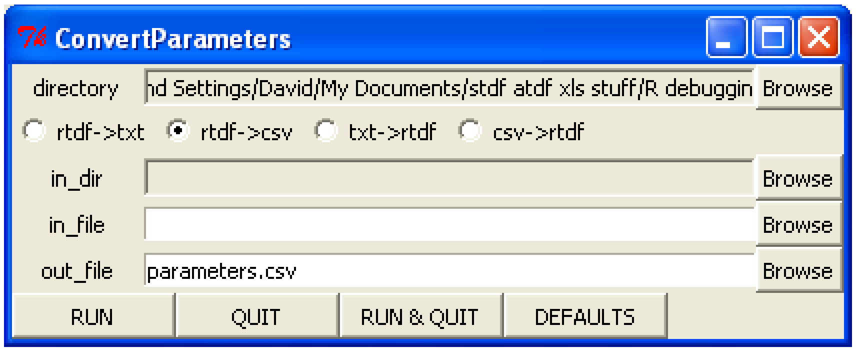
  This window supports converting multiple files with a single click of the RUN button. If you have more than 1 file, this will allow you to scroll through the list of kdf_name's and associated rtdf_name's. To select more than 1 file, click the kdf_name Browse button and either use the control key to select multiple files, or the shift key to select multiple sequential files in the browser window.
- kdf_name

  The name of the KDF file you want to convert into RTDF
- rtdf_name

  The name you want the converted file to have. The tool tries to automatically generate names by replacing the .kdf with .rtdf, but you can always replace this name on a file by file basis

3.2.11 ConvertParameters

This script is used to extract the ParametersFrame from an RTDF file and put it into either a text file or a CSV file. For most users, editing the Frame as a CSV file in a spreadsheet is preferred to doing the editing directly in R. Once the edits have been done, the same script can be used to convert the CSV or text file back into an RTDF file. NOTE: the RTDF file will only contain a ParametersFrame, it will not have any of the other RTDF objects, ie there will be no data. The PlotRtdf script supports using an alternate RTDF file's ParametersFrame for deciding which tests to plot and what limits to use. This script is used to generate that alternate RTDF file.


ConvertParameters

directory: nd Settings/David/My Documents/stdf atdf xls stuff/R debuggin

( ) rtdf->txt [✓ rtdf->csv] ( ) txt->rtdf ( ) csv->rtdf

in_dir:

in_file:

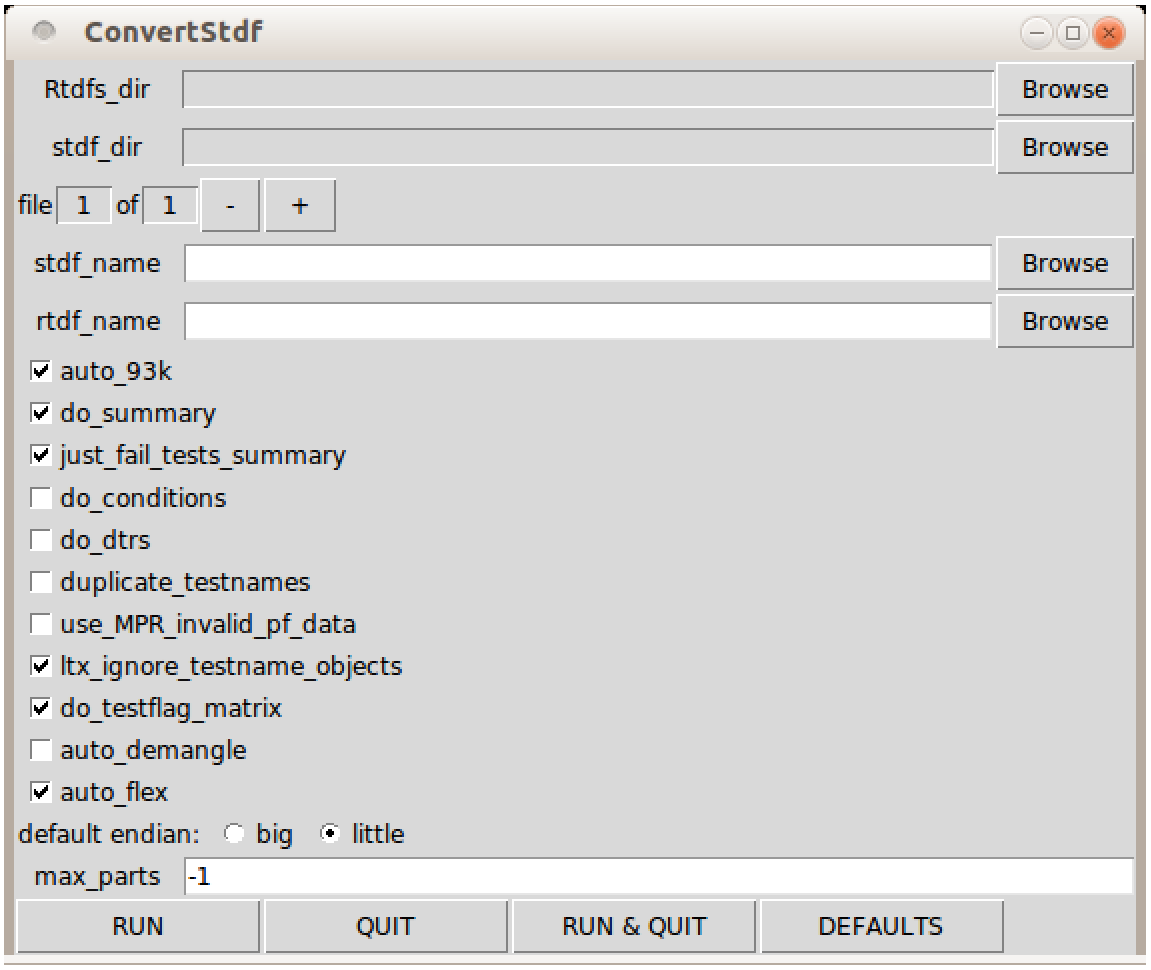
out_file: parameters.csv

The various fields in detail:

- directory
- conversion direction radio buttons
- in_dir
- in_file
- out_file

3.2.12 ConvertStdf

The ConvertStdf script is used to convert files in STDF format or gzipped STDF format into the native RTDF format used by the various RADAR manipulation and plotting scripts.


ConvertStdf

Rtdfs_dir		Browse
stdf_dir		Browse
file 1 of 1	-	+
stdf_name		Browse
rtdf_name		Browse

- [✓] auto_93k
- [✓] do_summary
- [✓] just_fail_tests_summary
- [ ] do_conditions
- [ ] do_dtrs
- [ ] duplicate_testnames
- [ ] use_MPR_invalid_pf_data
- [✓] ltx_ignore_testname_objects
- [✓] do_testflag_matrix
- [ ] auto_demangle
- [✓] auto_flex

default endian: big [✓ little]

max_parts -1

RUN QUIT RUN & QUIT DEFAULTS

The ConvertStdf GUI is shown above. It provides a point and click front end to the ConvertStdf script. When selecting an STDF file using the browse button, it will automatically generate the RTDF name for you by removing the stdf or std or stdf.gz end and appending .rtdf. Inputs specific to the GUI and not the underlying ConvertStdf script:

– Rtdfs_dir

  The directory where the ConvertStdf script will be run, aka where the output files will be put.

– stdf_dir

  The directory where the script will search for the stdf file(s) to process.

– file 1 of 1 -/+

  This window supports converting multiple files with a single click of the RUN button. If you have more than 1 file, this will allow you to scroll through the list of stdf_name's and associated rtdf_name's. To select more than 1 file, click the stdf_name Browse button and either use the control key to select multiple files, or the shift key to select multiple sequential files in the browser window.

The ConvertStdf script takes the following inputs:

- stdf_name – the name of the STDF file you wish to convert into RTDF
- rtdf_name – the name of the RTDF file you wish to create.
- auto_93k - if the stdf is from HP/Agilent/Verigy 93K, try to auto fix broken limits. There are 2 types of silliness that can occur in HPUX vintage files, negative continuity limits shown as positive, and empty limits shown as zero. If the the median measurement is outside the limits but inside the limits if we multiply the limits by -1, then do so. If the upper limit is less than the lower limit and the upper limit is 0, then remove the upper limit, else if the lower limit is 0, then remove the lower limit. For vintage Smartest 7.1 stdf files, if a part fails in an FTR record and the pin count of the pattern modulo 8 = 0, then the record will be corrupted with an extra byte, this flag will allow auto-detection and correction of this bug.
- do_summary – setting this flag to TRUE will cause ConvertStdf to generate a text .summary file as well as the .rtdf file. The summary file will contain lot information extracted from the MIR record of the stdf file as well as the binning and per test statistics as extracted from the SBR and TSR records respectively. It will also build a summary based on the PRR records.
- just_fail_tests_summary – if the do_summary flag is set, setting this flag will suppress per test statistics for tests that had no failures in the .summary file. This is useful in situations where you have lots of tests (500+) and only about 20 or so that actually fail.
- endian - initial guess at type of endianness of the stdf, either big or little, if stdf has FAR record, the FAR record will determine correct endian value. This is only critical if processing version 3 stdf files. (or some Flex stdf files that seem to have DTR records before the FAR record!)
- stdf_dir – if the stdf file is in a different directory, then put the absolute path here.
- do_conditions – enables the script to treat some PTR/MPR records as conditions rather than tests; PTR/MPR records that have test_txt fields that begin with "CONDITION=" will get added to the DevicesFrame rather than the ParametersFrame. For a given device (PIR/PRR group), if a condition is redefined, the script will start a new DeviceFrame device. If you had an STDF file with 40 devices, and in each device run you had 3 occurrences of "CONDITION=Clock_Freq" with values of 66MHz, 100MHz, and 133Mhz, the resulting RTDF file would have 120 devices
- duplicate_testnames – ConvertStdf uses testnames. If you have multiple different tests that have the same testname, the latest one will overwrite earlier ones in a single device run. To avoid this, set this flag to true, which will then append "_" and the testnumber to the testnames to hopefully generate unique testnames
- use_MPR_invalid_pf_data – A number of the Credence testers generate MPR records with the invalid pass/fail data flag set. Usually, this is immediately followed by another MPR record with the exact same information but with valid pass/fail information. As such, the default for this flag is to just ignore these records. On occasion, depending on how the programmer wrote their test program, the data in these records may be different, and possibly meaningful. In this situation, you would set this flag.
- ltx_ignore_testname_objects – LTX / enVision has an option where the testflow object and name can be appended to the testname. If this option has been enabled but was not wanted, this flag will try to remove it by looking for the first occurrence of "/" in a testname and removing everything afterwards.
- do_testflag_matrix – This flag enables the creation of a second matrix of the same size as the ResultsMatrix called TestFlagMatrix with the pass/fail flag data put in a format similar to ATDF. 0 = pass, 1 = pass alternate limits, 2 = fail, 4 = no pass/fail indication, and NaN where

the test was not run for that device.

- auto_demangle – This flag enables the detection and attempted repair of stdf files that have been corrupted by unintentionally running dos2unix on them.
- auto_flex – This flag enables the removal of site specific channels from testname so multiple sites will have the same name and can be readily compared.

The ConvertStdf script can be invoked directly from the R prompt as well as through the GUI. Here are some examples:

```
> ConvertStdf("a595.stdf.gz","a595.rtdf")


> ConvertStdf(do_summary=FALSE,stdf_name="a595.stdf.gz",rtdf_name="a595.rtdf")
```

In R, inputs to a function can be explicitly set with a ***variable_name* = *value***, or just ***value*** if the inputs are in the same order as the function declaration, or implicitly set to the defaults as defined in the function declaration if they are not present. The ConvertStdf function declaration is as follows:

```
ConvertStdf <- function(stdf_name="",rtdf_name="",auto_93k=TRUE,do_summary=TRUE,
                        just_fail_tests_summary=TRUE,endian="big",
                       stdf_dir="") {
```

3.2.12.1 ConvertStdf Output

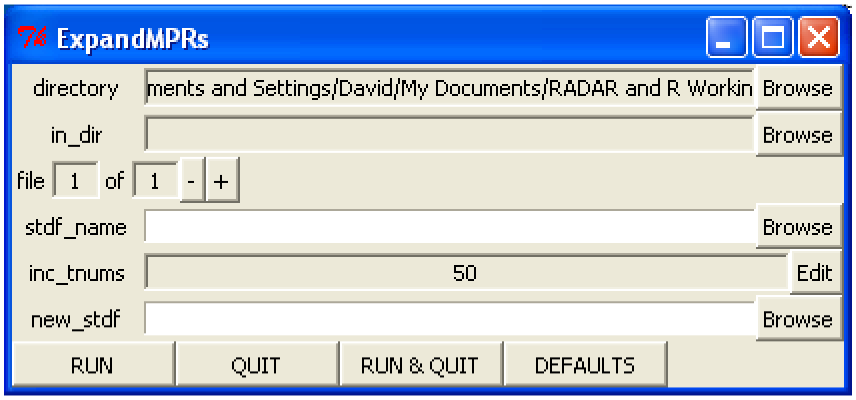
As well as generating the RTDF file, ConvertStdf can also generate a text formatted summary file. The summary file contains information extracted from the MIR record, the SBR records, the HBR records, the TSR records, and also the PRR records of the STDF file. If there are multiple sites enabled, the "Soft Binning Summary" section will be repeated, but with per site information rather than overall information.

```
SUMMARY for file: a595.stdf.gz
-----------------------------------------------------------------------------
Lot ID:       TEST
Sublot ID:
Test Program: mcc
Tester Type:  Catalyst
Tester ID:    a595-tst
Start Time:   2002-12-02 09:18:54 EST
Finish Time:  2002-12-02 09:37:41
-----------------------------------------------------------------------------
Soft Binning Summary: Part Count 336

_Count   _____% Bin_no  _  Soft_Bin_Name_____________
   263    78.3      1  P  PASS
    35    10.4      4  F  DELAY
    16     4.8      3  F  FUNC
    14     4.2     10  F  CONT
     4     1.2     11  F  POWER
     3     0.9      8  F  VIN
     1     0.3      9  F  VOUT
-----------------------------------------------------------------------------
Hard Binning Summary: Part Count 336
```

```
_Count   _____% Bin_no  _   Hard_Bin_Name_____________
   263    78.3      1
    55    16.4      3
    14     4.2      2
     4     1.2      4
-----------------------------------------------------------------------
Test Summary: Part Count 336

%_of_all  %_of_exec  execs_  fails_  Test_Name_____________
     4.2        4.2     336      14  Open/Short-
     1.2        1.2     322       4  IDD_Static
     1.2        1.3     318       4  VIN Test
     0.3        0.3     314       1  VTN_DTI_0_TO_7
     0.3        0.3     313       1  VTN_DCI
     0.3        0.3     312       1  VTP_DTIA_8_TO_15
     0.3        0.3     311       1  VOL_LV1P
     1.8        1.9     310       6  AMSDSM Test
     1.2        1.3     304       4  WRFEWNG Test
     0.3        0.3     300       1  OVERFLOW Test
     0.3        0.3     299       1  SCAN Test
     3.3        3.7     298      11  Delay Line #01
     1.2        1.4     287       4  Delay Line #03
     0.3        0.4     283       1  Delay Line #06
     5.7        6.7     282      19  Delay Line #07
-----------------------------------------------------------------------
PRR Soft Binning Summary: Part Count 336

_Count   _____% SBin_no  _
   263    78.3       1  P
    35    10.4       4  F
    16     4.8       3  F
    14     4.2      10  F
     4     1.2      11  F
     3     0.9       8  F
     1     0.3       9  F
-----------------------------------------------------------------------
```

3.2.13 ExpandMPRs

This script is used to generate PTR based stdf files from stdf files containing MPR records. This is of no real value for RADAR, but provided a solution for a proprietary package that wasn't able to process HPUX based 93K stdf files directly. For a given MPR record, multiple PTR records are generated with unique testnames built by appending "/" and pinname to the end of the MPR testname. The test numbers can either be all the same as the MPR, or if the inc_tnums flag is >0, they can increment starting at the MPR testnumber plus one. If an MPR creates more than "inc_tnums" PTRs, a warning is issue that you probably now have non unique test numbers.



The ExpandMPR Gui has the following parameters:

- directory
- in_dir
- file 1 of 1 +/-

  This window supports converting multiple files with a single click of the RUN button. If you have more than 1 file, this will allow you to scroll through the list of stdf_name's and associated new_stdf's. To select more than 1 file, click the stdf_name Browse button and either use the control key to select multiple files, or the shift key to select multiple sequential files in the browser window.
- stdf_name
- inc_tnums
- new_stdf

The ExpandMPRs script takes the following parameters:

- stdf_name – the name of the STDF file you wish to expand into a new STDF file
- new_stdf – the name of the larger but MPR free STDF file you wish to create
- inc_tnums - if set to 0, PTRs inherit the MPR test number, otherwise, the tnum is incremented for each PTR. A Warning is generated if the tnum incrementing goes beyond the value of inc_tnums
- in_dir – if the input stdf file is in a different directory than the directory you are working in, then include the absolute path here. If you are using the ExpandMPRs GUI, when you use the Browse button for the stdf_name, it will automatically fill in this field as appropriate.

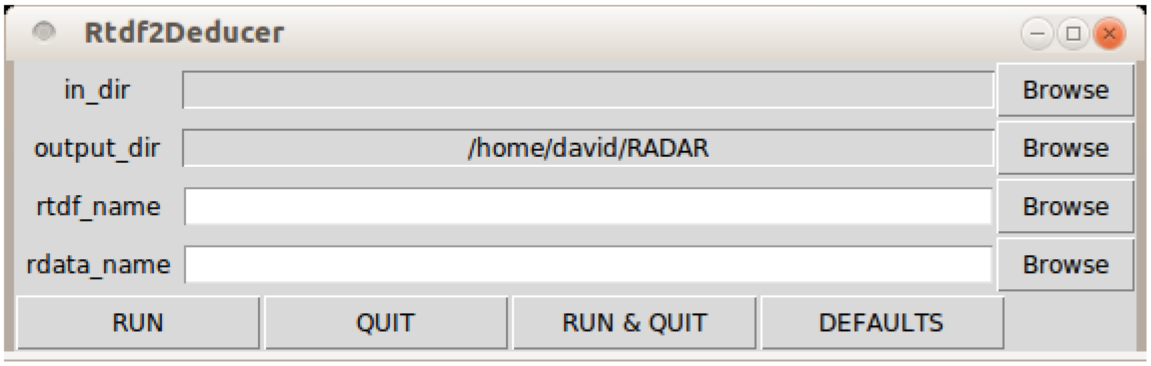
3.2.14 Rtdf2Deducer

Deducer is an open source project that "is designed to be a free easy to use alternative to proprietary data analysis software such as SPSS, JMP, and Minitab".

http://www.deducer.org/pmwiki/index.php?n=Main.DeducerManual

This script will read in an RTDF file and generate a .Rdata version of the file with a single object that is deducer friendly. It does this by:

- filter testnames to have only Deducer friendly characters
- appends testnames and units to ResultsMatrix column names
- combines DeviceFrame and ResultsMatrix into a single object



output_dir: /home/david/RADAR

- in_dir
- output_dir
- rtdf_name – the name of the RTDF file that you want to convert into a deducer friendly format.
- rdata_name – the name of the .Rdata file that you want to generate, that can then be loaded into Deducer.

3.3 Manipulating

There are several scripts included in this section to handle the more frequent types of manipulation that crop up in day to day ATE data analysis.

- FilterByBinning

  This script will generate a new RTDF file from an existing RTDF file by either keeping or removing devices matching the softbin list specified. For more details, see the FilterByBinning section.

- FilterByIndices

  This script will generate a new RTDF file from an existing RTDF file by either keeping or removing devices based on the indices in the DevicesFrame. NOTE: This is not the same as the part_id field! It can use a list of whole numbers for the indices, or it can also deal with a list of boolean logic that it will cycle through. If you wanted every 4th device, you would specify c(TRUE,FALSE,FALSE,FALSE) as thee indices. For more details, see the FilterByIndices section.

- FilterByResult

  This script will generate a new RTDF file from an existing RTDF file by either keeping or removing devices or results based on results matching the criteria specified. It can also just report the devices and results to the R console rather than generating a new RTDF. For more details, see the FilterByResult section.

- FindFirstFails

This script will generate a new RTDF file from an existing RTDF file, adding a “first_fail_test” field to the DevicesFrame and optionally a “fail_test_count” field. The “first_fail_test” field will be populated with the first failing test for each device (or “” if the device passes all tests), and the “fail_test_count” will be the total number of failing tests for each device. For more details, see the FindFirstFails section.

