

HRC Technical Memorandum

On-Orbit HVPS Performance Statistics and Trends

Project	HRC
File	hrc-tm-hvps.1.tex
Generation Date	August 9, 2003
RCS ID	<i>Revision : 1.1.1.1</i>
Prepared By	J. Chappell
Test Reference	
Test Date	
Test Conductor	

Contents

1	Introduction	1
1.1	Objective	1
1.2	Scope	1
1.3	Documentation Organization	1
2	RELATED DOCUMENTATION	2
2.1	Parent Documents	2
2.2	Applicable Documents	2
2.3	Information Documents	2
3	Data Processing	3
4	Processing Results	5
4.1	HVPS State Transitions vs. Time	5
4.2	HVPS Cumulative Seconds at Nominal Operating Voltage	9
4.3	HVPS Commanded Step and Monitor Voltages vs. Time	13
4.3.1	I HVPS Commanded Step Levels vs. time	13
4.3.2	S HVPS Commanded Step Levels vs. Time	17
4.3.3	Shld HVPS Commanded Step Levels vs. Time	21
A	Terms, Abbreviations and Definitions	25
A.1	Acronyms	25
B	Processing Scripts and Code	26
B.1	doit.master	26
B.2	extract_data.ksh	29
B.3	global_post_process_data.ksh	31
B.4	gen-cum_hvps_sc.pl	32
B.5	gen-cum_hvps_sec_at_step.pl	39

B.6	csdtrdb2gp	48
-----	----------------------	----

List of Figures

1	Launch-to-Date HVPS Cumulative State Change.	5
2	1999 HVPS Cumulative State Change.	6
3	2000 HVPS Cumulative State Change.	6
4	2001 HVPS Cumulative State Change.	7
5	2002 HVPS Cumulative State Change.	7
6	2003 HVPS Cumulative State Change.	8
7	HVPS L2D Cumulative On Time at Nominal Operating Voltage.	9
8	HVPS 1999 Cumulative On Time at Nominal Operating Voltage.	10
9	HVPS 2000 Cumulative On Time at Nominal Operating Voltage.	10
10	HVPS 2001 Cumulative On Time at Nominal Operating Voltage.	11
11	HVPS 2002 Cumulative On Time at Nominal Operating Voltage.	11
12	HVPS 2003 Cumulative On Time at Nominal Operating Voltage.	12
13	2000 I HVPS Commanded Step and Monitor Voltage.	14
14	2000 I HVPS Commanded Step and Monitor Voltage.	14
15	2001 I HVPS Commanded Step and Monitor Voltage.	15
16	2002 I HVPS Commanded Step and Monitor Voltage.	15
17	2003 I HVPS Commanded Step and Monitor Voltage.	16
18	2000 S HVPS Commanded Step and Monitor Voltage.	18
19	2000 S HVPS Commanded Step and Monitor Voltage.	18
20	2001 S HVPS Commanded Step and Monitor Voltage.	19
21	2002 S HVPS Commanded Step and Monitor Voltage.	19
22	2003 S HVPS Commanded Step and Monitor Voltage.	20
23	2000 Shld HVPS Commanded Step and Monitor Voltage.	22
24	2000 Shld HVPS Commanded Step and Monitor Voltage.	22

25	2001 Shld HVPS Commanded Step and Monitor Voltage.	23
26	2002 Shld HVPS Commanded Step and Monitor Voltage.	23
27	2003 Shld HVPS Commanded Step and Monitor Voltage.	24

1 Introduction

1.1 Objective

This report and associated data products are part of an on-going HRC trends and performance monitoring program. Various HRC subsystems are monitored for use and operational trends. These data sets provide the project with the current status of the HRC detector system.

1.2 Scope

This investigation covers selected High Voltage Power Supply (HVPS) launch-to-date operational statistics as a function of time. The HRC has three detector high power supplies, the imaging, spectroscopic, and shield. This analysis reports on the following metrics for each power supply:

1. The number of HVPS state changes.
A HVPS state change is defined by a "on" to "off" state or from a "off" to "on" state.
2. HV as a function of time
Data sets and plots of the commanded HV step and the corresponding HV monitor.
3. Cumulative HV on times at the operational voltage.
Data sets and plots showing the cumulative number of seconds each HVPS has been on and operating at the nominal

1.3 Documentation Organization

The document contains the following sections:

- Data Processing Discussion
- Data Presentations:
 - HVPS State Changes vs. time
 - HVPS Cumulative Seconds at Operating Voltage vs. time
 - HVPS Commanded Step Level vs. time

2 RELATED DOCUMENTATION

2.1 Parent Documents

None.

2.2 Applicable Documents

Document ID	DESCRIPTION
<i>Reference List - Last Page</i>	

2.3 Information Documents

None

3 Data Processing

The following outline details the HVPS processing steps.

1. `extract_data.ksh`

This module extracts selected HRC HVPS telemetry data from the full telemetry files. The MSID's are tabulated below:

HRC EGSE MSID	MSID
S_ImHvps	2IMONST
S_ImHvpsCurLim	2IMCLST
S_SpHvps	2SPONST
S_SpHvpsCurLim	2SPCLST
S_ShldA	2S1ONST
S_ShldB	2S2ONST
V_ImTopHVStep	2IMTPAST
V_ImBotHVStep	2IMBPAST
V_ImBotHV	2IMHVLV
V_ImBotHVStep	2IMBPAST
V_ImTopHV	2IMHBLV
V_ImTopHVStep	2IMTPAST
V_SpTopHVStep	2SPTPAST
V_SpBotHVStep	2SPBPAST
V_SpBotHV	2SPHVLV
V_SpBotHVStep	2SPBPAST
V_SpTopHV	2SPHBLV
V_SpTopHVStep	2SPTPAST
V_ShldAHV	2S1HVLV
V_ShldAHVStep	2S1HVST
V_ShldAHVStep	2S1HVST
V_ShldBHV	2S2HVLV
V_ShldBHVStep	2S2HVST
V_ShldBHVStep	2S2HVST

These MSIDs are reformatted into a single ASCII RDB formatted file for each year of operation. These have a file name format of YYYY_hvps.rdb

2. `global_post_process_data.ksh`

The '`global_post_process_data.ksh`' ksh script performs global corrections to the raw RDB files. These global corrections include the addition of a common time stamp string, filtering

out garbage telemetry that occur as a result of format changes or bad SSR data. MSID values are reported ≈ 2 seconds in format 1 and ≈ 32 seconds in format 2. These have a file name format of YYYY_hvps.csdt.f1.rdb.

3. gen-cum_hvps_sc.pl

The 'gen-cum_hvps_sc.pl' perl script filters the corrected and time stamped RDB files into a cumulative state change RDB file. For each reported MSID event in the input file, the cumulative number of state (on/off) changes are reported for each HVPS. These have a file name format of YYYY_cum_hvps_sc.rdb. This script generates the graphical products in postscript(.ps) and portable network graphics (png) formats for each year and launch-to-data.

4. gen-cum_hvps_sec_at_step.pl

The 'gen-cum_hvps_sec_at_step.pl' perl script filters the corrected and time stamped RDB files into a cumulative 'number of seconds at the nominal number of seconds at the operating voltage' RDB file. For each reported MSID event in the input file, the cumulative number of on time seconds are reported for each HVPS. These have a file name format of YYYY_cum_hvps_sec.rdb. This script generates the graphical products in postscript(.ps) and portable network graphics (png) formats for each year and launch-to-data.

5. doit.master

The 'doit.master' ksh script ties all of the pipeline processing into a single script. Currently, the execution time for all processes is ≈ 1.5 hours/year. A typical year has $\approx 2 \times 10^6$ reported entries (rows).

4 Processing Results

4.1 HVPS State Transitions vs. Time

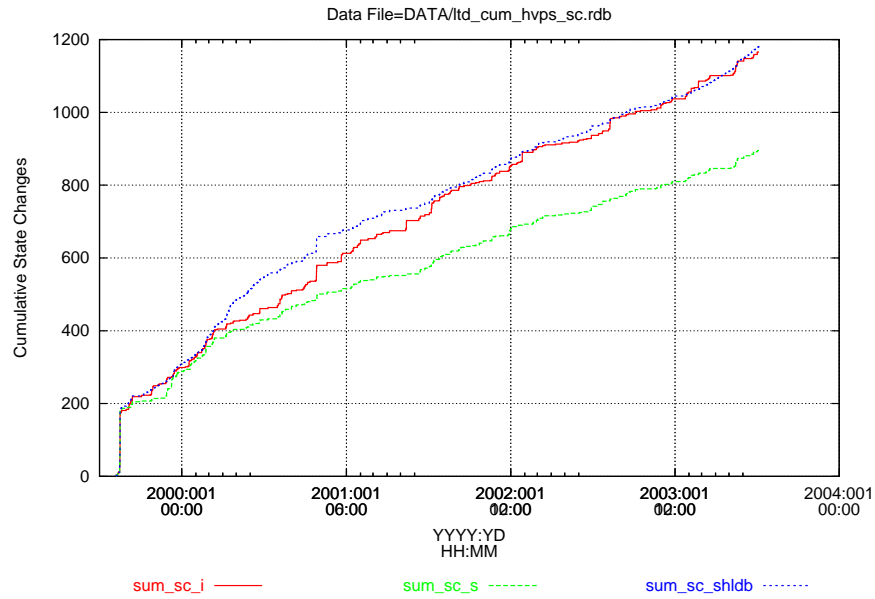


Figure 1: Launch-to-Date HVPS Cumulative State Change.

p

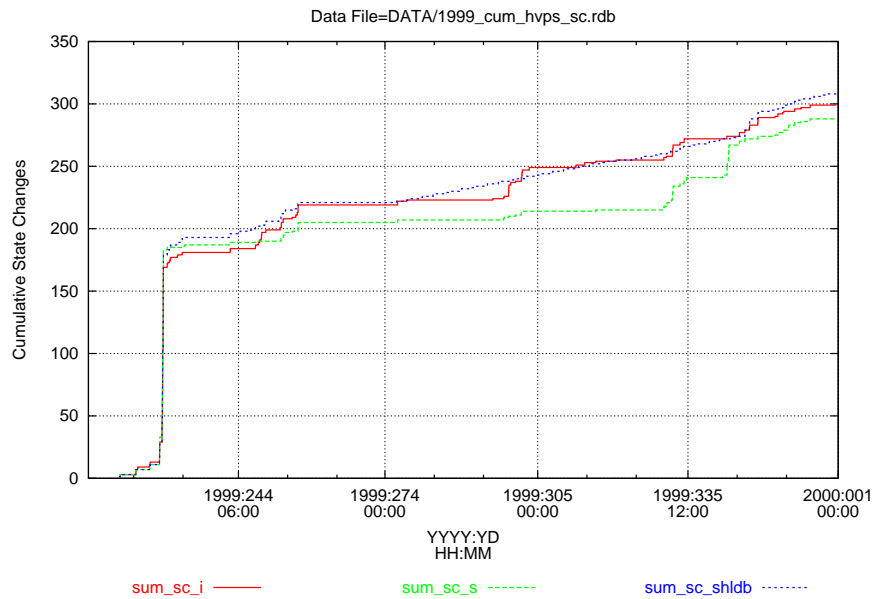


Figure 2: 1999 HVPS Cumulative State Change.