- Fingerprint

  This script does 2 related tasks. The first task is to compare 2 runs of the same group of devices to determine which tests are the best for fingerprinting. ie. Tests where a single device's repeatability is much tighter than the overall test repeatability. The second task is to use these tests to compare against a 3rd dataset to infer the part_id's for the 3rd set. This can be used to confirm the part_ids are correct for a particular “pull” during 1000 hour HTOL testing. For more details, see the Fingerprint section.

- MergeRtdf

  This script will take multiple RTDF files and merge them into a new single file. For more details, see the MergeRtdf section.

- RemoveAtXY

  This script will remove die at a specific X,Y coordinate location from an RTDF file. It is primarily used to remove gratuitous X=-1,Y=-1 parts that manage to creep into 93K wafer probe sessions and skew the wafermap. For more details, see the RemoveAtXY section

- ReplaceTestnames

  This script takes an RTDF file to be modified and a reference RTDF file with the desired testnames, and generates a new RTDF file with the testnames replaced, based on the test numbers. This script would be used for J9 data and for some cross-platform test correlation where the test numbers are consistent but the test names aren't. For more details, see the ReplaceTestnames section.

- RobustFilter

  This script can be used to screen outliers from a dataset. It uses robust statistics and will remove any measurements that are the specified # of robust standard deviations away from the robust mean for each test. For more details, see the RobustFilter section.

- RtdfTransform

  This script is used to modify or transform the tests in an RTDF file. The modification can be as simple as changing the testname or testnumber, to converting measurements from leakage current tests to resistance measurements, or voltage measurements to power in dBm. This script can be useful when trying to correlate data from 2 different test systems by coercing the data from one of the systems into the same format as the other to allow one-to-one comparisons. For more details, see the RtdfTransform section.

- ShrinkRetests

  This script is used to scan through an RTDF file, looking for multiple occurrences of the same part_id, and removing all but the last occurrence. This is typically used in HTRB or temperature characterization where the device contact can become unreliable and one may re-insert/retest a device multiple times until it passes. As of 0v6p4, it can instead look for repeated xy wafer coordinate pairs. For more details, see the ShrinkRetests section.

- SplitBySubstr

This script can be used to separate an RTDF file into multiple files based on test conditions coded into the testnames. An example would be a group of tests that are repeated 3 times in a test program, with the 2nd repetition having "_VMIN" appended to the testnames and the 3rd repetition having "_VMAX" appended to the testnames. For more details, see the SplitBySubstr section.

- SplitConditions

  For RTDF files that contain conditions in the DevicesFrame, this function can be used to separate the data into multiple RTDF files based on unique conditions. ie. if the data is from a characterization session where the part was run at conditions of min, nominal and max voltages, 3 separate RTDF files would be generated. For more details, see the SplitConditions section.

- SplitSites

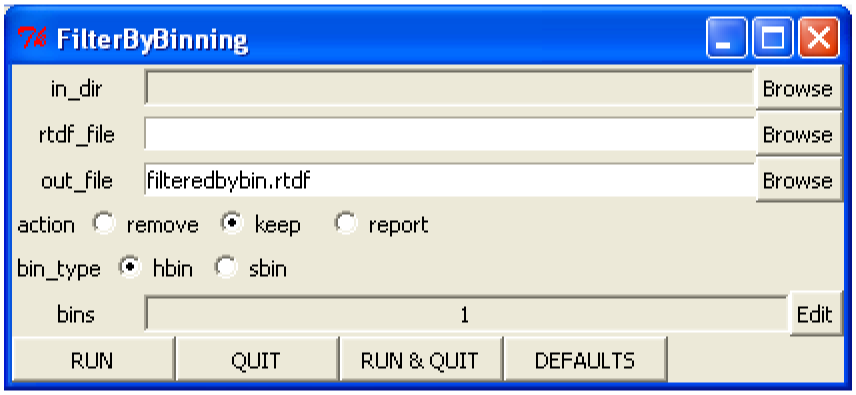
  For multisite datalogs, this function can be used to separate the data into multiple RTDF files based on test site. ie, if the data is for a quad site test session, SplitSites would generate 4 separate RTDF files. For more details, see the SplitSites section.

- SplitWafers

  For multiwafer datalogs, this function can be used to separate the data into per-wafer RTDF files. For more details, see the SplitWafers section.

3.3.1 FilterByBinning

FilterByBinning reads in an rtdf file, filters the data based on the binning criteria specified, and generates a new rtdf file



in_dir:
rtdf_file:
out_file: filteredbybin.rtdf
action: ( ) remove [✓ keep] ( ) report
bin_type: [✓ hbin] ( ) sbin
bins: 1

. The various fields in detail:

– in_dir

– rtdf_file

– out_file

– action

– bin_type

– bins

–

3.3.2 FilterByIndices

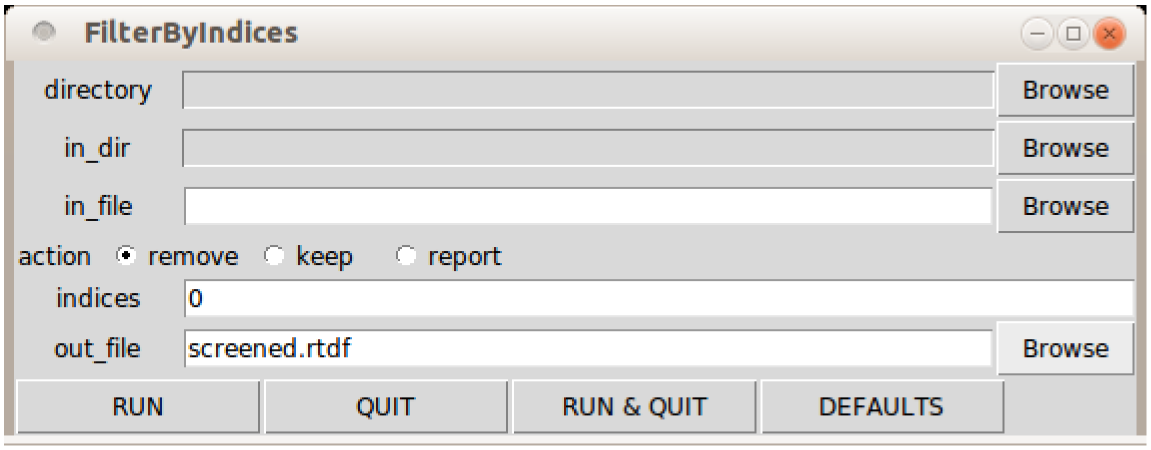
This script will generate a new RTDF file based on the input RTDF file, but with the devices at the indices specified removed or all other devices removed from the dataset. NOTE: the indices are the positions in the DevicesFrame, not the part_id's. ie. if you have 3 parts, with part_ids of 1, 3 and 4, then:

"1" = DevicesFrame[1,"part_id"]

"3" = DevicesFrame[2,"part_id"]

"4" = DevicesFrame[3,"part_id"]

If you wanted to remove part_id "3", the index would be 2.



FilterByIndices

directory:

in_dir:

in_file:

action: [✓ remove] keep report

indices: 0

out_file: screened.rtdf

The various fields in detail:

- directory
- in_dir
- in_file
- action, one of:
  - remove – will remove devices matching the indices field from the dataset
  - keep – will remove all devices except those matching the indices field from the dataset
  - report – will just print to the console the expanded version of the indices field
- indices - Indices are entered in R syntax. Indices can be a vector of numbers or it can be a vector of boolean values. Examples of numeric entry:

5:10	..ie indices 5,6,7,8,9,10
c(1:5,7)	...ie indices 1,2,3,4,5,7

  Examples of boolean values:

  c(TRUE,FALSE)

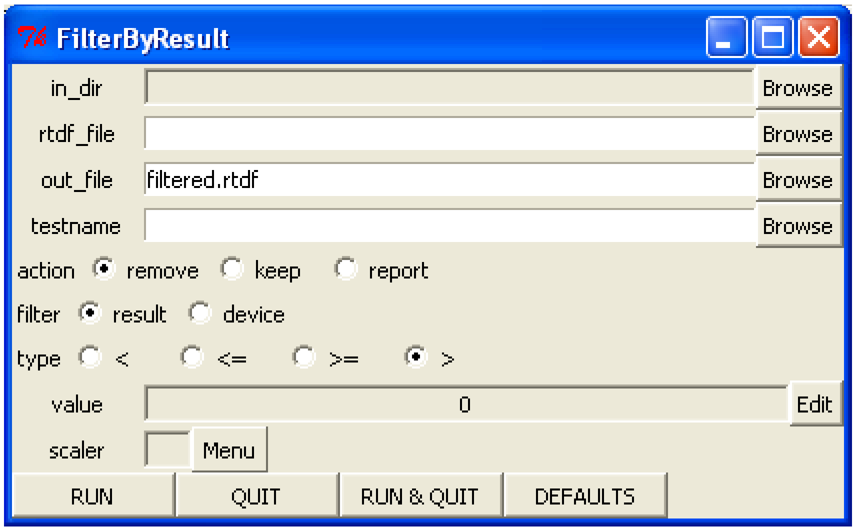
  c(F,T,F,F)

  If indices are boolean, then the indices vector will be recycled for the full length of DevicesFrame. In the above c(F,T,F,F) example every 4th device would be selected, starting at the 2nd device

- out_file

3.3.3 FilterByResult

This script can either report on or keep or remove devices or results based on whether the measurement is above or below the value specified. This is useful in determining which device is the outlier, and what it measured, or removing the part or measurement from the dataset so that it doesn't skew the statistics.



in_dir	
rtdf_file	
out_file	filtered.rtdf
testname	
action	[✓ remove] keep report
filter	[✓ result] device
type	< <= >= [✓ >]
value	0
scaler	

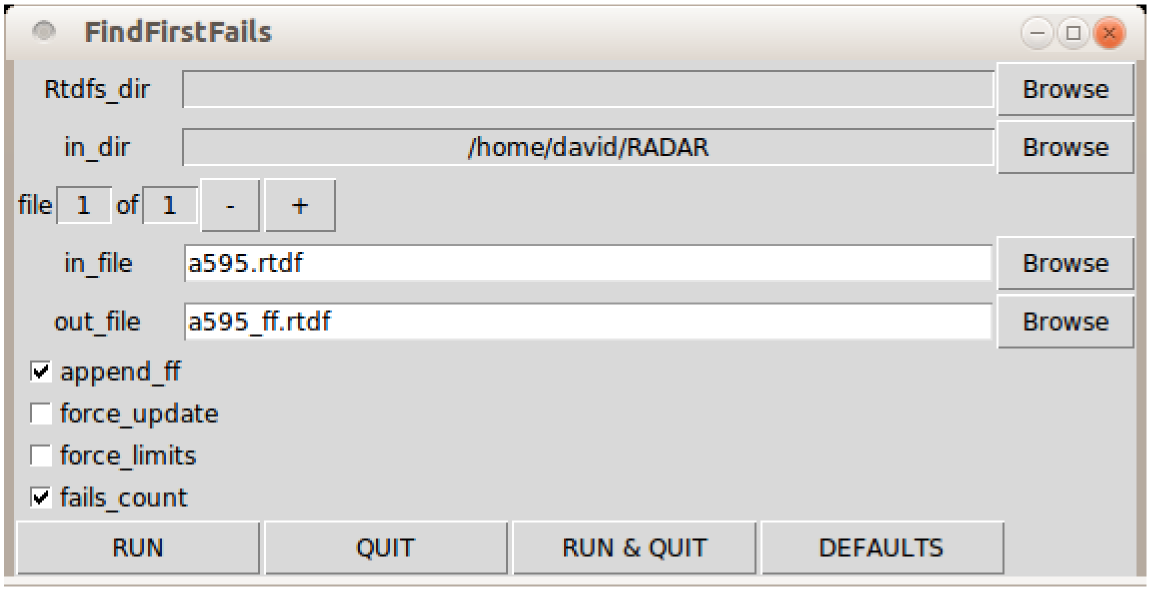
The various fields in detail:

- in_dir
- rtdf_file
- out_file
- testname
- action
- filter
- type
- value
- scaler

3.3.4 FindFirstFails

This script generates a “first_fail_test” field in the DevicesFrame and populates it with the name of the first failing test for the given device, or “” if the device passed. Optionally, a “fail_test_count” field can also be added that counts how many tests failed. (more useful if the datalog is in continue-on-fail mode!) If the RTDF file includes the TestFlagMatrix object (ie. do_testflag_matrix set to TRUE when running ConvertStdf), this script will use that matrix to determine failing tests. If the TestFlagMatrix object is not in the RTDF file, or if the “force_use_limits” flag is set to TRUE, this script will instead determine pass/fail status by looking at the result vs. the limits for each test for each device.

If the RTDF file already contains a ”first_fail_test” field in the DevicesFrame, this script will not change anything. To force it to overwrite this field, set the force_update flag to true. (useful if you are adjusting limits and want to confirm what the yield would be with the new limits)


FindFirstFails

Rtdfs_dir:

in_dir: /home/david/RADAR

file 1 of 1

in_file: a595.rtdf

out_file: a595_ff.rtdf

- [✓] append_ff
- [ ] force_update
- [ ] force_limits
- [✓] fails_count

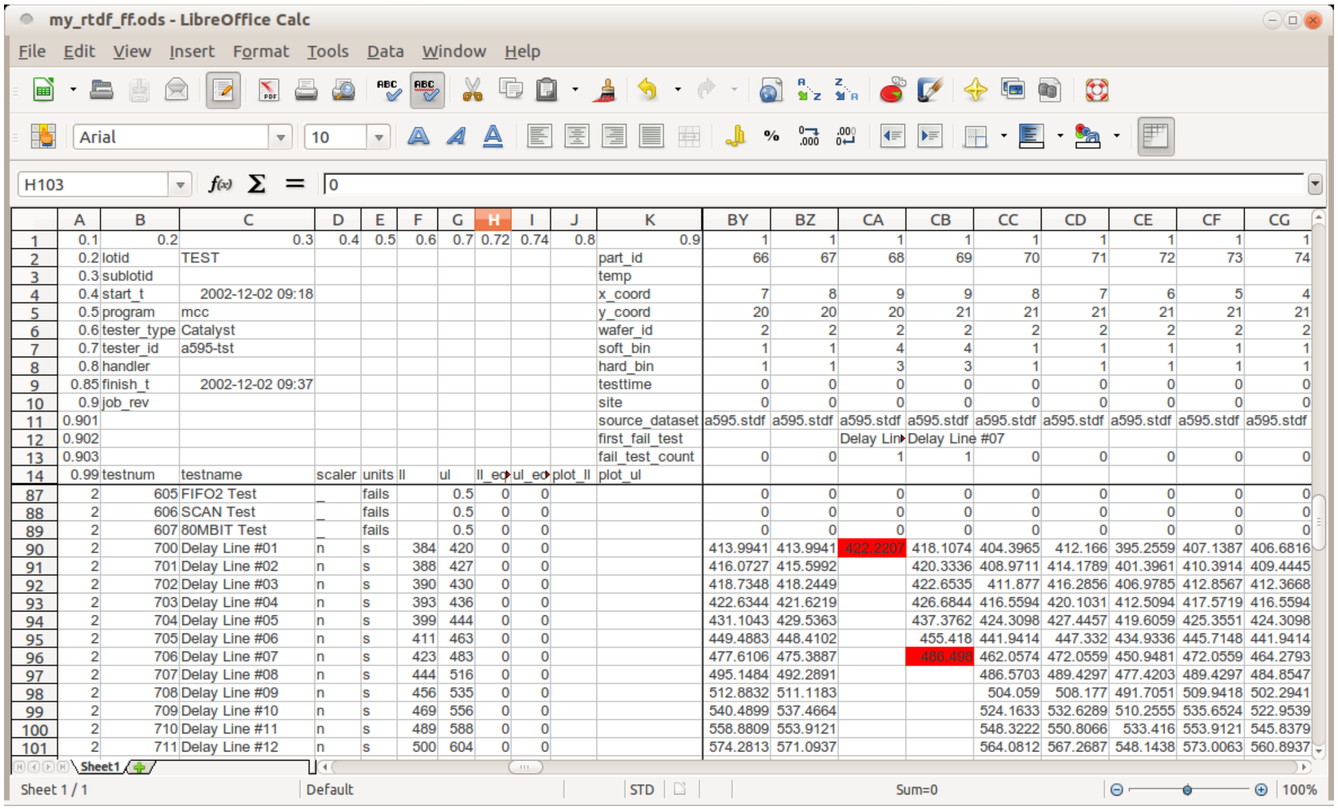
The various fields in detail:

- Rtdfs_dir
- in_dir
- file 1 of 1 +/-

  This window supports converting multiple files with a single click of the RUN button. If you have more than 1 file, this will allow you to scroll through the list of in_file name's and associated out_file's. To select more than 1 file, click the in_file Browse button and either use the control key to select multiple files, or the shift key to select multiple sequential files in the browser window.
- in_file – the name of the RTDF file to process.
- out_file – the name of the new RTDF file to create
- append_ff – if this flag is set, the GUI will automatically generate the out_file names by inserting “_ff” just before the “.rtdf” portion of the in_file names.
- force_update – if this flag is set, this script will overwrite the “first_fail_test” field if it already exists in the DevicesFrame.
- force_limits – if this flag is set, this script will use the ParametersFrame limits to determine pass/fail status even if a TestFlagMatrix object exists in the RTDF file.
- fails_count – if this flag is set, this script will also create a “fail_test_count” field in the DevicesFrame that shows how many tests failed for each device

3.3.4.1 FindFirstFails in CSV file

The additional DevicesFrame fields created by FindFirstFails are recognized by ConvertCSV. An example is shown below. NOTE: This example also shows conditional formatting, so that failing results have a red background.


my_rtdf_ff.ods - LibreOffice Calc
File Edit View Insert Format Tools Data Window Help
Arial
10
H103
0
Sheet1
Sheet 1 / 1
Default
STD
Sum=0
100%

	A	B	C	D	E	F	G	H	I	J	K
1	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.72	0.74	0.8	0.9
2	0.2	lotid	TEST								part_id
3	0.3	sublotid									temp
4	0.4	start_t	2002-12-02 09:18								x_coord
5	0.5	program	mcc								y_coord
6	0.6	tester_type	Catalyst								wafer_id
7	0.7	tester_id	a595-tst								soft_bin
8	0.8	handler									hard_bin
9	0.85	finish_t	2002-12-02 09:37								testtime
10	0.9	job_rev									site
11	0.901										source_dataset
12	0.902										first_fail_test
13	0.903										fail_test_count
14	0.99	testnum	testname	scaler	units	ll	ul	ll_eq	ul_eq	plot_ll	plot_ul
87	2	605	FIFO2 Test	_	fails		0.5	0	0		
88	2	606	SCAN Test	_	fails		0.5	0	0		
89	2	607	80MBIT Test	_	fails		0.5	0	0		
90	2	700	Delay Line #01	n	s	384	420	0	0		
91	2	701	Delay Line #02	n	s	388	427	0	0		
92	2	702	Delay Line #03	n	s	390	430	0	0		
93	2	703	Delay Line #04	n	s	393	436	0	0		
94	2	704	Delay Line #05	n	s	399	444	0	0		
95	2	705	Delay Line #06	n	s	411	463	0	0		
96	2	706	Delay Line #07	n	s	423	483	0	0		
97	2	707	Delay Line #08	n	s	444	516	0	0		
98	2	708	Delay Line #09	n	s	456	535	0	0		
99	2	709	Delay Line #10	n	s	469	556	0	0		
100	2	710	Delay Line #11	n	s	489	588	0	0		
101	2	711	Delay Line #12	n	s	500	604	0	0		

	BY	BZ	CA	CB	CC	CD	CE	CF	CG
1	1	1	1	1	1	1	1	1	1
2	66	67	68	69	70	71	72	73	74
3									
4	7	8	9	9	8	7	6	5	4
5	20	20	20	21	21	21	21	21	21
6	2	2	2	2	2	2	2	2	2
7	1	1	4	4	1	1	1	1	1
8	1	1	3	3	1	1	1	1	1
9	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0
11	a595.stdf	a595.stdf	a595.stdf	a595.stdf	a595.stdf	a595.stdf	a595.stdf	a595.stdf	a595.stdf
12			Delay Lin	Delay Line #07					
13	0	0	1	1	0	0	0	0	0
14									
87	0	0	0	0	0	0	0	0	0
88	0	0	0	0	0	0	0	0	0
89	0	0	0	0	0	0	0	0	0
90	413.9941	413.9941	422.2207	418.1074	404.3965	412.166	395.2559	407.1387	406.6816
91	416.0727	415.5992		420.3336	408.9711	414.1789	401.3961	410.3914	409.4445
92	418.7348	418.2449		422.6535	411.877	416.2856	406.9785	412.8567	412.3668
93	422.6344	421.6219		426.6844	416.5594	420.1031	412.5094	417.5719	416.5594
94	431.1043	429.5363		437.3762	424.3098	427.4457	419.6059	425.3551	424.3098
95	449.4883	448.4102		455.418	441.9414	447.332	434.9336	445.7148	441.9414
96	477.6106	475.3887		486.498	462.0574	472.0559	450.9481	472.0559	464.2793
97	495.1484	492.2891			486.5703	489.4297	477.4203	489.4297	484.8547
98	512.8832	511.1183			504.059	508.177	491.7051	509.9418	502.2941
99	540.4899	537.4664			524.1633	532.6289	510.2555	535.6524	522.9539
100	558.8809	553.9121			548.3222	550.8066	533.416	553.9121	545.8379
101	574.2813	571.0937			564.0812	567.2687	548.1438	573.0063	560.8937

3.3.5 Fingerprint

This script can perform 2 related tasks, either as separate executions, or in a single execution. The first task is to compare 2 datasets for the same group of devices where we know the part_id is correct to derive a list of "fingerprint" tests, tests that have very tight repeatability for a single part vs. the population. The second task is to then use this list of fingerprint tests to infer the correct part_id's of a 3rd dataset. This would be used when there is operator error in an HTOL pull, such that the parts / part_ids in the datalog are not 100% correct.