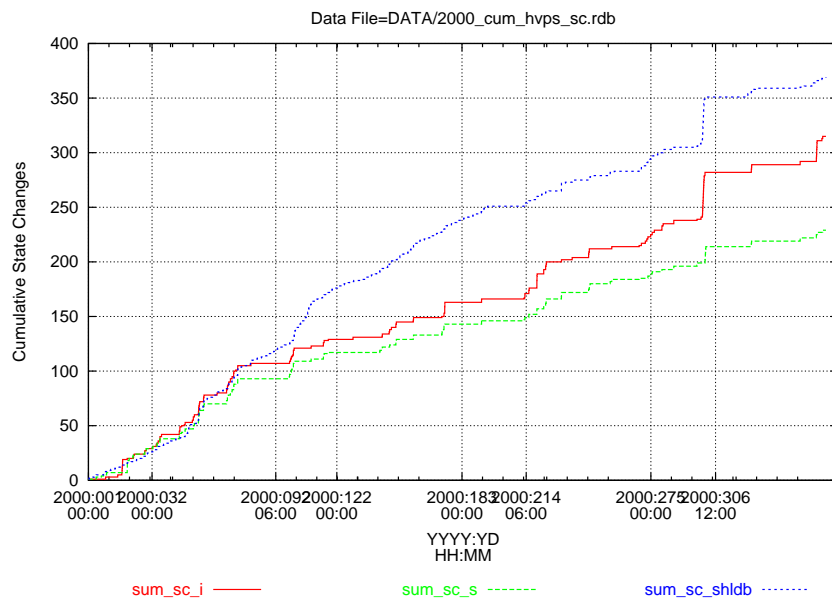


Figure 3: 2000 HVPS Cumulative State Change.

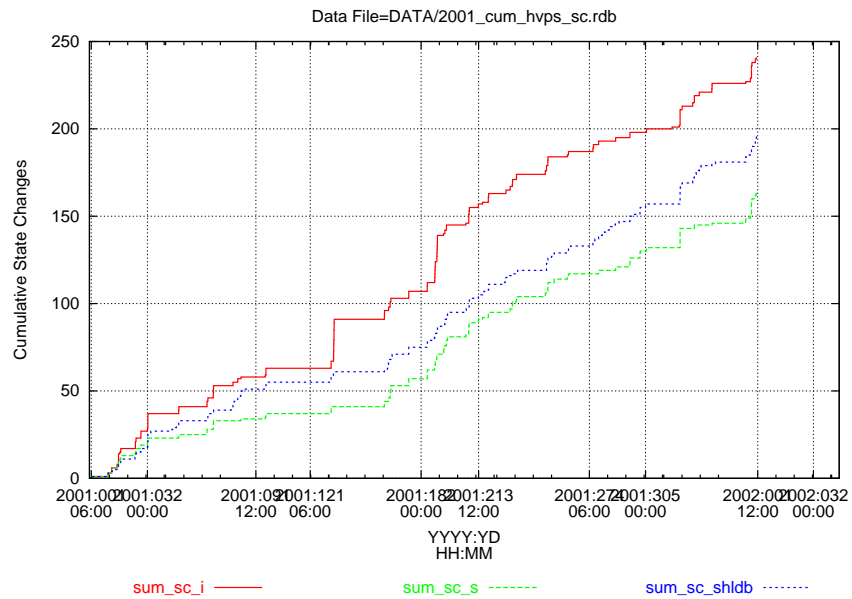


Figure 4: 2001 HVPS Cumulative State Change.

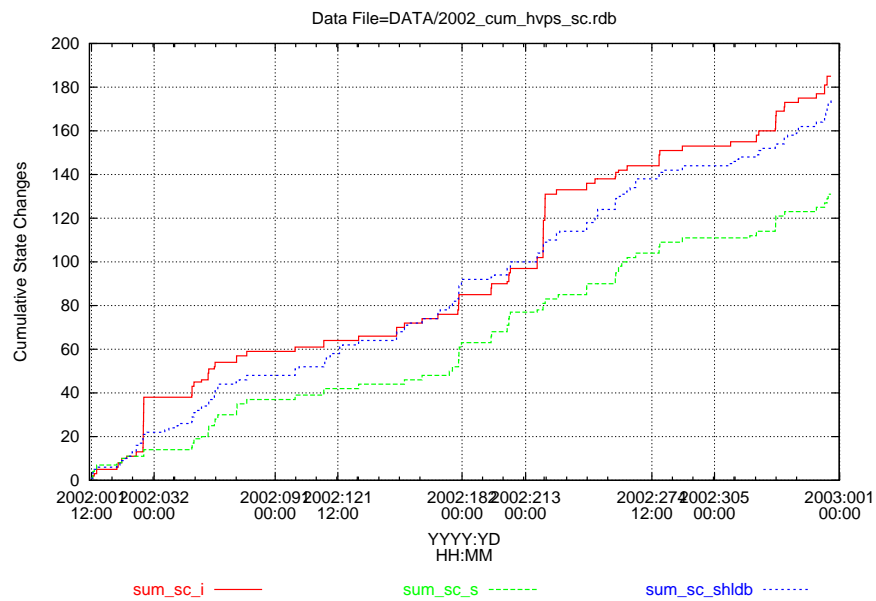


Figure 5: 2002 HVPS Cumulative State Change.

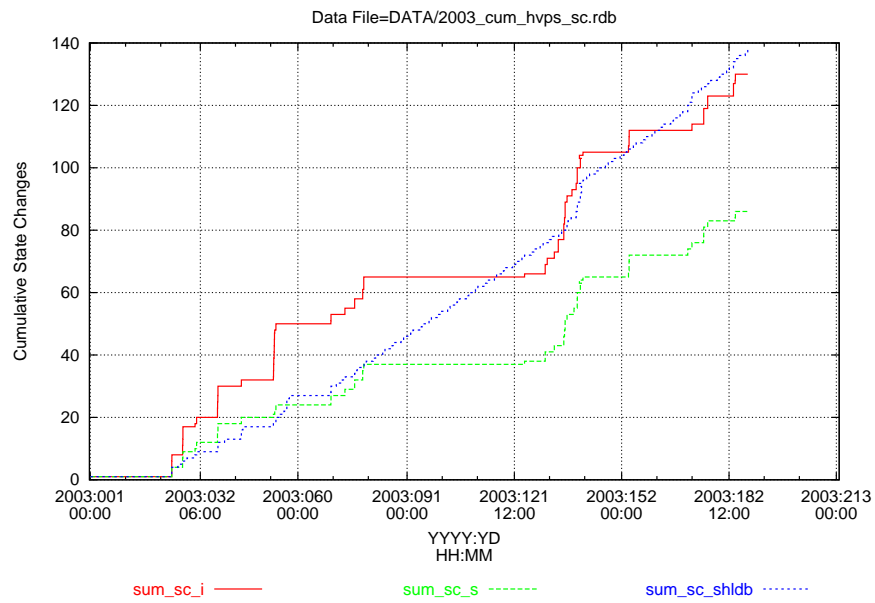


Figure 6: 2003 HVPS Cumulative State Change.

4.2 HVPS Cumulative Seconds at Nominal Operating Voltage

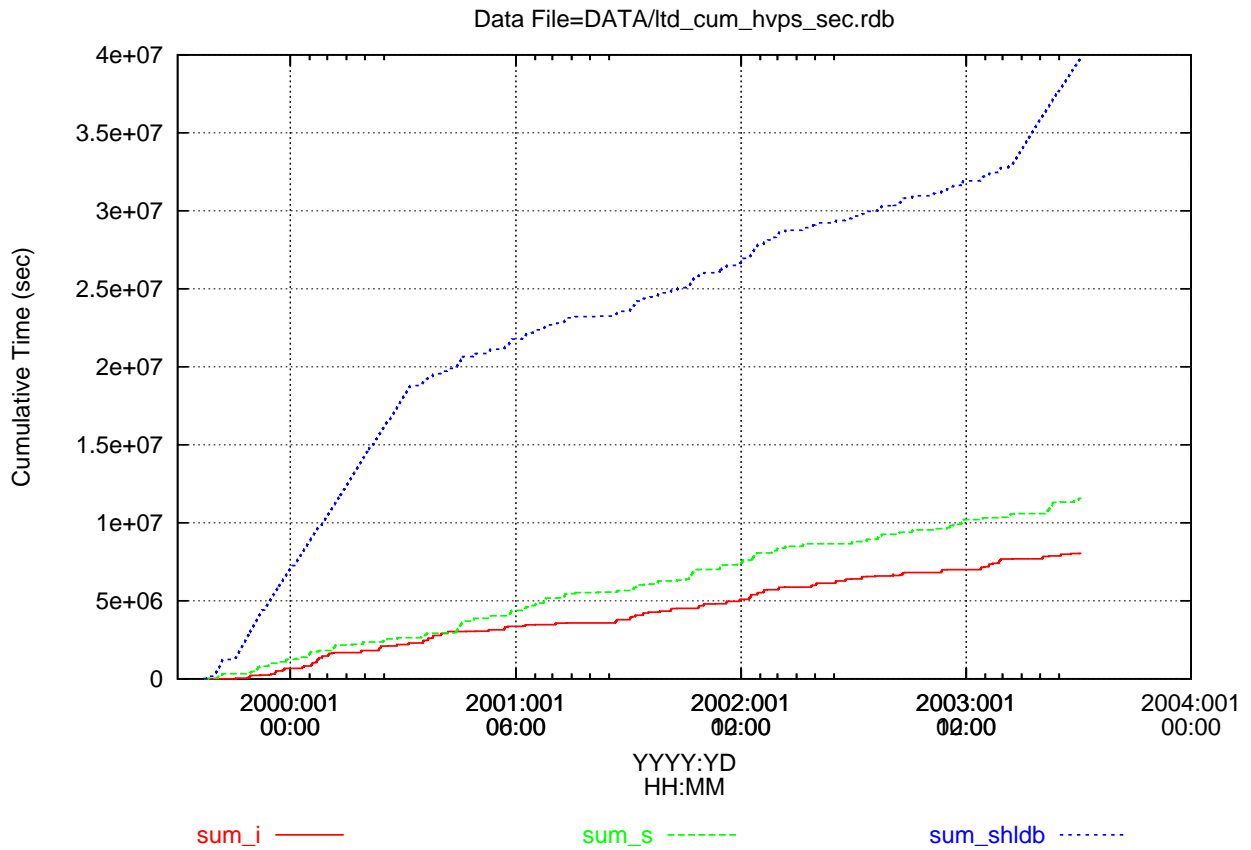


Figure 7: HVPS L2D Cumulative On Time at Nominal Operating Voltage.