3.3.5.1 Creating the fingerprint test list

The "fingerprint" list is created by sorting tests that have the highest correlation run vs. run down to a cut-off of r>0.95. For the second and subsequent tests in the list, the test also has to have a cross correlation to any preceding test in the "fingerprint" list of r<0.70. To perform the first task, both a ref_rtdf and a ref2_rtdf file is required. These files MUST have part_ids that are correct. The "fingerprint" list will be saved into a file called "my_testlist.R". An example my_testlist.R file would look like:

```
my_testlist = c(
    "FREQ_OSC" ,                               # corr 1.000, xcorr 0.00
    "V_VREG" ,                                 # corr 1.000, xcorr 0.24
    "R_PULLDOWN_RSTB" ,                        # corr 1.000, xcorr 0.62
    "V_OUTSWING_OPAMP1" ,                      # corr 1.000, xcorr 0.42
    "RTERM_RFIN" ,                                    # corr 1.000, xcorr 0.48
       :
    "I_VCC_SLEEP" ,                                   # corr 0.999, xcorr 0.32
    "V_OFFSET_OPAMP2" ,                               # corr 0.998, xcorr 0.69
    "V_OFFSET_OPAMP1" ,                               # corr 0.995, xcorr 0.68
    "VBIAS_LFPIN" ,                                   # corr 0.990, xcorr 0.52
    "V_MUTE_OPAMP2" ,                                 # corr 0.983, xcorr 0.66
    "VBIAS_RFIN1"                                     # corr 0.952, xcorr 0.62
    )
```

3.3.5.2 Fingerprinting

To do the fingerprinting, you should only need around 7 fingerprint qualifited tests to precisely sort parts even in samples of 100's of devices. The GUI supports up to 30 tests in the list and a separate field to specify how many of these to use. If you are supplying a testlist, you do not need to supply a ref2_rtdf file.

One benefit of doing the fingerprinting as 2 separate tasks is that the datalogs used to generate the fingerprint testlist do not need to include the same devices as used in the actual fingerprinting. An example could include the following 4 data files:

- run1_of_my_favourite_40parts.rtdf
- run2_of_my_favourite_40parts.rtdf
- htol_0hours_77units.rtdf
- htol_512hours_77units.rtdf

The first task would set..

- rtdf_name as “”
- ref_rtdf_name as “run1_of_my_favourite_40parts.rtdf”
- ref2_rtdf_name as “run1_of_my_favourite_40parts.rtdf”

… and would generate the my_testlist.R file.

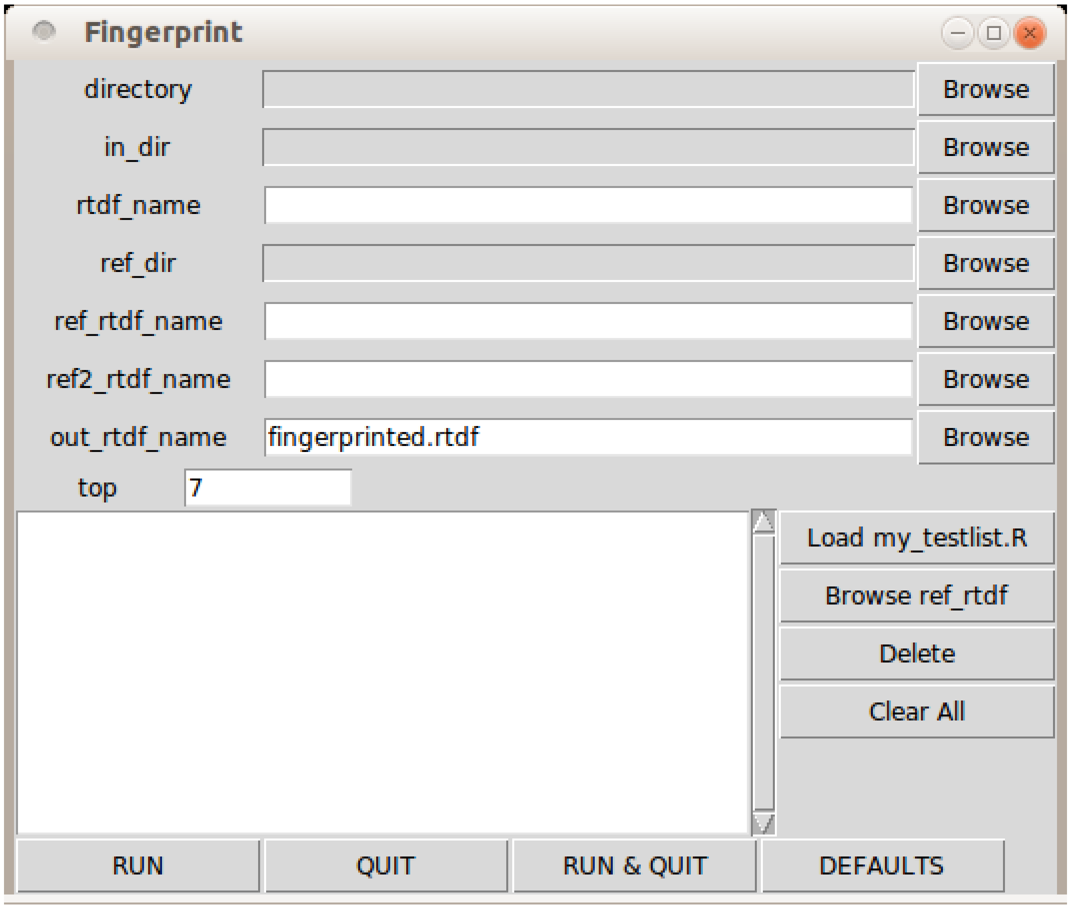
The 2nd task would set...

- rtdf_name as “htol_512hours_77units.rtdf”
- ref_rtdf_name as “htol_0hours_77units.rtdf”
- ref2_rtdf_name as “”
- Load my_testlist.R

… and would generate the fingerprinted.rtdf file. When generating the fingerprinted.rtdf file, the script also prints to the console any alternate parts that could also be a match, along with a numeric value of the mismatch for a particular part. An example is shown below. In this case, there were 2 devices that are fairly similar in the 40 part sample, but the script correctly finds the correct match. In the below example, the 40 devices were randomly scrambled.

```
For part  1, part_id    1, best fit: part 12 partid   12 max=0.006
For part  2, part_id    2, best fit: part 35 partid   35 max=0.011
                      alternate fit: part  4 partid    4 max=0.079
For part  3, part_id    3, best fit: part  2 partid    2 max=0.014
For part  4, part_id    4, best fit: part 33 partid   33 max=0.009
For part  5, part_id    5, best fit: part 40 partid   40 max=0.009
For part  6, part_id    6, best fit: part  5 partid    5 max=0.006
For part  7, part_id    7, best fit: part 14 partid   14 max=0.005
For part  8, part_id    8, best fit: part 22 partid   22 max=0.004
For part  9, part_id    9, best fit: part 16 partid   16 max=0.008
For part 10, part_id   10, best fit: part 10 partid   10 max=0.009
For part 11, part_id   11, best fit: part 19 partid   19 max=0.006
For part 12, part_id   12, best fit: part 38 partid   38 max=0.006
For part 13, part_id   13, best fit: part  8 partid    8 max=0.017
For part 14, part_id   14, best fit: part 21 partid   21 max=0.004
For part 15, part_id   15, best fit: part 17 partid   17 max=0.009
For part 16, part_id   16, best fit: part 27 partid   27 max=0.005
For part 17, part_id   17, best fit: part 23 partid   23 max=0.007
For part 18, part_id   18, best fit: part  9 partid    9 max=0.006
For part 19, part_id   19, best fit: part 34 partid   34 max=0.007
For part 20, part_id   20, best fit: part 24 partid   24 max=0.004
For part 21, part_id   21, best fit: part 31 partid   31 max=0.014
For part 22, part_id   22, best fit: part  1 partid    1 max=0.007
For part 23, part_id   23, best fit: part  7 partid    7 max=0.008
For part 24, part_id   24, best fit: part 20 partid   20 max=0.016
For part 25, part_id   25, best fit: part  3 partid    3 max=0.008
For part 26, part_id   26, best fit: part 25 partid   25 max=0.018
For part 27, part_id   27, best fit: part 30 partid   30 max=0.014
For part 28, part_id   28, best fit: part 15 partid   15 max=0.009
For part 29, part_id   29, best fit: part 36 partid   36 max=0.008
For part 30, part_id   30, best fit: part 32 partid   32 max=0.011
For part 31, part_id   31, best fit: part 37 partid   37 max=0.010
For part 32, part_id   32, best fit: part 11 partid   11 max=0.009
For part 33, part_id   33, best fit: part 26 partid   26 max=0.007
For part 34, part_id   34, best fit: part  4 partid    4 max=0.004
                      alternate fit: part 35 partid   35 max=0.081
For part 35, part_id   35, best fit: part 39 partid   39 max=0.011
For part 36, part_id   36, best fit: part 13 partid   13 max=0.005
For part 37, part_id   37, best fit: part  6 partid    6 max=0.008
For part 38, part_id   38, best fit: part 18 partid   18 max=0.007
For part 39, part_id   39, best fit: part 28 partid   28 max=0.009
For part 40, part_id   40, best fit: part 29 partid   29 max=0.007
Finished!
>
```

3.3.5.3 *The Fingerprint GUI*


Fingerprint

directory

in_dir

rtdf_name

ref_dir

ref_rtdf_name

ref2_rtdf_name

out_rtdf_name: fingerprinted.rtdf

top: 7

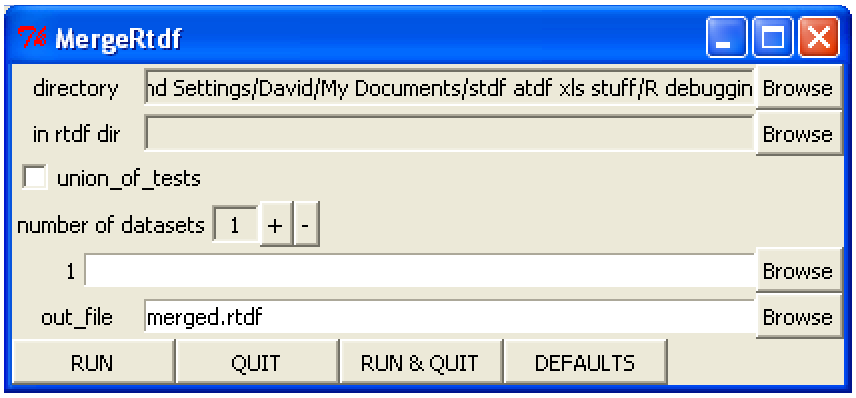
The various fields in detail:

- directory
- in_dir
- rtdf_name – the name of the RTDF file which you suspect has scrambled part_ids
- ref_dir
- ref_rtdf_name – the name of the RTDF file that contains the same devices as in rtdf_name, possibly more, but is known to be 'clean', ie the part_id's are correct.
- ref2_rtdf_name – optional, if you want the script to automatically generate the fingerprint test list. This should be a second datalog containing the same devices as the ref_rtdf file, where we know the part_id's are also correct.

- out_rtdf_name – optional, the name of the RTDF file generated by this script when fingerprinting. It should be the same as the rtdf_name file, except that the part_id fields will be overwritten with the closest match to ref_rtdf_name, when using the fingerprint testlist for inferring the part_ids.
- top – an integer number that specifies the maximum number of tests from the testlist to actually use when fingerprinting. 7 is a pretty good number to use.
- testlist box – If you are supplying both a ref and ref2 rtdf file, then you do not need to supply a testlist. If you are supplying a testlist, you can either import the list from a .R file by using the "Load my_testlist.R" button, or you can browse the available tests from the ref rtdf file by using the "Browse ref_rtdf" file to bring up the list of parameters available and then selecting the tests you want to use. You can also select items in the testlist display box and hit the "Delete" button beside the display box if there are tests you want to remove from the list, or you can hit the "Clear all" button to empty the list.

3.3.6 MergeRtdf

This script will take multiple RTDF files and combine them into a single file. This is useful for cases where a particular lot was tested as several test sessions, but you want to see the data as a single data set.


MergeRtdf

directory: nd Settings/David/My Documents/stdf atdf xls stuff/R debuggin

in rtdf dir:

union_of_tests

number of datasets: 1

1:

out_file: merged.rtdf

The various fields in detail:

- directory
- in rtdf dir
- union_of_tests

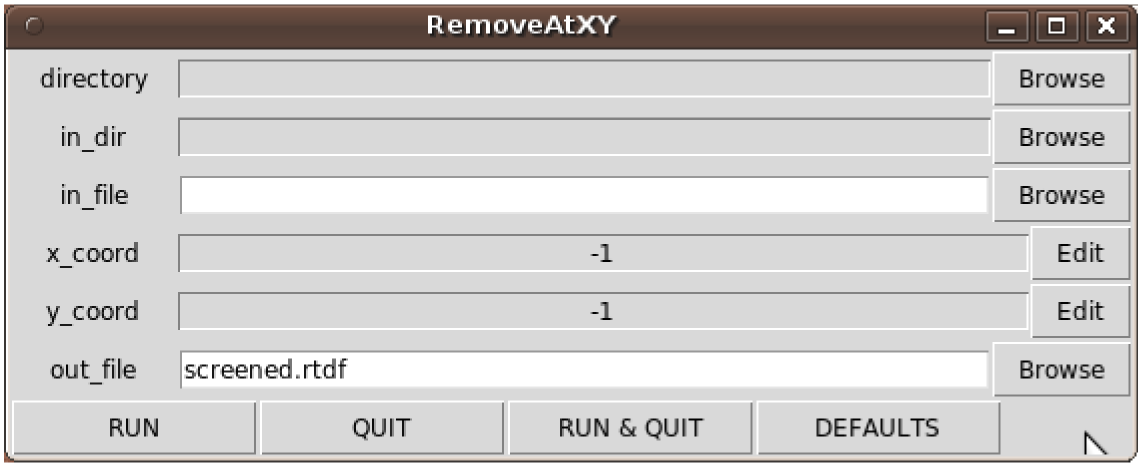
  If this flag is not set, then any tests that exist in the 2nd+ datasets but not in the 1st dataset are ignored. If this flag is set, the resulting ParametersFrame will be a superset of all the ParameterFrames, ie. the union of the different sets of tests.
- number of datasets
- datasets

  In the Browse window you can select one or more files. To select multiple adjacent files, hold the shift key when selecting. To select non-adjacent file names, use the control key when selecting.
- out_file

If you are calling MergeRtdf directly from the command line rather than through the GUI, there is no input field for number of datasets. The datasets field is a vector of strings of the RTDF filenames.

3.3.7 RemoveAtXY

This script is primarily used to remove those die with X=-1,Y=-1 coordinates that seem to occasionally sneak their way into V93K datalog files.


RemoveAtXY

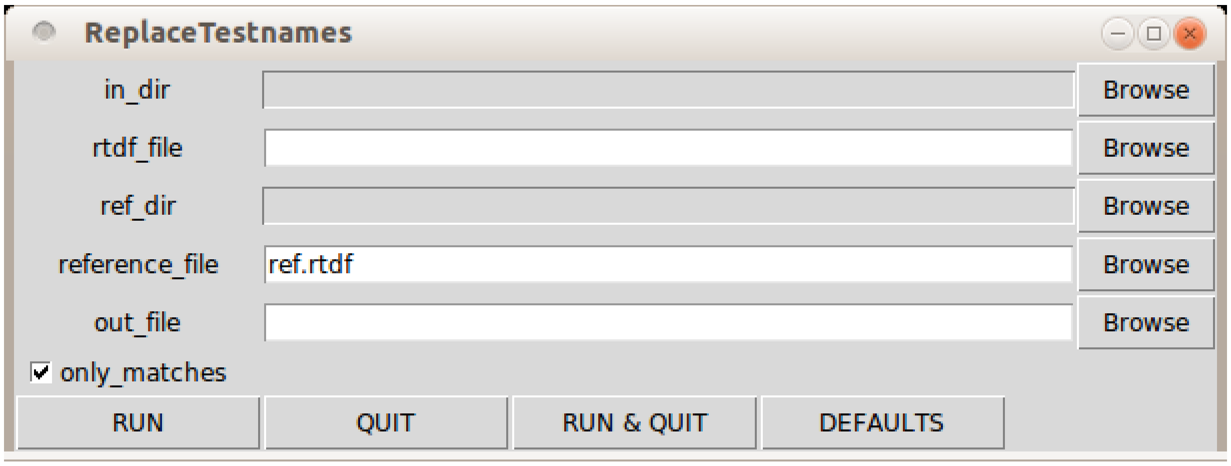
directory		Browse
in_dir		Browse
in_file		Browse
x_coord	-1	Edit
y_coord	-1	Edit
out_file	screened.rtdf	Browse

The various fields in detail:

- directory
- in_dir
- in_file
- x_coord
- y_coord
- out_file

3.3.8 ReplaceTestnames

This script takes an RTDF file to be modified and a reference RTDF file with the desired testnames, and generates a new RTDF file with the testnames replaced, based on the test numbers. This script would be used for J9 data and for some cross-platform test correlation where the test numbers are consistent but the test names aren't. The reference_file RTDF only needs to contain the ParametersFrame object.


ReplaceTestnames

in_dir		Browse
rtdf_file		Browse
ref_dir		Browse
reference_file	ref.rtdf	Browse
out_file		Browse

✓ only_matches

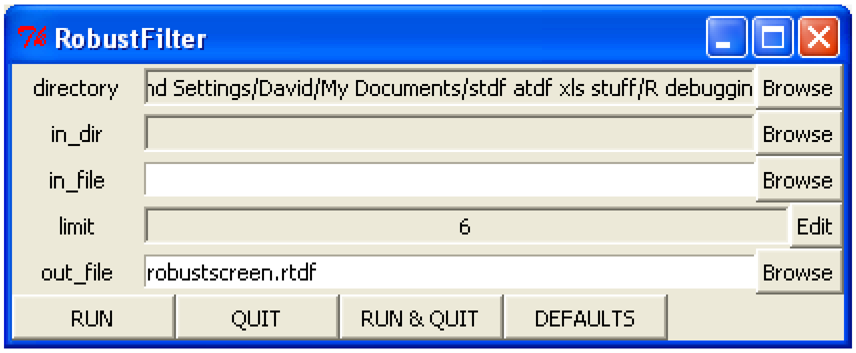
The various fields in detail:

- in_dir
- rtdf_file
- ref_dir
- reference_file
- out_file
- only_matches checkbox

See the tutorial section for a walk-through of how this script would be used with J9 data.

3.3.9 RobustFilter

This script can be used to screen outliers from a dataset. It uses robust statistics and will remove any measurements that are the specified # of robust standard deviations away from the robust mean for each test.