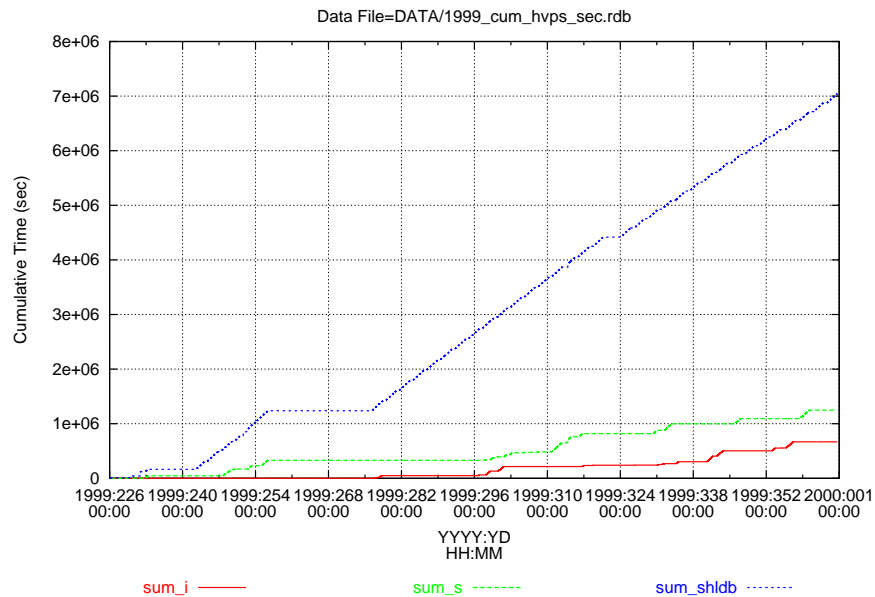


Figure 8: HVPS 1999 Cumulative On Time at Nominal Operating Voltage.

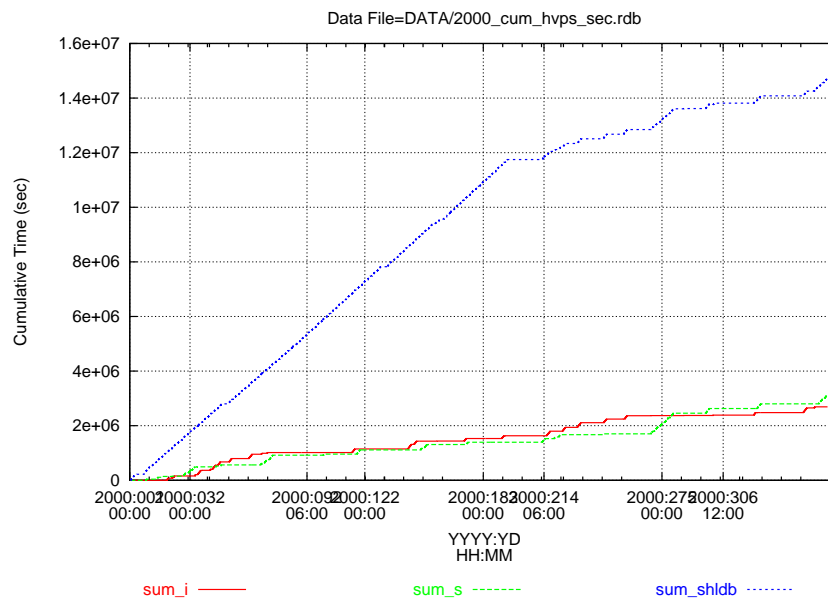


Figure 9: HVPS 2000 Cumulative On Time at Nominal Operating Voltage.

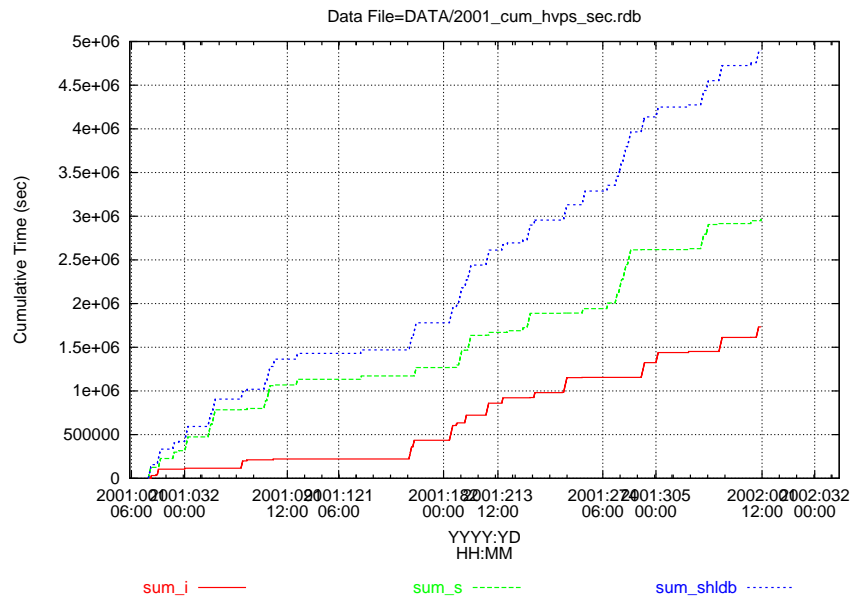


Figure 10: HVPS 2001 Cumulative On Time at Nominal Operating Voltage.