RobustFilter

directory	nd Settings/David/My Documents/stdf atdf xls stuff/R debuggin	Browse
in_dir		Browse
in_file		Browse
limit	6	Edit
out_file	robustscreen.rtdf	Browse

The various fields in detail:

- directory
- in_dir
- in_file
- limit: This is the number of robust standard deviations away from the robust mean to use as the

criteria for removing measurements

– out_file

3.3.10 RtdfTransform

This script will read in an RTDF file and a CSV file containing a set of transform instructions and generate a new RTDF file based on these transform instructions. The instructions can be as basic as just changing test numbers or test names, or as elaborate as converting measurements from voltage to dBm, period measurements to frequency measurements, or leakage measurements to pull-up resistor measurements.

The transform.csv file conforms to a specific format. The first row is the header row. The headers should be: testnum, new_testname, old_testname, prefix, units, new_limits_flag, new_ll, new_ul, multiplier, power, log, and shift. The description of each column is as follows:

- testnum

  this is the test number to use for the new test. If it is empty, the RtdfTransform script will use the old_testname's test number.

- new_testname

  This is the test name as it will appear in the transformed file. This is the only mandatory column

- old_testname

  This specifies the test to use from the input RTDF file to generate the transformed test. If this field is empty, the script will look for a test matching the new_testname name.

- prefix

  This specifies the units prefix for the new test. It would typically be a single character like "m" for milli, "n" for nano, "_" for no scaling, "M" for Mega, etc. If this field is empty and the units field is not empty, then there is no scaling. If this field is empty and the units field is also empty, then the prefix from the old_testname is used.

- units

  This specifies the units for the new test. If this field is empty, use the units from the old_testname

- new_limits_flag

  if this field is empty, the old_testname limits will be used, else the new limits will be used

- new_ll

  lower limit to use if the new_limits_flag is set

- new_ul

  upper limit to use if the new_limits_flag is set

- multiplier

  if empty, a multiplier of x1 is used

- power

  if empty, a power of +1 is used

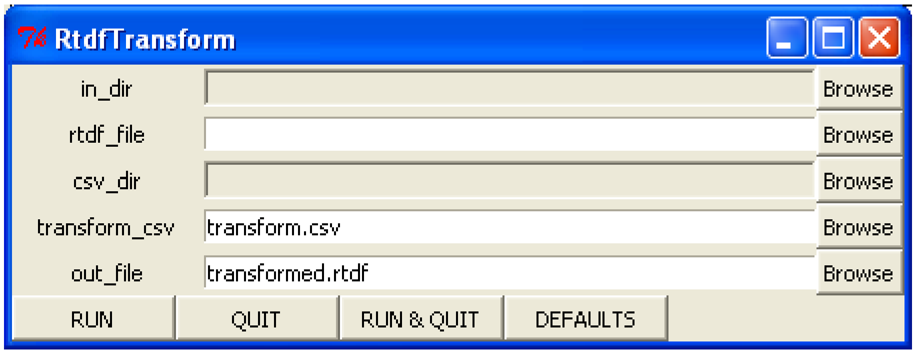
- log

  if empty or 0, no log is applied, if =1 then 10*log() is applied, if =2 then 20*log() is applied, if=-1 then 10^(/10) is applied, if -2 then 10^(/20.0) is applied.

  (ie for converting V or W to dB, or dB to V or W)

- shift

  if empty a value of 0 is assumed. the shift is added to the result



in_dir	
rtdf_file	
csv_dir	
transform_csv	transform.csv
out_file	transformed.rtdf

The various fields in detail:

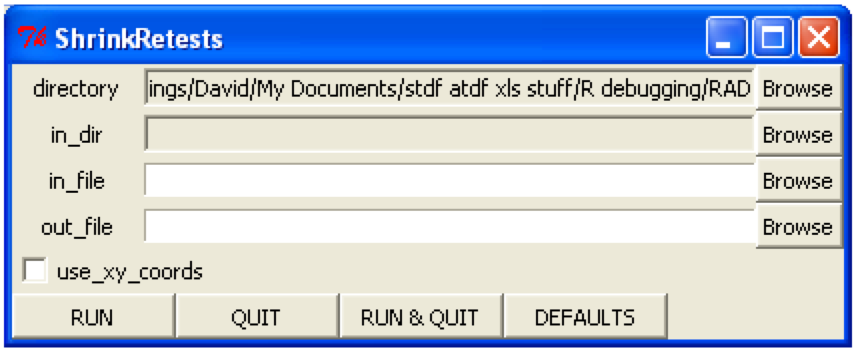
- in_dir
- rtdf_file
- csv_dir
- transform_csv
- out_file

3.3.10.1 *Example transform CSV file*

```
"test_num_","test_name_________","stdf_testname","prefix","units","limits_flag","new_ll","new_ul","multiplier","power","log","shift"
2601,"freq_error_clock2M","freq_clock2M","M","Hz",1,-0.098,0.102,,,,-2.048
2602,"period_clock2M","freq_clock2M","u","sec",1,0.40,0.60,,-1,,
28000,"r_pulldown/SEL","IIH_SEL","K","ohm",1,33,94.3,3.3,-1,,
```

3.3.11 ShrinkRetests

This script is used to search for occurences of retest (ie the part_id occurs more than once in the dataset) and to then remove all but the last occurrence.



directory: ings/David/My Documents/stdf atdf xls stuff/R debugging/RAD

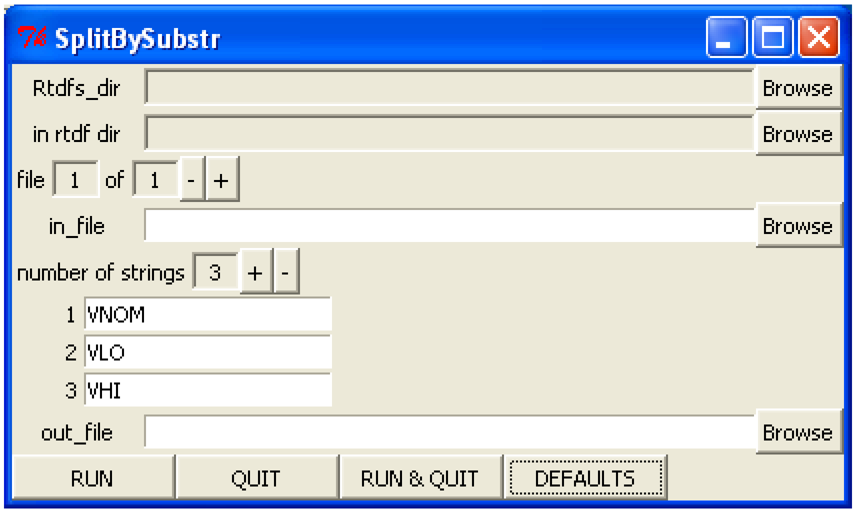
use_xy_coords: ☐

The various fields in detail:

- directory
- in_dir
- in_file
- out_file
- use_xy_coords – if the data is from wafer probe, then it is most likely that the part_id field will not be as useful as the x_coord and y_coord fields for determining retest die. In this case, check this box.

3.3.12 SplitBySubstr

This script will split an RTDF file into multiple RTDF files based on parameters which contain the substrings provided. In the example below, the number of strings was set to 3, and the substrings "Vnom", "Vmin", and "Vmax" were entered. Any parameters that have testnames that either have "_Vnom_" in their testname, or end in "_Vnom" will be split into a separate RTDF file. The "_Vnom" will also be removed from these testnames. Of the remaining parameters, any that have testnames that either have "_Vmin_" in their testname or end in "_Vmin" will be split into a second output RTDF file, etc. Any tests that do not match any of the strings will end up in the first RTDF file. This script is useful as another approach for tracking test conditions in test program and for being able to separate the data easily afterwards. In the case of below, the test program runs some of the tests 3 times, the first time with the normal name, the second time at minimum supply voltages and has "_Vmin" appended to these testnames, and the third time at maximum supply voltages and has "_Vmax" appended to these names. After running this script, you would then be able to run PlotRtdf on the 3 generated files and compare the measurements between the 3 conditions.



file 1 of 1

number of strings 3

1 VNOM
2 VLO
3 VHI

The various fields in detail:

- Rtdfs_dir

  the directory that the output RTDF files will be written to. If empty, will default to the Output_dir

- in rtdf dir

  the directory that the input RTDF files will be found. If empty, will default to the Rtdfs_dir.

- file 1 of 1 +/-

  This window supports converting multiple files with a single click of the RUN button. If you have more than 1 file, this will allow you to scroll through the list of in_file name's and associated out_file's. To select more than 1 file, click the in_file Browse button and either use the control key to select multiple files, or the shift key to select multiple sequential files in the browser window.

- in_file

  The name of the RTDF file that will be parsed and split into multiple RTDF files based on parameters matching the substrings supplied.

- strings

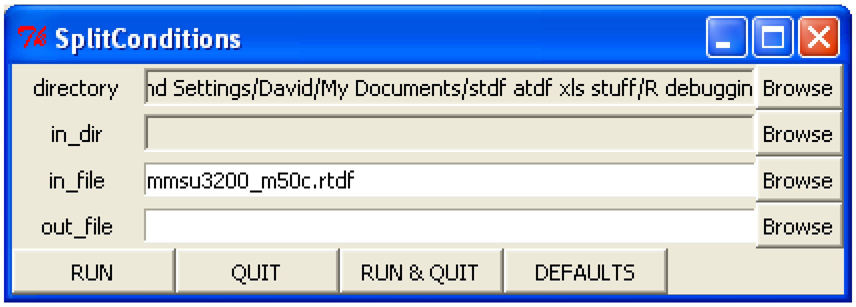
  the list of substrings to search for in the testnames. testnames containing these substrings will be split out into separate RTDF files, with these substrings removed from those testnames.

- out_file

  if this field is empty, the output name will be built from the in_file name with "_"+ associated string appended to the name before the ".rtdf" extension. Otherwise, the "_" + associated string will be appended to the supplied name before the ".rtdf" extension.

3.3.13 SplitConditions

This script will take an RTDF file that contains condition fields in the DevicesFrame, determine the number of unique conditions, and split the file into smaller files based on these unique conditions.


SplitConditions

directory: nd Settings/David/My Documents/stdf atdf xls stuff/R debuggin

in_dir:

in_file: mmsu3200_m50c.rtdf

out_file:

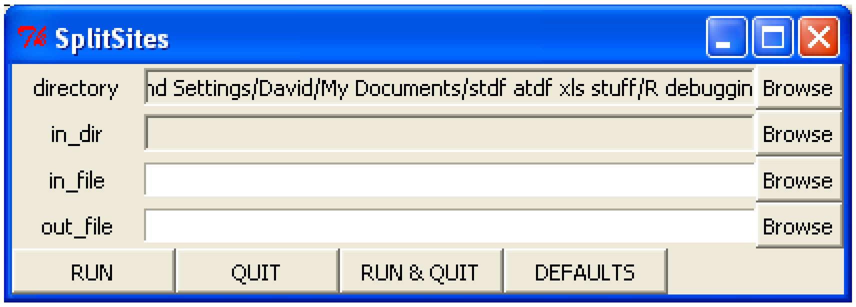
The various fields in detail:

- directory
- in_dir

- in_file
- out_file – if this is left empty, the in_file name will be used as the basis for building the output file names, otherwise this string will be used. If there is a .rtdf extension at the end of this name, it will be removed and then _condX.rtdf will be appended where X will be a number incrementing from 1 for the number of unique conditions found in the in_file.

3.3.14 SplitSites

This script will take a multisite RTDF file and split it into separate files based on site.

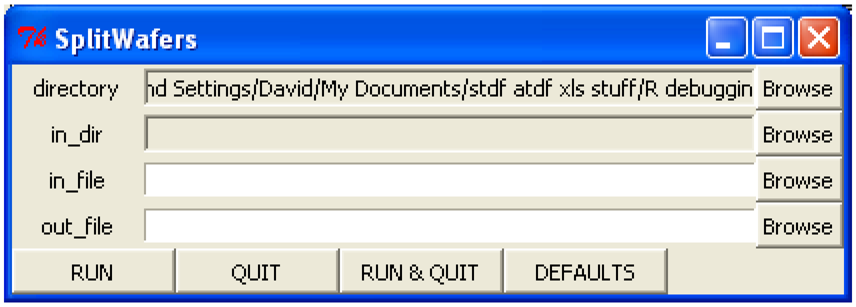


The various fields in detail:

- directory
- in_dir
- in_file
- out_file – if this is left empty, the in_file name will be used as the basis for building the output file names, otherwise this string will be used. If there is a .rtdf extension at the end of this name, it will be removed and then _siteXX.rtdf will be appended for site XX.

3.3.15 SplitWafers

This script will take a multiwafer RTDF file and split it into separate files based on wafer.



The various fields in detail:

- directory
- in_dir
- in_file
- out_file – if this is left empty, the in_file name will be used as the basis for building the output file names, otherwise this string will be used. If there is a .rtdf extension at the end of this name, it will be removed and then _waferXX.rtdf will be appended for wafer XX.

3.4 Plots and Statistics

There are currently 7 scripts in this category, as follows:

- AsciiWaferMap

  This script will generate a text based wafer map in a format that is accepted by most assembly houses.

- ControlCharts

  This script will generate SPC control charts (aka Shewart charts or process behaviour charts) for each of the parameters in an RTDF file. For more details, see the ControlCharts section.

- PlotRtdf

  This script will generate histograms in PDF and statistics in CSV for one or more RTDF files. It can also generate XY plots for multiple RTDF files where they have the same part_id's, such as characterization data at different temperatures, or various hour samples from 1000 hour life testing. For more details, see the PlotRtdf section.

- PlotTestvsTest

  This script will generate XY plots between 2 different tests for the specified RTDF file. For more details, see the PlotTestvsTest section.

- PlotVsRun

  This script generates plots of value on the Y axis and device run number on the X axis. For more details, see the PlotVsRun section.

- WaferMap

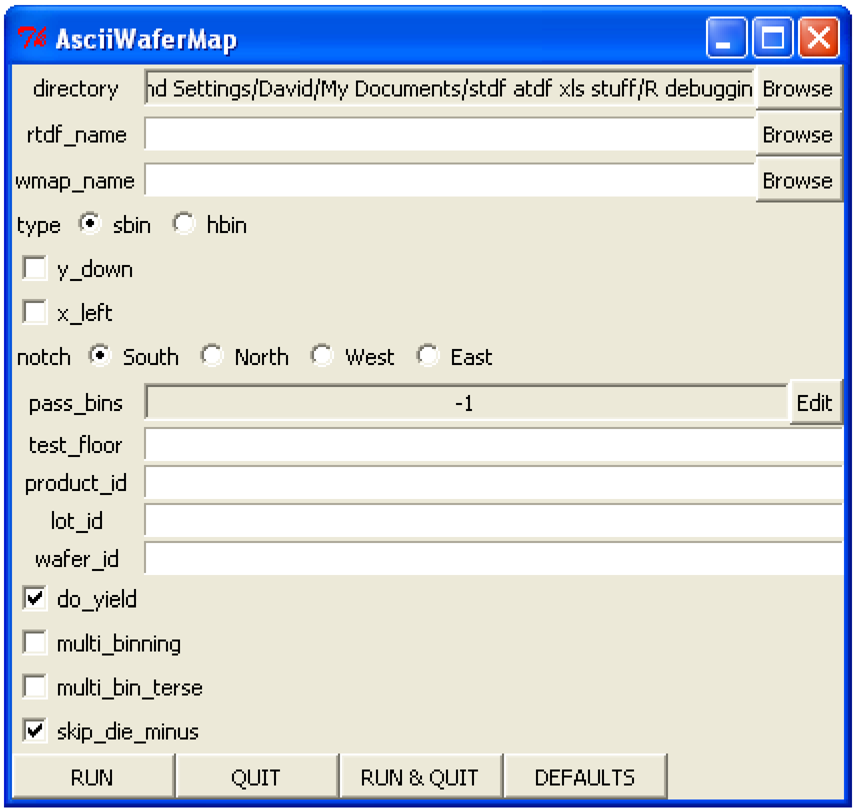
  This script can generate wafer maps from RTDF files that contain valid wafer coordinates. It can generate soft binning maps, hard binning maps, or parametric test maps. For more details, see the WaferMap section.

- XformWaferMap

  This script is a superset of the WaferMap above. It is used when there is a desire to have the wafermap coordinates displayed in a manner different than what the prober/probe card have defined in the stdf file. For more details, see the section.

3.4.1 AsciiWaferMap

This script will generate a text based wafer map in a format that is accepted by most assembly houses.


AsciiWaferMap

directory: nd Settings/David/My Documents/stdf atdf xls stuff/R debuggin [Browse]

rtdf_name: [Browse]

wmap_name: [Browse]

type: [● sbin] ○ hbin

☐ y_down

☐ x_left

notch: [● South] ○ North ○ West ○ East

pass_bins: -1 [Edit]

test_floor:

product_id:

lot_id:

wafer_id:

☑ do_yield

☐ multi_binning

☐ multi_bin_terse

☑ skip_die_minus

The various fields in detail:

- directory – the directory in which the .wmap file will be created
- rtdf_name – the name of the rtdf file to process when generating the .wmap file
- wmap_name – the name of the ascii file generated. If this field is left empty, the script will automatically generate the name based on the lot and wafer id.
- type – select either the soft binning or the hard binning to determine which die should be assembled.
- y_down – If the wafer Y coordinates increase in the down direction, then select this option.
- x_left – If the wafer X coordinates increase in the left direction, then select this option.
- notch – indicate the orientation of the notch for the probing session that generated the .rtdf file. The .wmap file will then be rotated accordingly so the notch is at the bottom
- pass_bins – If this is left as -1, then the pass bins will be determined by the information within the rtdf file, or if the pass/fail status of the bins isn't in the rtdf file, it will assume bin 1 is the pass bin. You can also explicitly supply a list of bins here. If you had a situation where you had low yield due to one particular bin that you wanted to package anyways, you would then enter that bin along with the pass bin(s), so that the resultant .wmap file would indicate all these die should be assembled.
- test_floor – This is the string that will go into the "test floor" field of the .wmap file.
- product_id – This is the string that will go into the "product id" field of the .wmap file.
- lot_id – If this field is left empty, it will be extracted from the rtdf file. Otherwise, whatever is

put here will override the lot_id information as found in the rtdf file

- wafer_id – If this field is left empty, it will be extracted from the rtdf file. Otherwise, whatever is put here will override the wafer_id information as found in the rtf file
- do_yield – this will add a line at the end of the .wmap file that gives the yield as a percentage and shows the good and total die counts. It is not a requirement for assembly, but it can be useful.
- multi_binning – If this box is checked, then the wafer map will contain either the soft binning or the hard binning for each die instead of just pass/fail status. This would be used in situations where you are assembling multiple groups of devices, either where you have multiple variants of a part, or have some speed sorting where the parts are either fast/nominal/slow, or where you want to assemble the goods in one group, as well as a particular failure type in another group.
- multi_bin_terse – If this box is checked, then the different pass bins will be assigned numbers starting at 1, and the failing die will be marked with "x" in the ascii map.
- skip_die_minus – for die that are not probed, the default character is "." Optionally, die that are not probed but are surrounded by probed die can be replaced with the "-" character. This would be used by some pick and place machines to signify unique test structures on the wafer that the machine can then use for alignment as compared to "." which may or may not be even on the wafer

3.4.1.1 AsciiWaferMap Output

Below is an example Ascii Wafer Map:

```
Test Floor:  my_floor
Product Id:  a595
Lot Id:      demo
Wafer Id:    02
Finish Time: 2002-12-02 09:37:41
<Bin_Map>
.................
......x11x.......
....xx11x1x1.....
...xx11111111....
...1xxx11111x....
..xx111111111x...
..1x111111111x...
..111111111111...
..111111x1x111...
...111xx111x1....
...11111111x1....
....1x111111.....
......1111.......
```

```
   Notch Down
<\Bin_Map>

Yield: 78.6 percent [88/112]
```

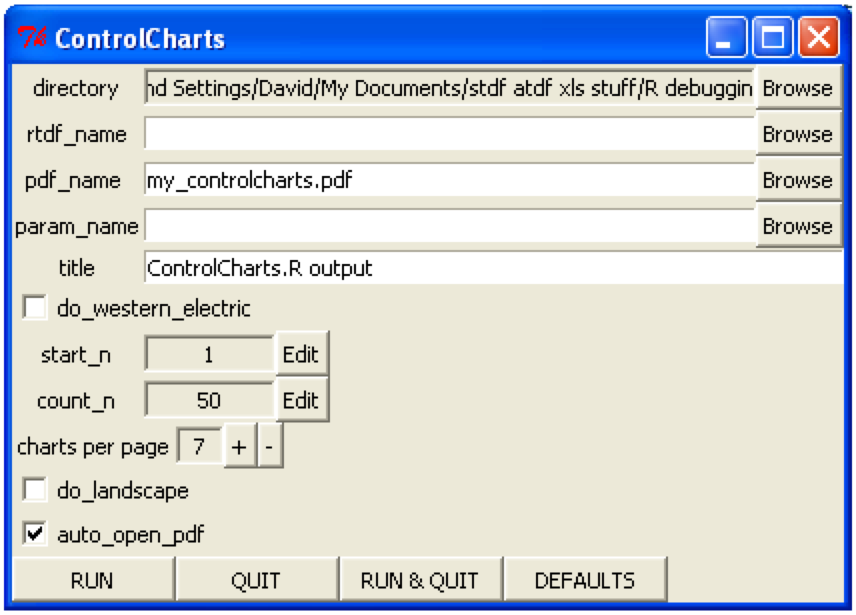
Below is an example of the same data, but using the multi_binning option of Ascii Wafer Map:

```
Test Floor:  my_floor
Product Id:  a595
Lot Id:      multi_demo
Wafer Id:    02
Finish Time: 2002-12-02 09:37:41
<Bin_Map>
.. .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
.. .. .. .. .. .. 10 01 01 10 .. .. .. .. .. .. ..
.. .. .. .. 10 11 01 01 04 01 03 01 .. .. .. .. ..
.. .. .. 10 03 01 01 01 01 01 01 01 01 .. .. .. ..
.. .. .. 01 03 04 04 01 01 01 01 01 04 .. .. .. ..
.. .. 10 04 01 01 01 01 01 01 01 01 01 04 .. .. ..
.. .. 01 04 01 01 01 01 01 01 01 01 01 04 .. .. ..
.. .. 01 01 01 01 01 01 01 01 01 01 01 01 .. .. ..
.. .. 01 01 01 01 01 01 04 01 11 01 01 01 .. .. ..
.. .. .. 01 01 01 08 08 01 01 01 04 01 .. .. .. ..
.. .. .. 01 01 01 01 01 01 01 01 04 01 .. .. .. ..
.. .. .. .. 01 04 01 01 01 01 01 01 .. .. .. .. ..
.. .. .. .. .. .. 01 01 01 01 .. .. .. .. .. .. ..
                    Notch Down
<\Bin_Map>

Bin: Sbin  Count  Yield  P/F  Bin_Name
Bin:   1     88   78.6%   P   PASS
Bin:   3      3    2.7%   F   FUNC
Bin:   4     12   10.7%   F   DELAY
Bin:   8      2    1.8%   F   VIN
Bin:  10      5    4.5%   F   CONT
Bin:  11      2    1.8%   F   POWER
```

```
Yield: 78.6 percent [88/112]
```

3.4.2 ControlCharts

This script will generate SPC control charts (aka Shewart charts or process behaviour charts) for each of the tests/parameters in an RTDF file.