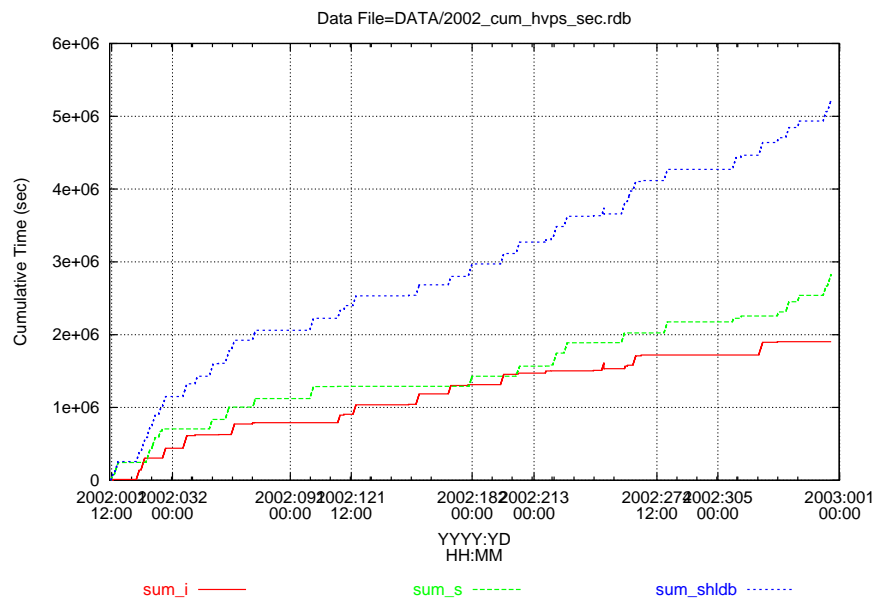


Figure 11: HVPS 2002 Cumulative On Time at Nominal Operating Voltage.

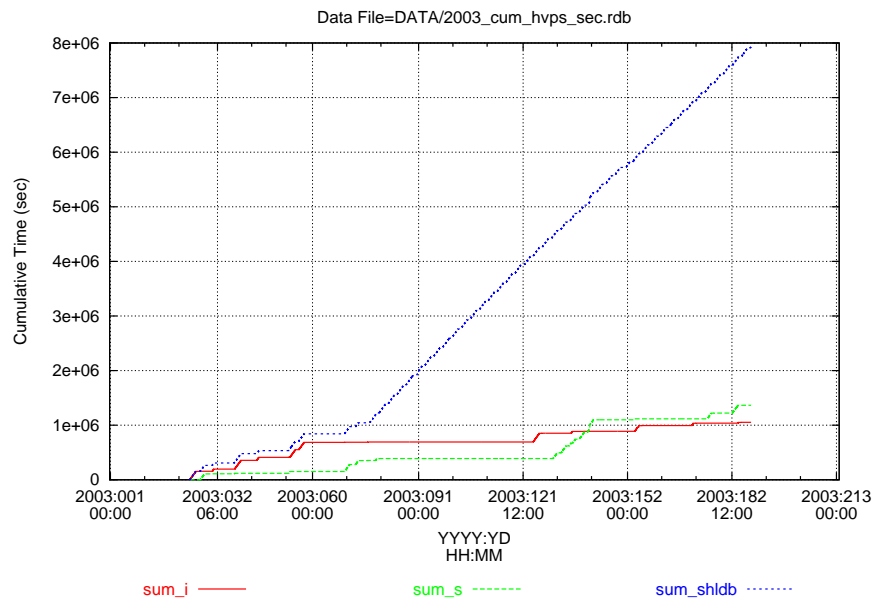


Figure 12: HVPS 2003 Cumulative On Time at Nominal Operating Voltage.

4.3 HVPS Commanded Step and Monitor Voltages vs. Time

4.3.1 I HVPS Commanded Step Levels vs. time

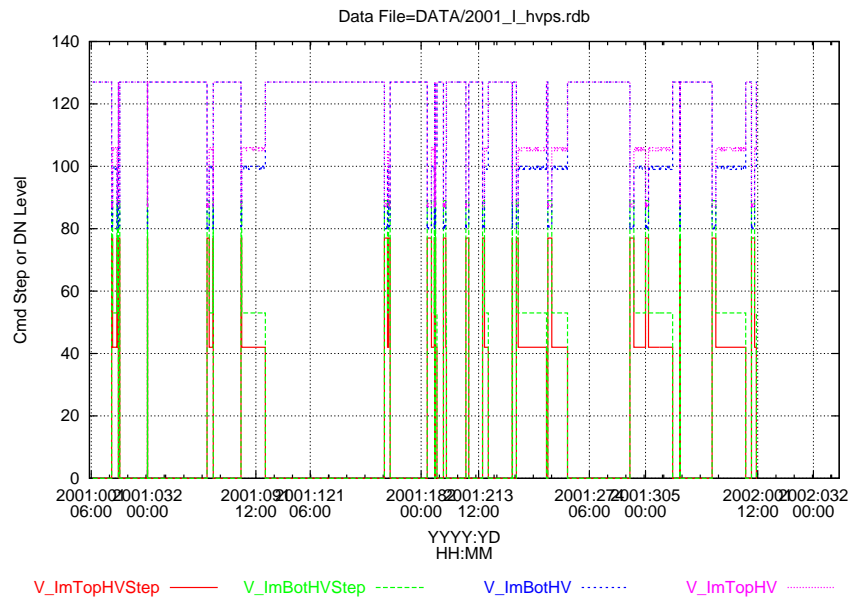


Figure 15: 2001 I HVPS Commanded Step and Monitor Voltage.