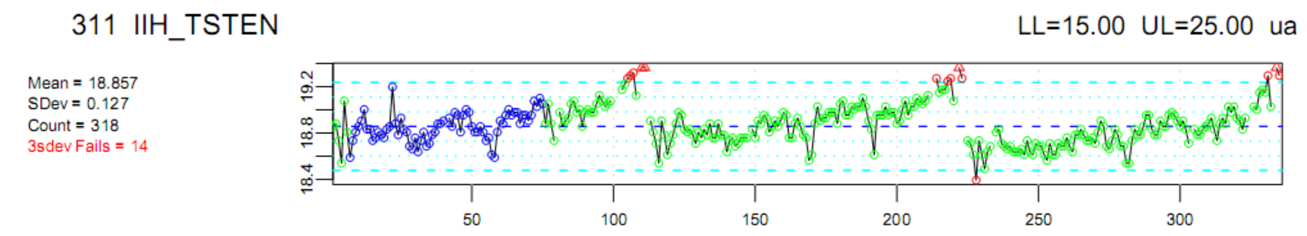
directory: nd Settings/David/My Documents/stdf atdf xls stuff/R debuggin
rtdf_name:
pdf_name: my_controlcharts.pdf
param_name:
title: ControlCharts.R output
[ ] do_western_electric
start_n: 1
count_n: 50
charts per page: 7
[ ] do_landscape
[✓] auto_open_pdf

The various fields in detail:

- directory
- rtdf_name – the name of the RTDF file to extract the data from for generating the charts
- pdf_name – the name of the PDF file created that contains the charts
- param_name – the name of a file containing an RTDF formatted ParametersFrame. If this is specified, then this file is used to determine which tests and what limits are used for plotting. If this is empty, the script will use the ParametersFrame in the RTDF file specified in rtdf_name above to determine the tests and limits used for plotting.
- title – a string, the title to appear at the bottom of each page.
- do_western_electric – setting this flag on will enable the application of the Western Electric Rules to the charts. There will be an extra line in the statistics section showing the number of devices flagged by the Western Electric Rules, and these devices will be coloured orange in the charts.
- start_n – To calculate the mean and standard deviation for each test/parameter, a subset of the data points are used. These data points are the ones starting at the start_n index, and including count_n points. The mean and standard deviation are the robust mean and standard deviation.
- count_n – see start_n above
- charts_per_page – this determines how many charts will be fit onto an 8.5x11 inch page.

- do_landscape – setting this flag will change the page orientation from portrait to landscape.
- auto_open_pdf - if this flag is set, as soon as the PDF file has been created, it will be opened using your default PDF reader.

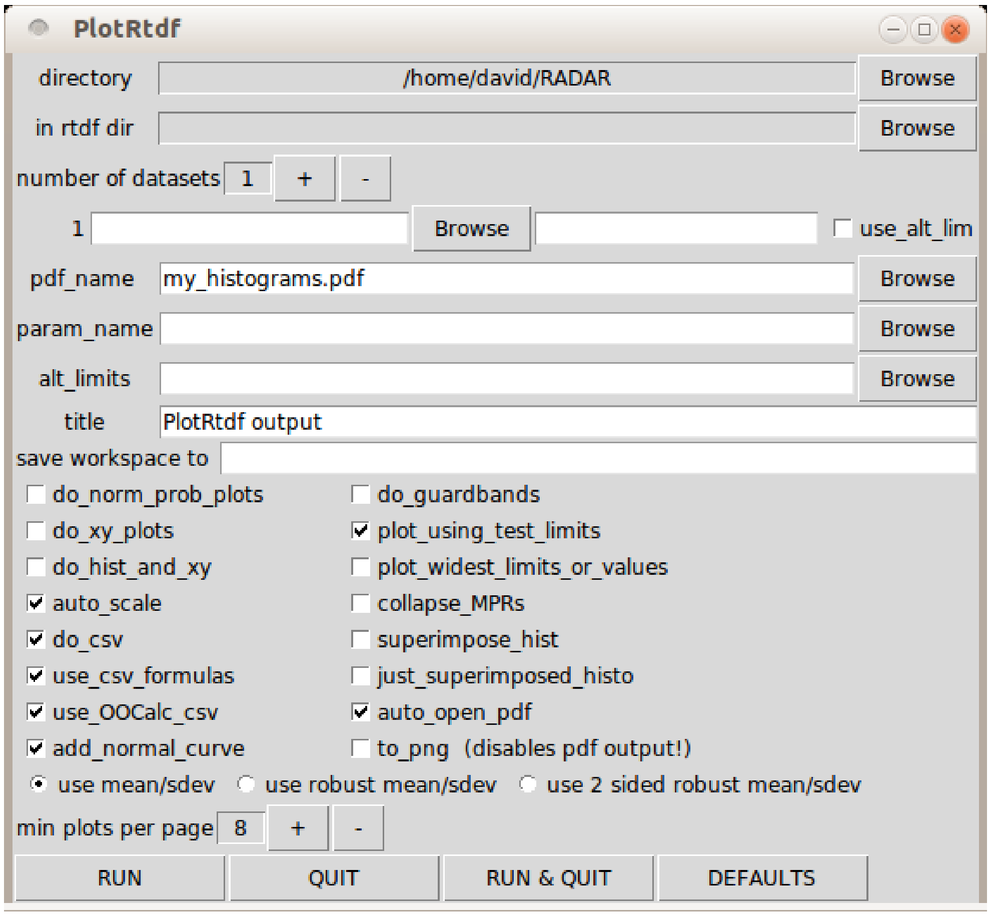
3.4.2.1 ControlCharts Output



The above control chart is from an RTDF file that contains 3 wafers of data for a parameter that trends across a wafer. The blue circles indicate the points used when deriving the mean and standard deviation. In this example, start_n was set to 7 and count_n was set to 69. The red circles indicate the points that lie outside the +/- 3 standard deviations. The red triangles indicate parts that are outside of the chart region. The script automatically truncates the chart at the +/- 4 standard deviations to stop wild data from ruining the scale of the chart.

3.4.3 PlotRtdf

This script will take one or more RTDF files and generate histograms or normal probability plot and/or xy plots and optionally a CSV file with the per plot statistics for all the tests. The output can be a single PDF file or a per-test PNG file.



directory /home/david/RADAR
in rtdf dir
number of datasets 1
1 use_alt_lim
pdf_name my_histograms.pdf
param_name
alt_limits
title PlotRtdf output
save workspace to

do_norm_prob_plots
do_xy_plots
do_hist_and_xy
[✓ auto_scale]
[✓ do_csv]
[✓ use_csv_formulas]
[✓ use_OOCalc_csv]
[✓ add_normal_curve]
do_guardbands
[✓ plot_using_test_limits]
plot_widest_limits_or_values
collapse_MPRs
superimpose_hist
just_superimposed_histo
[✓ auto_open_pdf]
to_png (disables pdf output!)

[✓ use mean/sdev] use robust mean/sdev use 2 sided robust mean/sdev

min plots per page 8

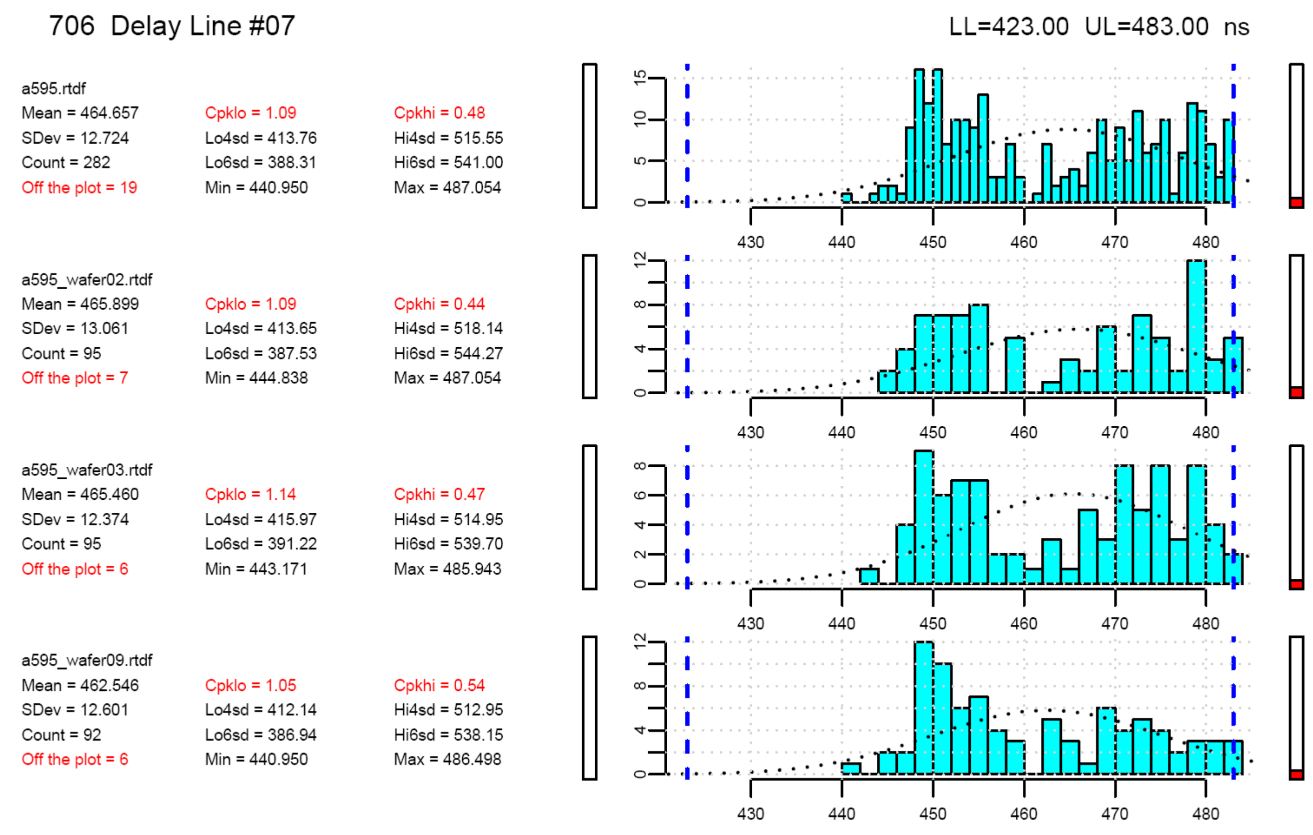
The various fields in detail:

- directory – the directory where the generated files go.
- in rtdf dir – the directory to use when looking for RTDF files
- number of datasets – you can select multiple RTDF files and each file will have its own histogram on a per test basis
- datasets – the list of RTDF files you wish to plot. The first field is the RTDF file name. The next field is an optional character string that will be shown on the plots rather than the file name. If the use_alt_lim checkbox is checked and there is an RTDF file entered in the alt_limits field lower down, this dataset's Cpk values will be relative to the limits found in the alternate file.
- pdf_name – the name of the PDF file created that contains the plots
- param_name – the name of a file containing an RTDF formatted ParametersFrame. If this is specified, then this file is used to determine which tests and what limits are used for plotting. If this is empty, the script will use the first dataset RTDF file's ParametersFrame to determine the tests and limits used for plotting, or if a test isn't present in the first RTDF, the script will use the first RTDF file that does have the test to determine units and limits.
- alt_limits – the name of a file containing an RTDF formatted ParametersFrame. If this is specified, a second set of limits will be added to the histograms. If the use_alt_limits checkbox is set for a particular dataset, its Cpk statistics will be based on the alternate limits.
- title – a string, the title to appear at the bottom of each page. If the field is empty, "PlotRtdf" will be used.
- save workspace to – if specified, this is the .Rdata file that the workspace with all the datasets will be saved to. If empty, no .Rdata formatted file will be generated.
- do_norm_prob_plots – setting this will cause the script to generate normalized probability plots rather than the default histograms.
- do_xy_plots – setting this will cause the 2nd to nth datasets to be plotted as xy plots relative to the first dataset based on matching part_ids. If there are 10 or more data points, PlotRtdf will add a linear fit line to the XY plot.
- do_hist_and_xy – setting this will cause both histograms and XY plots to be generated for the 2nd to nth datasets.
- auto_scale – override the scaler extracted from the tester. Instead, base the scaling on the measurement range and limit range. (ie. decide if current tests should be displayed in mA or uA or nA)
- do_csv – if this flag is set, a file with the same base name as the pdf_name, but with a ".csv" extension will be created containing a more complete list of statistics than what is displayed to the left of the histograms in the PDF file.
- use_csv_formulas – if this flag is set, and the do_csv flag is set, then, in the generated CSV file, the Cpk numbers will be formulas based on the standard deviation, mean and limits rather than hardcoded values. This allows you to then play with limits in the table afterwards and see the immediate impact on Cpks.
- use_OOCalc_csv – if the use_csv_formulas flag above is set, then this flag will put the formulas in the format that OpenOffice Calc expects in a CSV file. Otherwise, it puts the formulas in the format expected by Excel.

- add_normal_curve – if this flag is set, a normal curve based on the mean, standard deviation, and sample size will be calculated and overlaid on top of the histogram
- use mean/sdev OR use robust mean/sdev OR use 2 sided robust mean/sdev – this radio button determines which statistics will be shown to the left of the histograms when generating histograms. The .csv file generated will always include all 3 calculations.
- do_guardbands – if this flag is set, the script will generate a guardband.csv file. If number of datasets is less than 2, or xy_plots is not set, this option is ignored. Dataset1 should use limits and the other datasets should use alt_limits. the script assumes datasets are for the same devices at different conditions, ie. dataset 1 is final/room, dataset 2 is lowV/hot, dataset3 is highV/cold. The xy linear fit should have an R>0.9 for the guardband calculation for a specific test to be meaningful.
- plot_using_test_limits – if this flag is set, the upper and lower limits of the histogram plot are determined by the limits. If a limit is missing, then the most extreme measurement is used. When this flag is not set, the minimum and maximum measurement values are used for the plot limits.
- plot_widest_limits_or_values – if this flag is set, each histogram plot will be scaled to whichever is the widest, the limits or the min/max values.
- collapse_MPRs – if this flag is set, adjacent tests in the testlist that have the same test number and same testnames upto the last slash are combined into a single histogram rather than as separate histograms. When MPR records are encountered in the ConvertStdf script, a slash and the pinname is appended to the testname and stored in the RTDF file as a unique test. This flag has the effect of 'collapsing' all the pins into a single histogram.
- superimpose_hist – This flag will add an extra histogram plot that will include all the subplots on a single histogram.
- just_superimposed_histo – This flag will suppress individual dataset histograms and just plot a single histogram plot with the various datasets superimposed.
- auto_open_pdf – If this flag is set, once the script has finished generating the PDF file, The default application for PDF files will be invoked to open this file. Warning: occasionally, for large files, the PDF file isn't actually finished when the PDF write process claims it is done and so opening the file will fail.
- to_png – if this flag is set, a PNG file will be generated per test parameter instead of a single PDF file. NOTE: this does run noticably slower than PDF file generation.
- min plots per page – sets how many individual histograms one wants to cram onto a single page. If it is set to 3, and you have 2 datasets then you would get 2 tests/parameters per page.

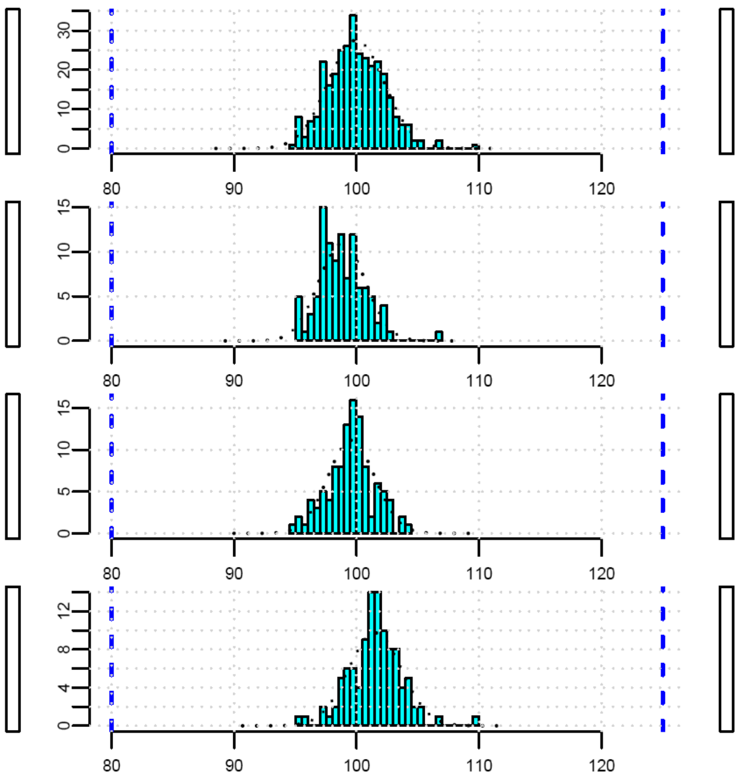
3.4.3.1 PlotRtdf Output Histograms

Below are 2 examples of what the histograms look like. In the first case, we have a situation where the testing is cutting a distribution, and in the second case we have a more ideal set of data vs. limits. The first line contains the test number and test name at the left and the limits and units at the right. For each histogram, there is a text section to the left and the histogram to the right. In the text section, the first line is the either the name of the rtdf file, or if a different title is provided, the title associated with that file. The mean and standard deviation are based on all the parts, not just the parts within the plot area. The count reflects this number. If there are parts outside the plot area, then the "Off the plot" field is red highlighted and the count is shown. Off the plot parts are represented in the histogram portion by the 2 narrow boxes on either side of the histogram plot. The box is proportionally filled with red representing the % of parts that are either lower or higher in value than the histogram. In this case, about 7% of the parts are above the upper limit of the histogram. The next 2 columns of text are some key values related to the lower and upper limits; the Cpk for each limit, highlighted in red if it is below 1.33, and the 4 and 6 sigma points [ie limits to get Cpk of 1.33 and 2.00], and the min or max measurement. In the histogram are, just to the right of the histogram is a Y-axis that is in # of devices. Blue dashed vertical lines show the limits, and if there is an alternate set of limits, these are shown with green dashed lines.



a595.rtdf		
Mean = 464.657	Cpklo = 1.09	Cpkhi = 0.48
SDev = 12.724	Lo4sd = 413.76	Hi4sd = 515.55
Count = 282	Lo6sd = 388.31	Hi6sd = 541.00
Off the plot = 19	Min = 440.950	Max = 487.054
a595_wafer02.rtdf		
Mean = 465.899	Cpklo = 1.09	Cpkhi = 0.44
SDev = 13.061	Lo4sd = 413.65	Hi4sd = 518.14
Count = 95	Lo6sd = 387.53	Hi6sd = 544.27
Off the plot = 7	Min = 444.838	Max = 487.054
a595_wafer03.rtdf		
Mean = 465.460	Cpklo = 1.14	Cpkhi = 0.47
SDev = 12.374	Lo4sd = 415.97	Hi4sd = 514.95
Count = 95	Lo6sd = 391.22	Hi6sd = 539.70
Off the plot = 6	Min = 443.171	Max = 485.943
a595_wafer09.rtdf		
Mean = 462.546	Cpklo = 1.05	Cpkhi = 0.54
SDev = 12.601	Lo4sd = 412.14	Hi4sd = 512.95
Count = 92	Lo6sd = 386.94	Hi6sd = 538.15
Off the plot = 6	Min = 440.950	Max = 486.498

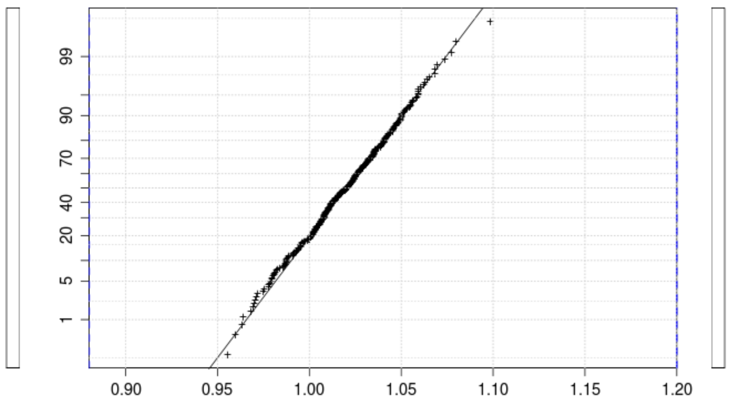
```
230  IDD1 @ 40MHz                                                    LL=80.00  UL=125.00  ma

a595.rtdf
Mean = 99.968        Cpklo = 2.91        Cpkhi = 3.64
SDev = 2.290         Lo4sd = 90.81       Hi4sd = 109.13
Count = 318          Lo6sd = 86.23       Hi6sd = 113.71
Off the plot = 0     Min = 94.874        Max = 109.870

a595_wafer02.rtdf
Mean = 98.825        Cpklo = 3.29        Cpkhi = 4.57
SDev = 1.909         Lo4sd = 91.19       Hi4sd = 106.46
Count = 105          Lo6sd = 87.37       Hi6sd = 110.28
Off the plot = 0     Min = 95.318        Max = 106.822

a595_wafer03.rtdf
Mean = 99.566        Cpklo = 3.41        Cpkhi = 4.43
SDev = 1.914         Lo4sd = 91.91       Hi4sd = 107.22
Count = 107          Lo6sd = 88.08       Hi6sd = 111.05
Off the plot = 0     Min = 94.874        Max = 104.219

a595_wafer09.rtdf
Mean = 101.505       Cpklo = 3.32        Cpkhi = 3.62
SDev = 2.161         Lo4sd = 92.86       Hi4sd = 110.15
Count = 106          Lo6sd = 88.54       Hi6sd = 114.47
Off the plot = 0     Min = 95.227        Max = 109.870
```



3.4.3.2 *PlotRtdf Output Normalized Probability Plots*

Below are 2 examples of Normalized Probability plots, the first for a test with a typical normal distribution and the second for a test that is not so normal.

```
541  VOH_DTO2P                                                        LL=0.88  UL=1.20  v

a595.rtdf

Mean = 1.020         Cpklo = 1.93        Cpkhi = 2.49

SDev = 0.024         Lo4sd = 0.92        Hi4sd = 1.12

Count = 310          Lo6sd = 0.87        Hi6sd = 1.16

Off the plot = 0     Min = 0.955         Max = 1.098
```

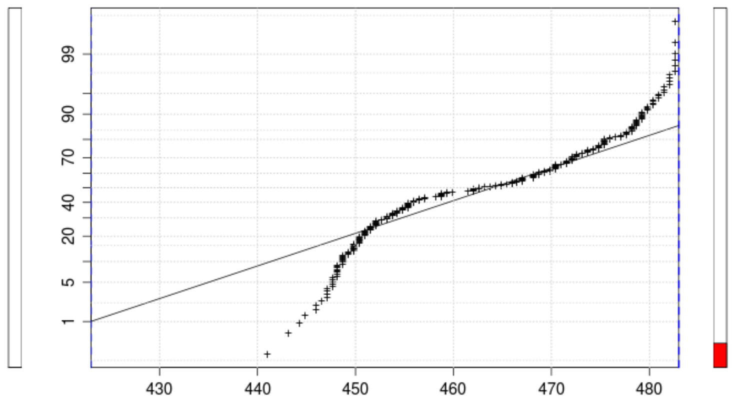


```
PlotRtdf output   2012-01-26 21:46:24                                          Page 63
```

706 Delay Line #07 LL=423.00 UL=483.00 ns

a595.rtdf

Mean = 464.657	Cpklo = 1.09	Cpkhi = 0.48
SDev = 12.724	Lo4sd = 413.76	Hi4sd = 515.55
Count = 282	Lo6sd = 388.31	Hi6sd = 541.00
Off the plot = 19	Min = 440.950	Max = 487.054

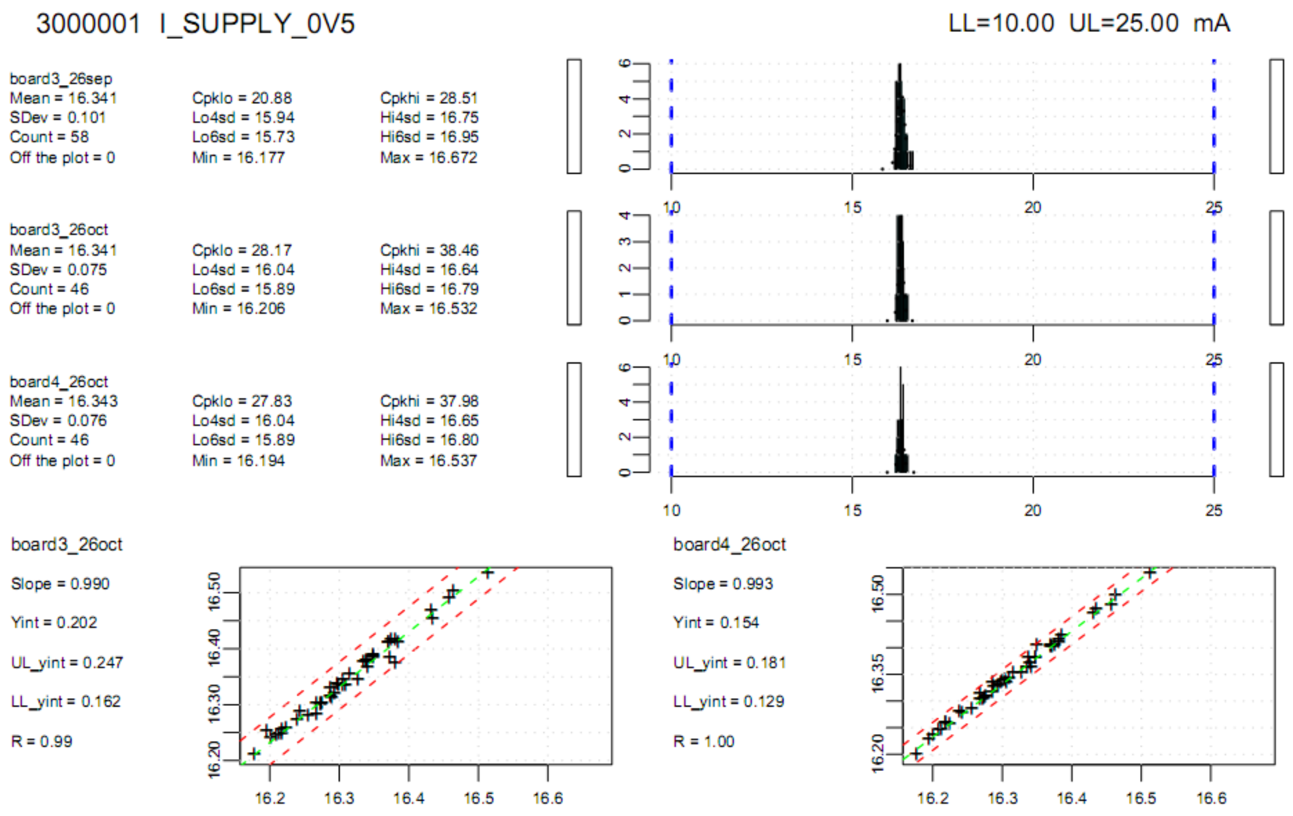


PlotRtdf output 2012-01-26 21:46:24 Page 82

3.4.3.3 *PlotRtdf Output X-Y Plots*

The below shows an example of a loadboard correlation exercise showing a set of correlation devices run on one loadboard, and a month later rerun on the same loadboard and a new loadboard as part of the qualification exercise for the new loadboard before releasing it to production. For the XY plots, a green line shows the linear fit, and the red lines are the +/- 4 sigma of the linear fit. To the left of the XY plots are the statistics:

- slope – the slope of the linear fit line
- Yint – Y intercept of the linear fit line
- UL_Yint – the Y intercept for the line that is 4 sigma above the linear fit
- LL_Uint – the Y intercept for the line that is 4 sigma below the linear fit
- R – the correlation of the linear fit to the data. A perfect fit would be 1.0. Values above 0.95 indicate that the linear fit is fairly good and hence useful.

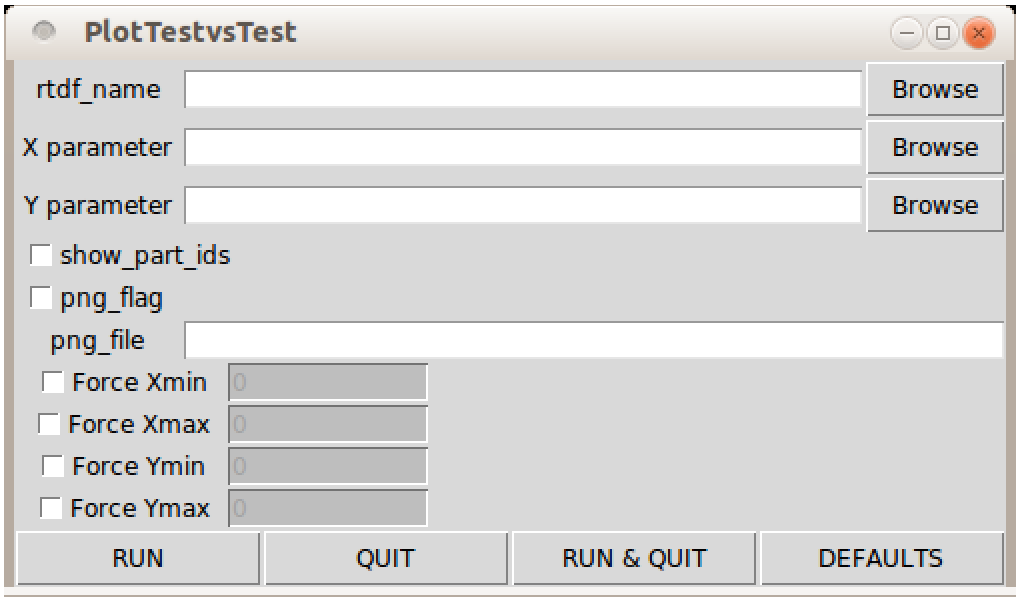


The +/- 4 sigma linear fit lines can be used for generating guardbands between Final and QC tests. From the above example, one would set the Final limits to 15.7 < <17.0mA to get Cpk of >=2.0. QC limits would be relaxed to 15.6< <17.1mA to cover test/tester/loadboard repeatability.

3.4.3.4 *PlotRtdf Output CSV file*

3.4.4 PlotTestvsTest

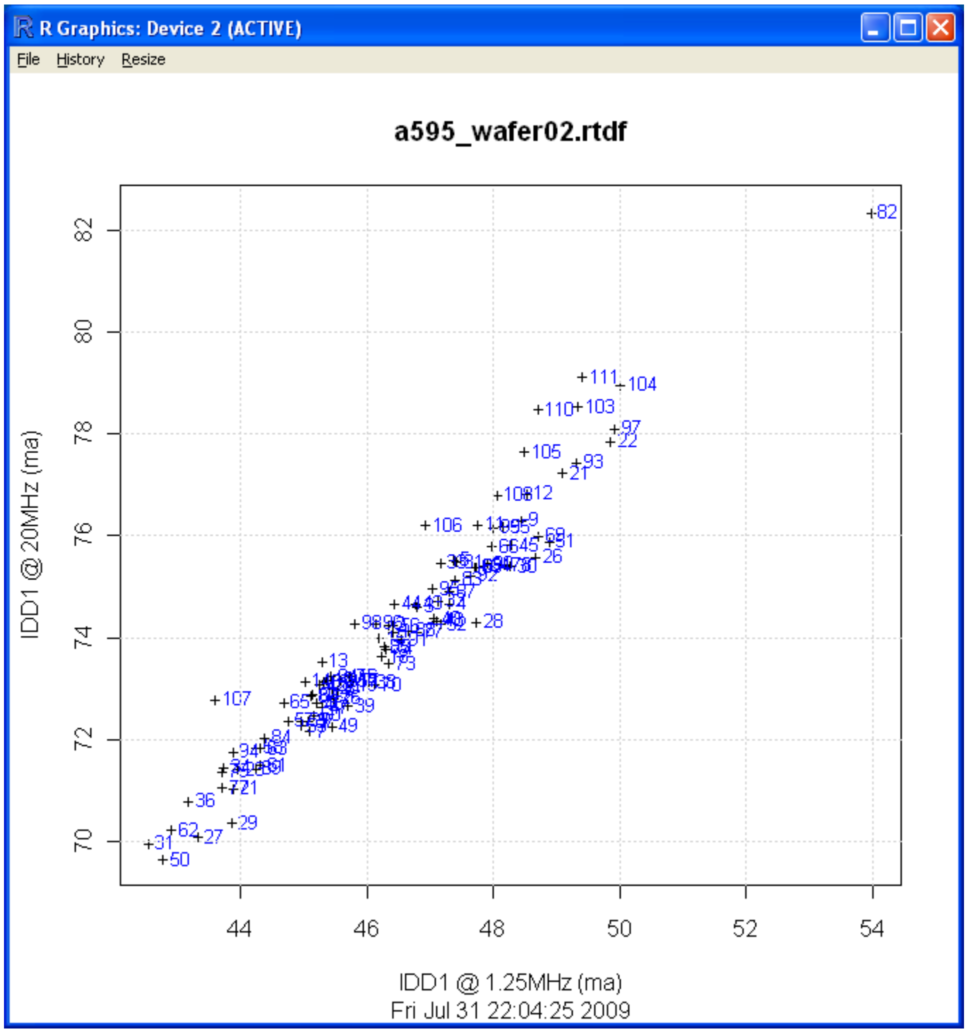
This script will generate an XY plot between 2 different tests for a given RTDF file.



The various fields in detail:

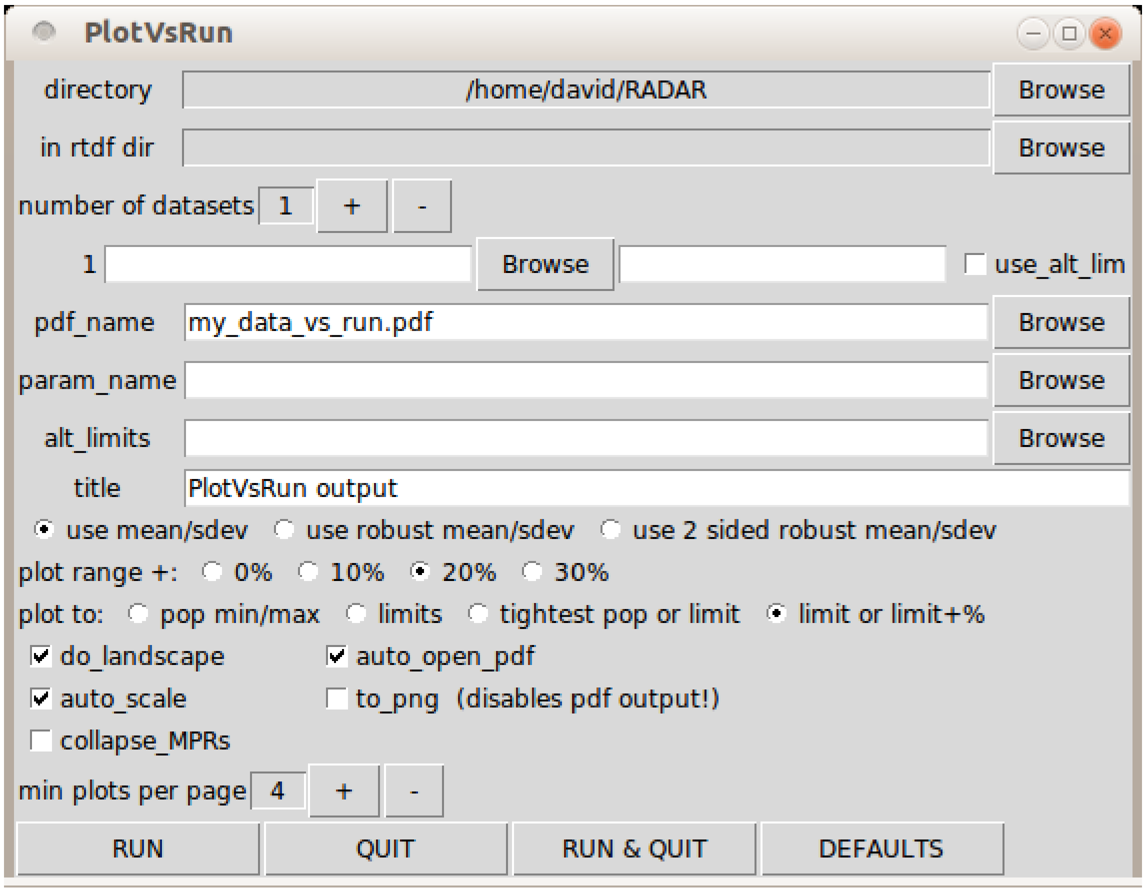
- rtdf_name
- X parameter
- Y parameter
- show_part_ids – if this box is checked, then the XY plot will be annotated with the part_id for each of the devices plotted. The example below was done with this box checked.
- png_flag – if this flag is set, the xy plot will be saved to a PNG file as well as displayed in a window on your computer screen.
- png_file – if png_flag is set and this field is not empty, the xy plot will be saved to a PNG file with this name. If the png_flag is set and this field is empty, the filename will be automatically created along the lines of testX_vs_testY.png where testX and testY are the names of the 2 tests/parameters selected.
- Force Xmin – by default, the plot will auto-scale to include all the finite X-Y pairs of values. The Xmin value of the plot can be explicitly set by clicking this button and entering a valid number in the entry box beside the select button. NOTE: The value is in the scaled units. In the example output below, if you wanted to ignore device 82, you could set the Ymax to 80 (implicit ma) not 0.080 (amperes).
- Force Xmax / Ymin / Ymax – Similar to above.