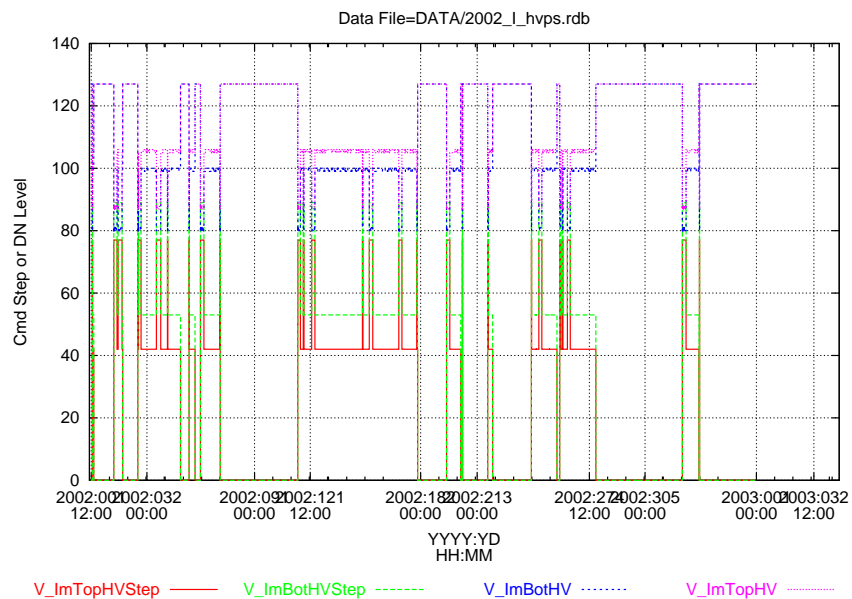


Figure 16: 2002 I HVPS Commanded Step and Monitor Voltage.

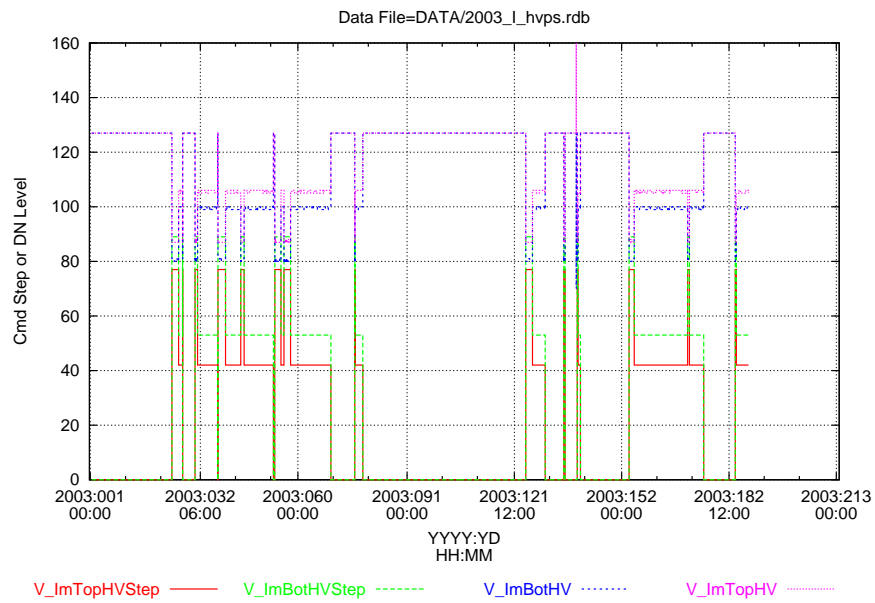


Figure 17: 2003 I HVPS Commanded Step and Monitor Voltage.

4.3.2 S HVPS Commanded Step Levels vs. Time

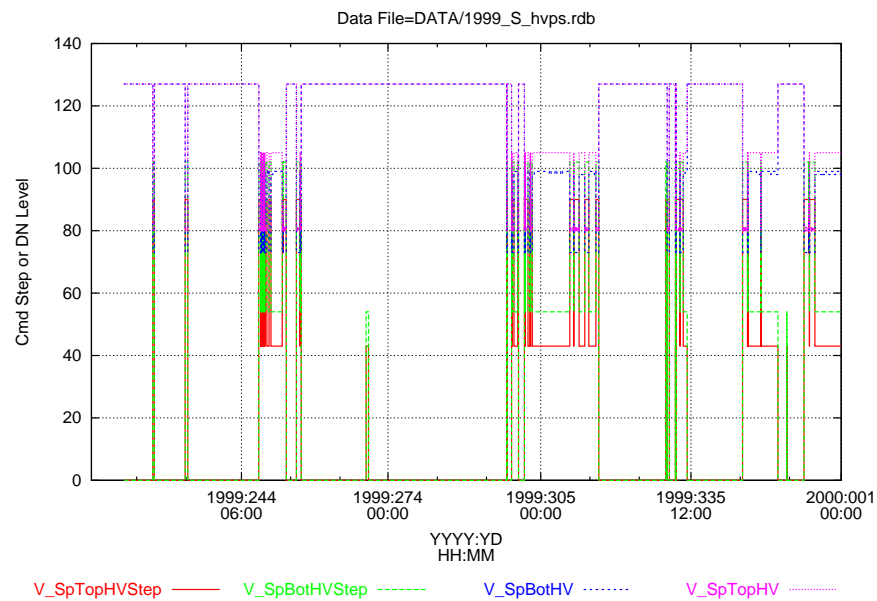


Figure 18: 2000 S HVPS Commanded Step and Monitor Voltage.