3.4.4.1 *PlotTestvsTest Output (Graphics window)*



3.4.5 PlotVsRun

This script generates plots for each test with values in the Y axis and device/run count in the X axis.



directory /home/david/RADAR Browse

in rtdf dir Browse

number of datasets 1 + -

1 Browse ☐ use_alt_lim

pdf_name my_data_vs_run.pdf Browse

param_name Browse

alt_limits Browse

title PlotVsRun output

[✓ use mean/sdev] use robust mean/sdev use 2 sided robust mean/sdev

plot range +: 0% 10% [✓ 20%] 30%

plot to: pop min/max limits tightest pop or limit [✓ limit or limit+%]

☑ do_landscape ☑ auto_open_pdf

☑ auto_scale ☐ to_png (disables pdf output!)

☐ collapse_MPRs

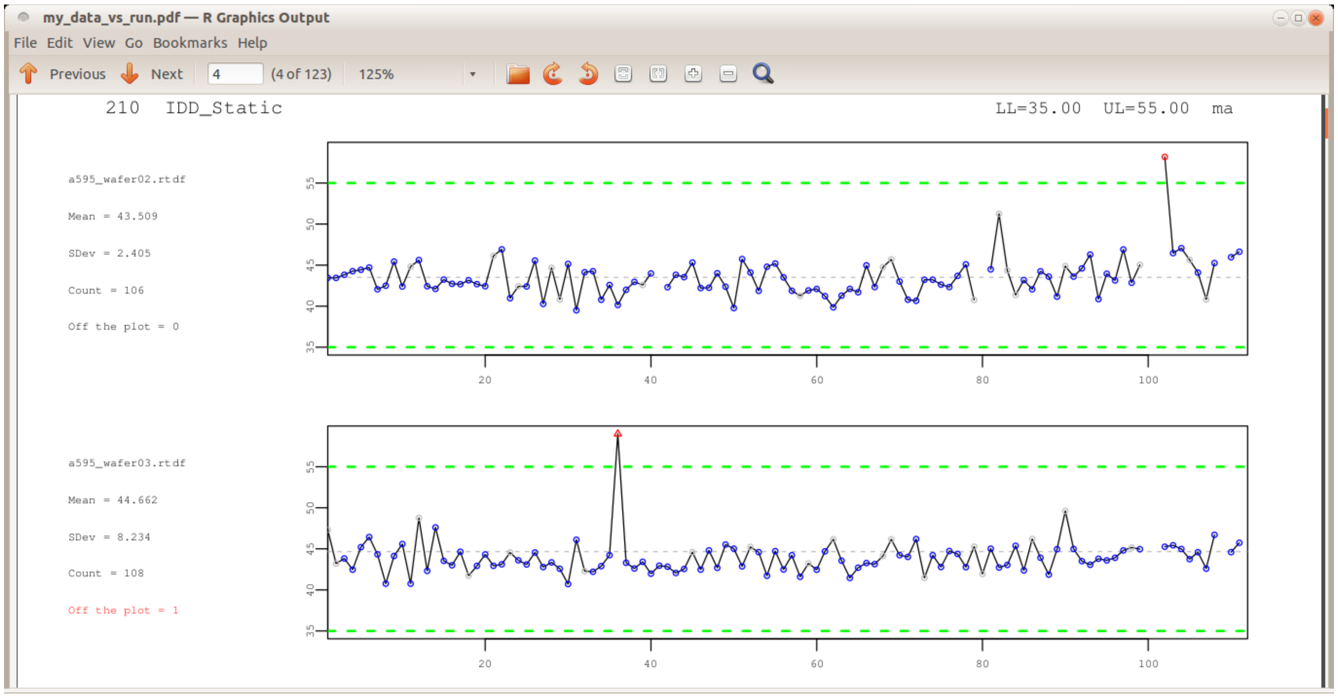
min plots per page 4 + -

RUN QUIT RUN & QUIT DEFAULTS

The various fields in detail:

- directory
- in rtdf dir
- number of datasets

3.4.5.1 Example PlotVsRun plot

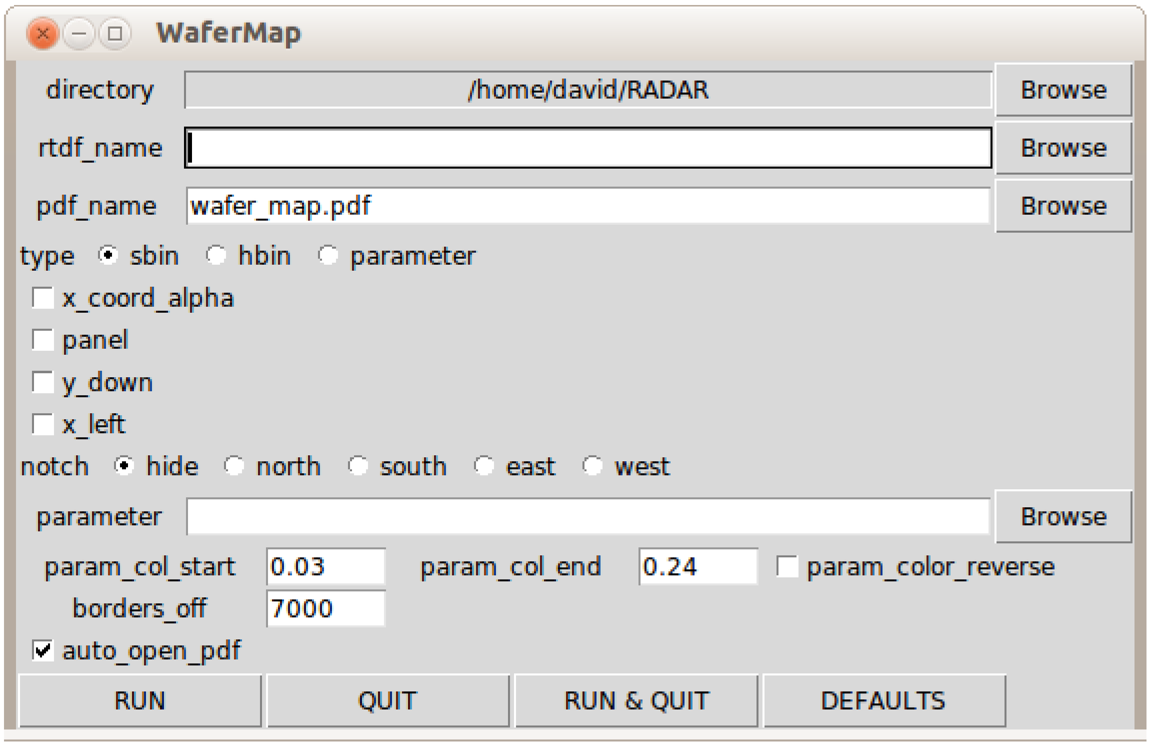


The measurement symbols signify:

- A blue circle – this device is a passing device
- A grey circle – this device passes this test, but fails some other test(s)
- A red circle – this device fails this test
- A red triangle – this device fails this test and is above/below the range of this plot

3.4.6 WaferMap

This script can generate wafer maps from RTDF files that contain valid wafer coordinates. It can generate soft binning maps, hard binning maps, or parametric test maps.


WaferMap

directory /home/david/RADAR Browse

rtdf_name Browse

pdf_name wafer_map.pdf Browse

type [✓ sbin] hbin parameter

☐ x_coord_alpha

☐ panel

☐ y_down

☐ x_left

notch [✓ hide] north south east west

parameter Browse

param_col_start 0.03 param_col_end 0.24 ☐ param_color_reverse

borders_off 7000

☑ auto_open_pdf

RUN QUIT RUN & QUIT DEFAULTS

The various fields in detail:

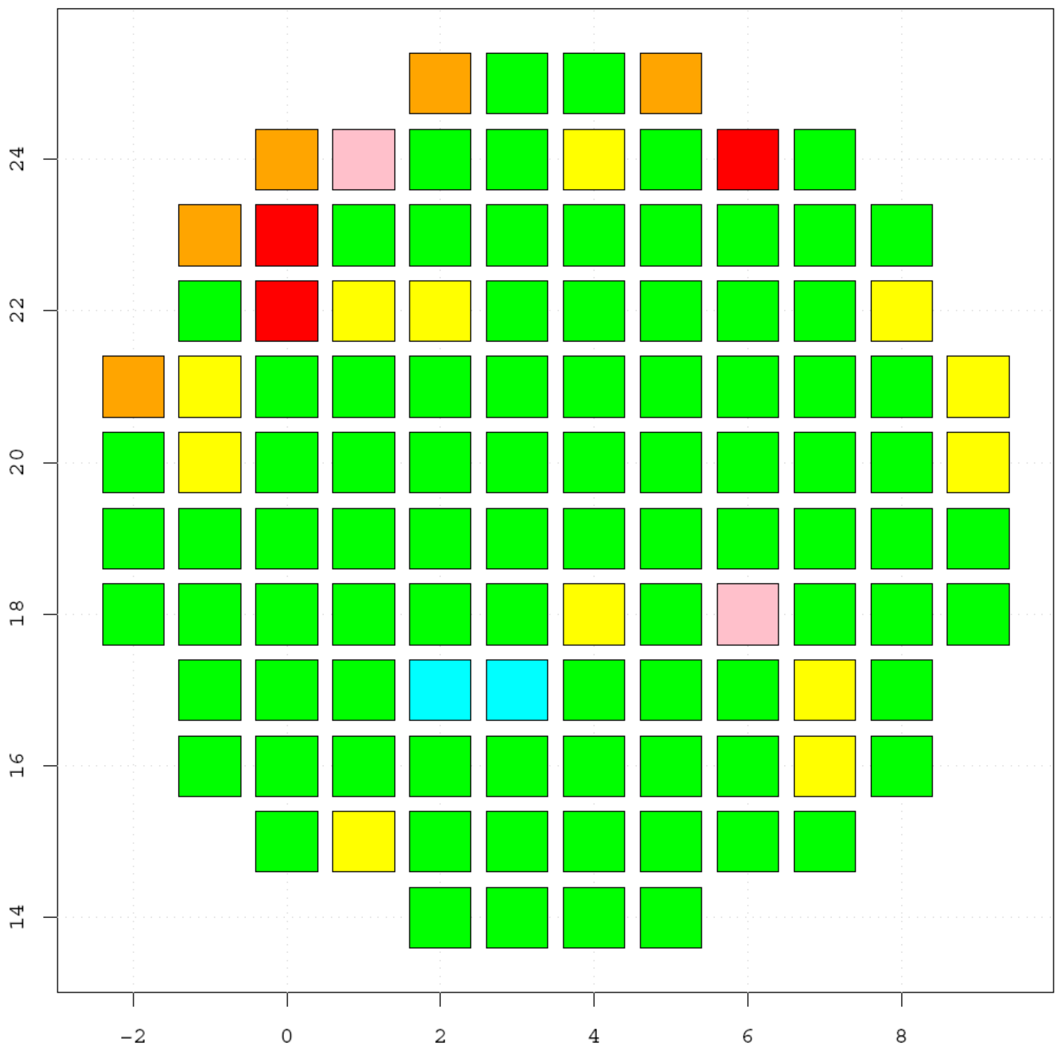
- directory
- rtdf_name
- pdf_name
- type – the type of wafermap you want to generate, either sbin aka soft binning, hbin aka hard binning, or parametric, which will do a gradient map based on the measurements for the selected test from the parameter field below.
- x_coord_alpha – show X coordinates as A,B,C,... rather than 1,2,3... . Tapestry strip handler uses alphabetic coordinates in the X direction, but STDF files convert these to numbers.
- panel - the changes the page from portrait to landscape, so that the wafermap area becomes a long rectangle rather than a square. Use when looking at strip handler panels.
- y_down – If the wafer Y coordinates increase in the down direction, then select this option.
- x_left – If the wafer X coordinates increase in the left direction, then select this option.
- Notch – can add the word "NOTCH" to the map, at the location specified
- parameter – if the wafer map type is parametric, then the testname of the parameter is entered here, and a single wafermap per wafer will be generated. If this field is left empty, then a wafermap for each parameter will be generated.
- param_col_xxx – the color scheme used for the parametric wafer maps can be overridden. By default, the lowest value is assigned to red and the highest value is assigned to green. The underlying colors come from the R language rainbow() command. If you want to change the start or stop colors, or the direction around the color wheel, these 3 overrides are available. The rainbow values are numbers in the range of 0.0 to 1.0, with 0.0 being red, and 0.24 being green.
- borders_off – by default, the plot function puts a black rectangular border around each die it

plots in the wafer map. For wafers with small die, this can dominate the individual die, so once the die count gets to 7000, the border colour is changed from black to the die fill colour. If you always want the black border, you can set this value to -1. If you never want the black border, you can set this value to 0. If you want to set the automatic switch point to something other than 7000, you can overwrite this value with your preference.

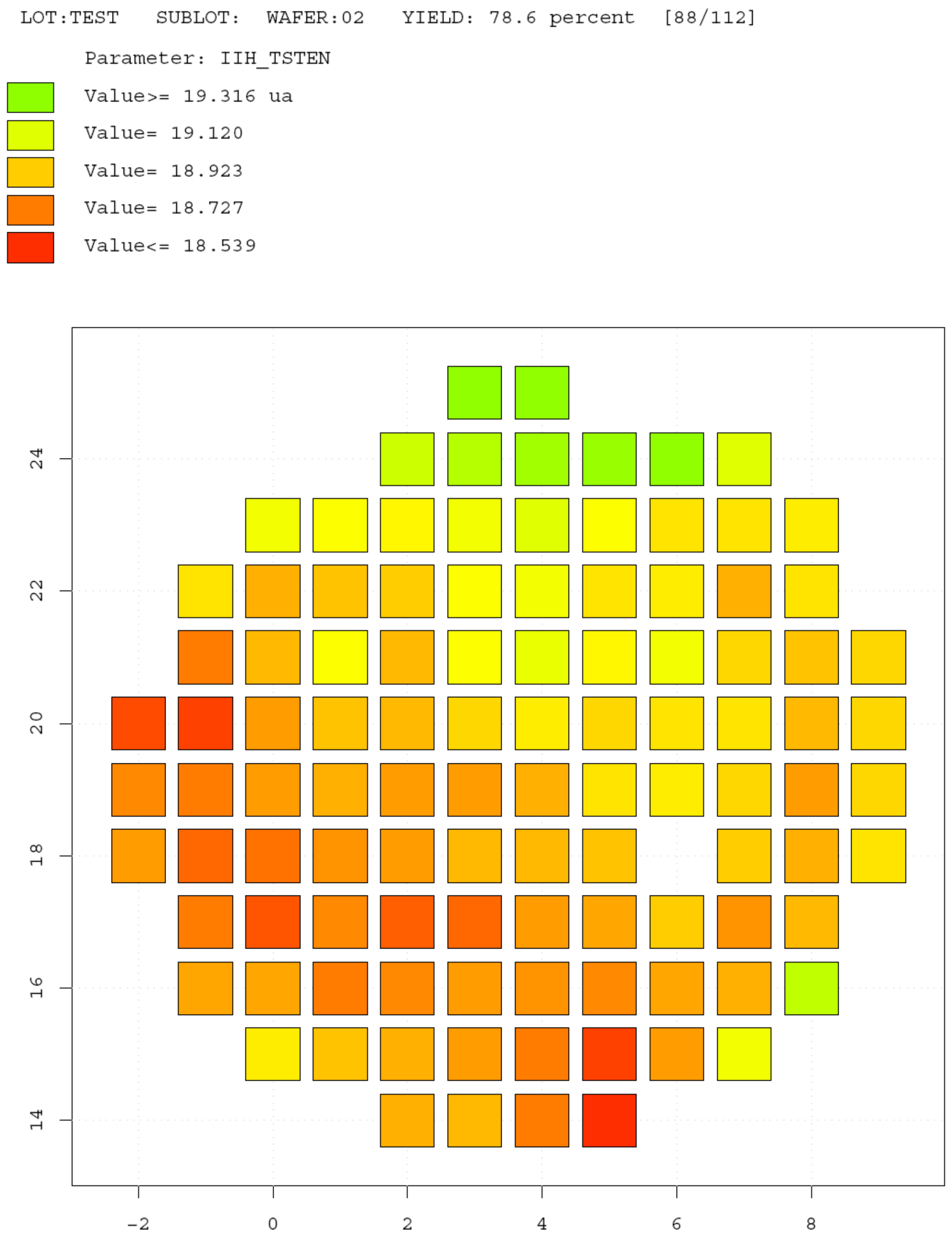
- auto_open_pdf – if this flag is set, as soon as the PDF file has been created, it will be opened using your default PDF reader.

3.4.6.1 Example Soft Binning Wafer Map

```
 LOT:TEST   SUBLOT:  WAFER:02   YIELD: 78.6 percent  [88/112]
🟩  sbin  1:   88 ( 78.6%) PASS
🟨  sbin  4:   12 ( 10.7%) DELAY
🟧  sbin 10:    5 (  4.5%) CONT
🟥  sbin  3:    3 (  2.7%) FUNC
🟦  sbin  8:    2 (  1.8%) VIN
🟪  sbin 11:    2 (  1.8%) POWER
```



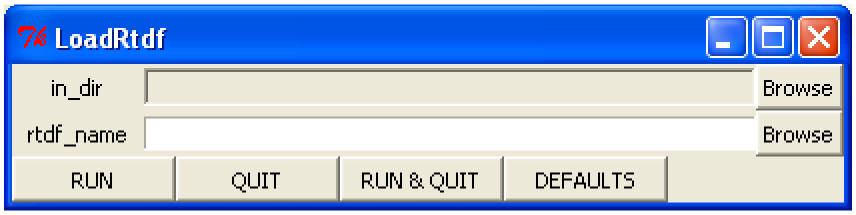
3.4.6.2 Example Parametric Wafer Map



3.5 Manual Edits

3.5.1 LoadRtdf

This script will load the specified RTDF file into the R workspace. Once an RTDF file has been loaded, you can type "ls()" in the R console window and you will see the objects are now loaded.

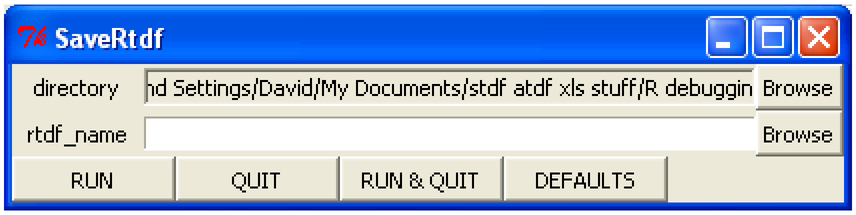


The various fields in detail:

- in_dir
- rtdf_name

3.5.2 SaveRtdf

This script will collect any RTDF specific objects in the R workspace and write them to the specified RTDF file.



directory: nd Settings/David/My Documents/stdf atdf xls stuff/R debuggin

The various fields in detail:

- directory
- rtdf_name

4 Tutorials

The following are some examples that show how some of the various scripts can be used.

4.1 Introductory Tutorial Using The Example STDF

As part of the RADAR package, an example STDF file has been included. (This STDF file was found on the internet at http://www.ge.infn.it/ATLAS/Electronics/home.html) This tutorial guides you through the process of importing the example STDF file into RADAR (aka into an .rtdf file), and then generating histograms, statistics, wafermaps, and test vs test plots from this data.

4.1.1 Getting the data into RADAR (aka rtdf format)

- Convert the STDF file into RTDF:

  In the RADAR window, Converting Menu -> ConvertStdf

  In the ConvertStdf window, click the stdf_name Browse button

  In the Open window, navigate to the RADAR package directory, select the a595.stdf.gz file, click the Open button.

  In the ConvertStdf window, the stdf_name and the rtdf_name fields should now be filled in, click the RUN button.

  In the R Console window, you should see that the ConvertStdf command has been sent, and about every 5 seconds there should be an update message on the conversion progress. On an Intel T2400 @ 1.83GHz, conversion takes a bit under 35 seconds. When the conversion is done, the R Console window should look something like:

  ```
  ConvertStdf(stdf_name = "a595.stdf.gz", rtdf_name = "a595.rtdf",
       auto_93k = TRUE, do_summary = TRUE, just_fail_tests_summary = TRUE,
       endian = "big", stdf_dir = "")
  Tester Type is: Catalyst
  processing PIR record 49 ...  15.2% through file
  processing PIR record 104 ...  31.1% through file
  processing PIR record 157 ...  47.0% through file
  processing PIR record 209 ...  62.8% through file
  processing PIR record 263 ...  78.6% through file
  processing PIR record 315 ...  94.8% through file
  Conversion Finished!
   processed 336 Devices x 123 Parameters in 32.14 seconds
  Writing summary file to a595.summary...
  FINISHED
  >
  ```

  In the ConvertStdf window, you can now click the QUIT button.

- Separating the 3 wafers into their own RTDF files

  In the RADAR window, Manipulating Menu -> SplitWafers

  In the SplitWafers window, the in_file should already be filled in with the most recent .rtdf file you were working with. Leave the out_file field blank so the script will automatically name the output files. All you have to do is hit the RUN & QUIT button.

You have now successfully created 4 different RTDF files:

a595.rtdf

a595_wafer02.rtdf

a595_wafer03.rtdf

a595_wafer09.rtdf

4.1.2 Generating histograms

Now that we have the 4 different RTDF files, let's generate some histograms.

- In the RADAR window, click the PlotRtdf button.
- In the PlotRtdf window, increment the "number of datasets" to 4 (ie. hit + button a few times)
- For dataset 2, hit the Browse button to bring up the Open window, and select the a595_wafer02.rtdf file. Similarly for the dataset 3 and 4, select wafers 3 and 9.
- If you are going to be using Microsoft Excel to look at the statistics table, then either uncheck the use_csv_formulas box or uncheck the use_OOCalc_csv box. Excel and OpenOffice Calc expect formulas to be in slightly different formats. (semi-colon vs. comma field separators)
- Hit the RUN button at the bottom of the PlotRtdf window.

  In the R Console window, you should see that the PlotRtdf command has been sent, and about every 5 seconds there should be an update message on the plotting progress. On an Intel T2400 @ 1.83GHz, plotting takes about 14 seconds. When the conversion is done, the R Console window should look something like:

```
PlotRtdf(rtdf_name = c("a595.rtdf", "a595_wafer02.rtdf", "a595_wafer03.rtdf",
     ...)
...now processing parameter 32 of 123 ...
...now processing parameter 85 of 123 ...
Finished! processed 123 parameters in 14.04 seconds
>
```

If you now look in your working folder, you should find 2 new files, my_histograms.pdf and my_histograms.csv. For each test, you will have 4 histograms, one for all the parts, and 3 others on a per wafer basis.

If you look at the histograms for test 210, IDD_Static, you will find that the standard deviation and the Cpk numbers vary a fair bit. There are 2 approaches you can take to get more useful statistics, you can go through and screen out individual outliers, or you can switch to robust statistics. The CSV file already contains both the normal and robust statistics. To have the histograms use robust statistics, you would need to select the checkbox next to "do_robust_stats" in the PlotRtdf window.

4.1.3 Generating histograms for a reduced test list and with differing limits

If you only want to generate histograms for a subset of the tests, or if you wanted to use different test limits or plot limits than what is the default ones for a given RTDF file, you would use the ConvertParameters script to generate an alternate ParametersFrame.

- In the RADAR window, click on the Converting Menu and select ConvertParameters.
- In the ConvertParameters window, make sure the rtdf->csv direction is selected, and set the in_file to the name of the RTDF file you want to work with (in our example, a595.rtdf), then hit the RUN button.

You should now have a parameters.csv file in you working folder. Open this file in you prefered spreadsheet. In our example, we'll delete rows 2 to 76, leaving the header row and the Delay Line tests. For "Delay Line #01" , change the hi_lim value from 420 to 430. For "Delay Line #07", add the value

495 to the plothl cell. Now save the file as "parameters2.csv".

- In the ConvertParameters window, now select the csv->rtdf direction. Either enter the in_file as parameters2.csv, or Browse and select this file, then click RUN & QUIT at the bottom of the window.
- Hopefully you still have the PlotRtdf window open from the previous section, now click on the Browse button for the param_name field, and browse to the parameters.rtdf file you just created. Change the pdf_name to a595_shortlist_histos.pdf and then hit the RUN button.

You should now have a new .pdf and .csv file in your working folder, with the test list, plot limits, and test limits reflecting what we put in the parameters.rtdf file.

4.1.4 Generating binning wafer maps

Since the example STDF file contains wafer data, we can generate wafer maps.

- In the RADAR window, click the WaferMap button
- In the WaferMap window, the rtdf_name may already be "a595.rtdf". If not, hit the Browse button to select this file. The wafermap type radio buttons should already have sbin selected. If not, click on the circle in front of sbin to select soft binning. Now, hit the RUN button.
- In the R Console window, you should see the WaferMap(...) call and with in a few seconds the "Finished!" line.

If you now look in your working folder, you should find a new file, wafer_map.pdf. This PDF file will have 3 pages, one for each wafer.

4.1.5 Generating parametric wafer maps

As well as soft bin or hard bin wafer maps, we can also generate parametric wafer maps.

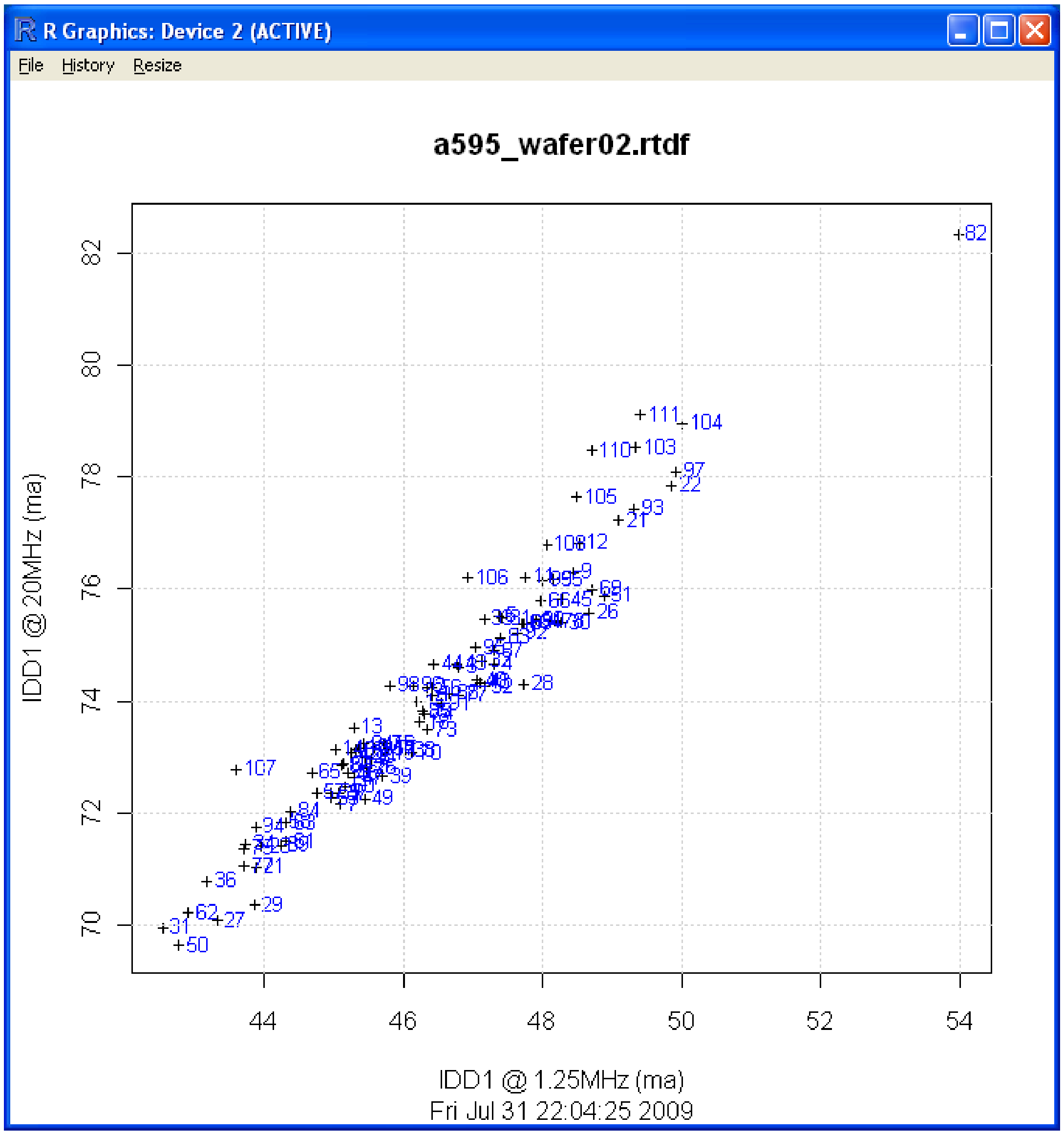
- In the RADAR window, click the WaferMap button
- In the WaferMap window, the rtdf_name may already be "a595.rtdf". If not, hit the Browse button to select this file. Click on the radio button in front of parameter to select parametric wafer mapping as the type of wafer map. To select a particular test for the mapping, click on the Browse button on the parameter field line.
- In the Parameter Browser window, you can now scroll to the test you want to plot, select that name, and hit the OK button. For this example, select IIH_TSTEN, it should be the 19th test in the list. This test is an input leakage test on a pin with a pull-down resistor.
- In the WaferMap window, let's change the pdf_name to "a595_IIH_TSTEN_wafermap.pdf", and then hit the RUN button at the bottom of the window. [To be really efficient (aka lazy), select the filename above and copy/paste it into the WaferMap window.]
- In the R Console window, you should see the WaferMap(...) call and with in a few seconds the "Finished!" line.

If you now look in your working folder, you should find a new file, a595_IIH_TSTEN_wafermap.pdf. This PDF file will have 3 pages, one for each wafer. You should see a similar gradient across all 3 wafers reflecting the sheet resistance.

4.1.6 Comparing 2 different tests on an X-Y plot

If you want to see how one test correlates to another, you may find the PlotTestvsTest script useful.

- In the RADAR window, click on the PlotTestvsTest button
- In the PlotTestvsTest window, hit the rtdf_name Browse button and browse to the a595_wafer02.rtdf file.
- In the PlotTestvsTest window, hit the x parameter Browse button and browse to the "IDD1 @ 1.25MHz" test. Similarly, hit the y parameter Browse button and browse to the "IDD1 @ 20MHz" test. Now hit the RUN button. A R graphics window should pop up looking like below:



- In the PlotTestvsTest window, unclick the "show_part_ids" check box and rehit RUN. The part ids should now disappear from the plot. If you want to save this plot, click on the File menu at the top of the Graphics window and select the Save as... option and choose the format you prefer.

4.2 RtdfTransform and Correlating Between Different Test Systems

This tutorial introduces you to the RtdfTransform function and how it can be used. Sometimes you are trying to correlate data between one ATE system and another, or against bench data. In many cases, the testnames for similar tests differ, or the measurement is made slightly differently on one system compared to the other. This script helps in transforming data from the different systems into the same format for direct comparison.

- Get the data from the 2 different sources into RTDF. Use one or more of ConvertStdf, or ConvertCsv, or other Conversion scripts depending on what format the original data is in.
- Create the transform command file. A good place to start is to extract the parameter information from the different datalogs. Run ConvertParameters on both RTDF files to generate "desired_params.csv" and "orig_params.csv". Start with the first 2 columns of the "desired_params.csv" as the basis for your "my_transform.csv" file. For tests that have a direct testname to testname match, you are done. For tests that are the same except for testnames, then add a 3rd column,"orig_tname", and put the appropriate "orig_params.csv" testname in that column for these tests. A more elaborate example would be if the original data was for a leakage test on a pin with a pull-up or pull-down resistor, and you wanted the data to be the resistance. In this case, you would also fill in the next 9 columns...

  "prefix","units","limits_flag","new_ll","new_ul","multiplier","power","log","shift"

  with

  "K","ohm",1,33,94.3,3.3,-1,,

  In this example, the test is an IIH test on a 3.3V part, and the pulldown resistor is in the 33Kohm to 94.3Kohm range.

  new_value = 3.3V * orig_value^(-1)   [aka R = V/I]
- Transform the one dataset into the same format as the other: RtdfTransform, setting the transform_csv field to the file you created in the previous step.

4.3 ReplaceTestnames and Dealing with J9xx data

This tutorial introduces you to the ReplaceTestnames function and how it can be used. The Teradyne J971 and similar testers can datalog in an ascii format and in stdf format. To get this data into RADAR, one would use ConvertStdf for the STDF file, or ConvertJ9 for the ascii file. A shortcoming of the J9's STDF generation is that the pin name information is missing from the testnames. A shortcoming of the ascii files is that the precision of the measured data is limited to the precision of the ascii print format, not the actual measured value. Using the ReplaceTestnames function, we can merge the ascii file testname information with the stdf file measurements and have an RTDF file that combines the best of both datalog approaches. NOTE: this script assumes the test numbers are unique and uses those to cross-reference the testnames. The steps you would go through would be:

- Generate an RTDF file that has the correct test names: Use ConvertJ9 on an ascii datalog file from the same test program as the STDF file(s) you wish to convert. You only need to do this once.
- Convert the STDF file(s) into RTDF format: Use ConvertStdf

- For each RTDF file created with ConvertStdf, update its test names: Use ReplaceTestnames, setting the reference file to the RTDF file generated by ConvertJ9 in the first step.

4.4 1000 Hour Life Test

As part of 1000 hour life testing, you should eventually end up with something like a collection of stdf datalogs at 0 hours, 24 hours, 250 hours, 500 hours, and 1000 hours for 77 or so serialized devices. The steps you would go through would be:

- Get each data file into RTDF: ConvertStdf if STDF files, otherwise one of the other Conversion scripts if the data is in a different format.
- Clean up data files. Usually, by the 500 hour and 1000 hour test session, contact issues start occurring. Usually, if there are failures, the leads of the part are cleaned and the part is retested until a pass occurs. The ShrinkRetests script will search through the RTDF file and remove devices where a later occurrence of the same device number is found in the file. This would sanitize this data.
- Compare data at the different time points: Use PlotRtdf GUI, set number of datasets to 5 for this example, then set the input files 1: 0 hours, 2: 24 hours, ... 5: 1000 hours. check the checkbox for do_hist_and_xy. By looking at both the histograms and the XY plots, you can quickly see overall distribution shifts, or single device shifts that are still within the total population.

  .

4.5 Looking inside an RTDF file

If you get to the point where you want to write your own scripts to do specific things, you will probably want to know a bit about what the RTDF file looks like and how you interact with R frames and matrices. Let's do a walk through with the RTDF file created in the first tutorial.

- In RADAR window, click on LoadRtdf button
- In the LoadRtdf window, the rtdf_name may already be set to "a595.rtdf". If not, enter it or use the Browse button. Now hit the RUN & QUIT button.
- In the R console window, enter the command: ls() This will list all the objects in the workspace, which will be the objects loaded from the RTDF file.

```
> ls()
 [1] "DevicesFrame"    "HbinInfoFrame"   "LotInfoFrame"     "ParametersFrame"
 [5] "ResultsMatrix"   "SbinInfoFrame"   "TSRFrame"         "WaferInfoFrame"
 [9] "WafersFrame"
>
```

- In the R console window, enter the commands: dim(DevicesFrame), dim(ParametersFrame) and dim(ResultsMatrix)

```
> dim(DevicesFrame)
[1] 336   9
> dim(ParametersFrame)
[1] 123   8
> dim(ResultsMatrix)
[1] 336 123
```

  This gives the dimensions of the 3 objects. There should be a direct correspondence between the size of the ResultsMatrix and the first dimensions of the Devices and Parameters Frames.

- In the R console window, enter the commands DevicesFrame[1,] and ParametersFrame[4,]

  > DevicesFrame[1,]

```
        part_id temp x_coord y_coord wafer_index soft_bin hard_bin testtime site
my_list       1   NA       2      14           1        1        1        0    0
> ParametersFrame[4,]
   testnum   testname scaler units    ll    ul plot_ll plot_ul
4      210 IDD_Static      3     a 0.035 0.055     NaN     NaN
>
```

  Note that empty fields can be populated with NA or NaN.

- Now, what are the indices in the DevicesFrame for parts that were binned into soft bin 10?

```
> which(DevicesFrame["soft_bin"]==10)
[1]  80 100 101 109 112 212 213 221 224 304 324 325 333 336
```

- Now, find the ParameterFrame index for the test "VOH_DTOP"

```
> match("VOH_DTOP",ParametersFrame[["testname"]],nomatch=0)
[1] 62
> grep("VOH",ParametersFrame[["testname"]])
[1] 51 52 53 54 55 56 57 58 59 60 62 63 66
```

4.6 Finding Histogram Outliers

From the first tutorial, we generated histograms for the various tests. If we now want to find out which devices were the outliers for a specific test, or if we wanted to remove that measurement or that entire device from the dataset, we would use the FilterByResult script. From the histograms for test number 210, IDD_Static, we see that there are 2 outliers.

- In the RADAR window, click Manipulating Menu -> FilterByResult
- In the FilterByResult window, the rtdf_file should already be set to a595.rtdf, but if not then either enter this file name or hit the Browse button and navigate to this file. Next, select the testname, either by typing in IDD_Static, or hitting the Browse button and selecting this name from the list. Then change the action from remove to report. Since we are just reporting, the filter doesn't matter if it is result or device. From the histograms, we know the 2 outliers are both above the upper limit of 55mA. The type is already ">", so we don't need to change that. For the scaler, click on the Menu button and select "m". For the value, click on the Edit button, enter the value of 55 in the Numeric Entry window and hit the OK button. Finally, hit the RUN button at the bottom of the FilterByResult window.
- In the R Console, you should see the following output:

```
FilterByResult(in_file = "a595.rtdf", action = "report", filter = "result",
     ...)
Index: 102  Part_id: 102    Value: 58.184989 m
Index: 148  Part_id: 148    Value: 127.806455 m
```

```
Finished!
>
```
If we had set the action to remove, and the filter to device, then these 2 devices would be removed from the RTDF file written to the out_file name, which by default is filtered.rtdf.

4.7 PlotRtdf XY plots from wafer data

If you want to compare a wafer probing session against a reprobe session and the part_id numbers don't match, then running PlotRtdf won't give you the results you want. To achieve this, overwrite the part_id's of the 2 different files with ones derived from the X and Y coordinates.

- LoadRtdf() the first file
- In the R console window, run these 3 commands:

  ```
  xs = as.numeric(DevicesFrame[["x_coord"]])
  ys = as.numeric(DevicesFrame[["y_coord"]])
  ```
  And if your X and Y coordinates are always positive and <1000...

  ```
  DevicesFrame[["part_id"]] = 1000*xs + ys
  ```
  Otherwise you will need to do something a bit more complicated...

  ```
  min_x = min(xs)
  min_y = min(ys)
  ... etc
  ```

- SaveRtdf() to a new name
- Repeat the above steps on the 2nd file, and now PlotRtdf XY plots should work.

5 Useful Snippets of R code

Understanding loop behavior (ie. this isn't "c"):

```
> for (i in 1:5) {
+ if (i==3) i=6
+ cat(sprintf("i is %d \n",i))
+ }
i is 1
i is 2
i is 6
i is 4
i is 5
>
```

5.1 For dealing with rtdf data

View the device numbers...

```
as.character(DevicesFrame[["part_id"]])
```

list the rtdf files in the current directory...

```
system("ls *rtdf",intern=TRUE)
or
dir(pattern="rtdf$")
```

find index for a testname...

```
match("MY_TESTNAME",ParametersFrame[["testname"]],nomatch=0)
```

convert unix time to ascii time...

```
start_t = ISOdatetime(1970,1,1,0,0,0) + as.numeric(LotInfoFrame[["start_t"]])
```

6 Useful Snippets for CSV files in OpenOffice/LibreOffice

The longer term goal is to generate files directly into .odf with conditional formatting, but, in the short term, the following conditional formatting steps can be done after importing the .csv file created by ConvertCsv(). With conditional formatting, failing test values will have a red background. As you adjust limits, the coloring will update.

For LibreOffice 3.5.4... for CSV file without ll_eq_flag and ul_eq_flag's present...

When importing the .csv file, make sure to check the "Detect special numbers" box. This will avoid 1.3e-4 being interpreted as a string instead of a number, as an example.

Click on the J17 cell and select "Format → Conditional Formatting..."

For condition 1, select "Formula is" and enter

```
ISBLANK(J17)
```

as the formula.

For condition 2, select "Formula is" and enter

```
AND(ISNUMBER($F17),($F17>J17))
```

as the formula. Click on "new style" and in the popup window enter "failing" as the name, click on the background tab and select the red color, then the OK button.

For condition 3, select "Formula is" and enter

```
AND(ISNUMBER($G17),($G17<J17))
```

as the formula. Change the style to "failing" and hit the OK button.

Now copy this cell (J17), select all the data cells (J12-> ZZ99 or whatever your sheet size) and paste special, just pasting the format.

All the failing measurements should now have a red background. If you relax a limit, the colors will adjust accordingly.