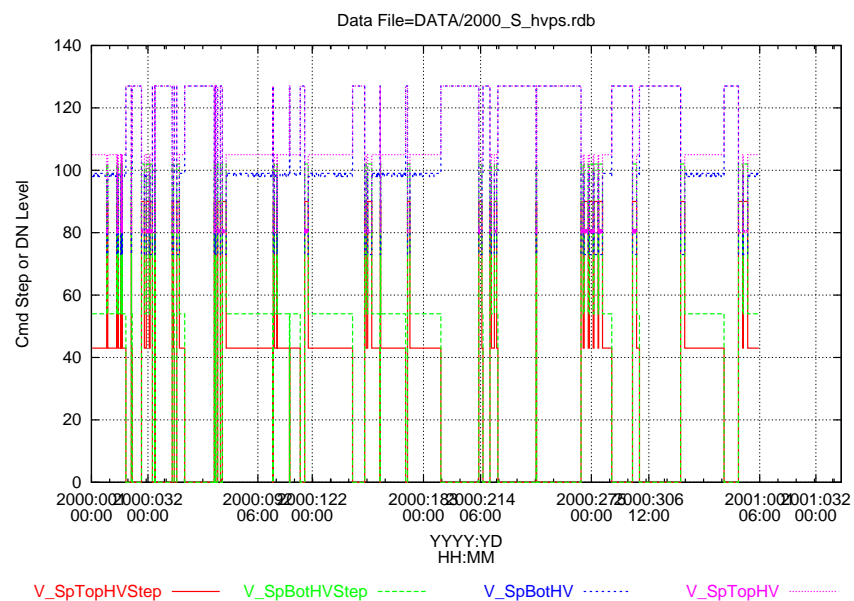


Figure 19: 2000 S HVPS Commanded Step and Monitor Voltage.

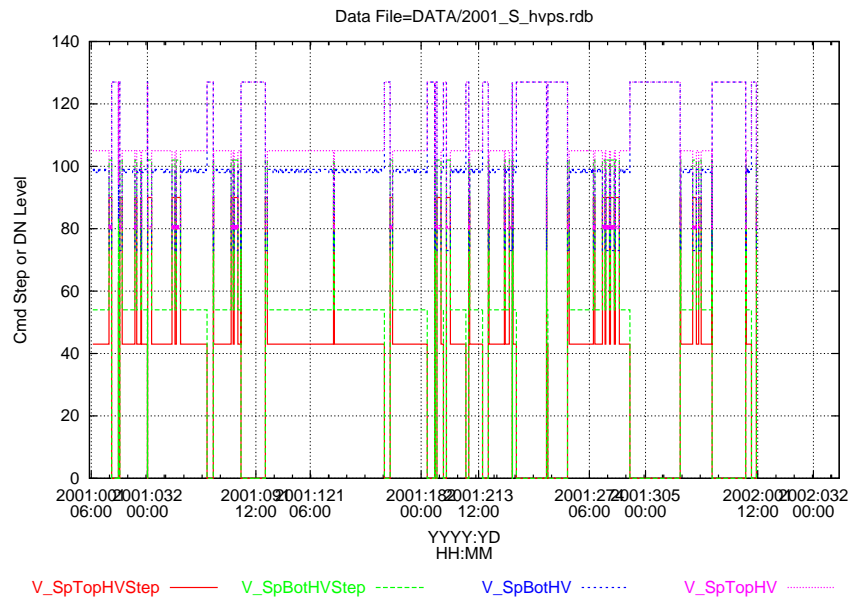


Figure 20: 2001 S HVPS Commanded Step and Monitor Voltage.

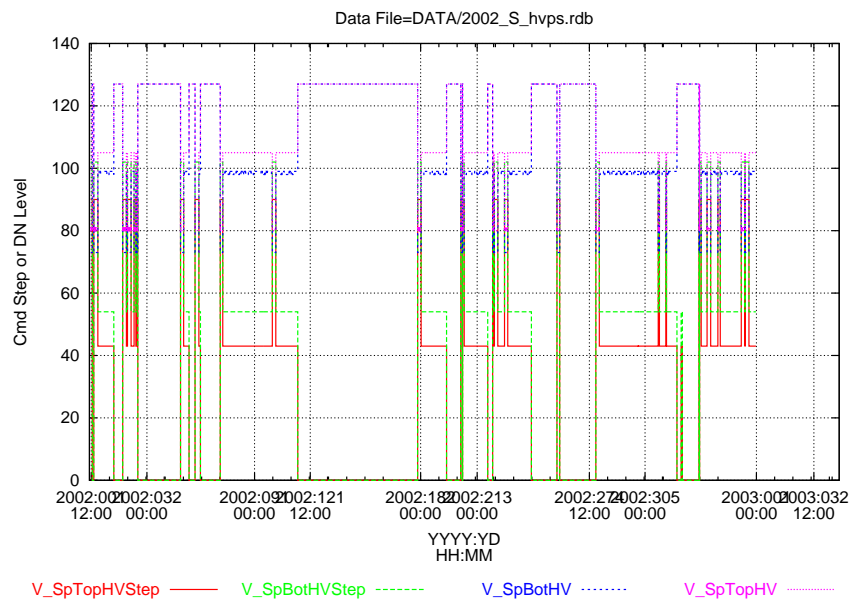


Figure 21: 2002 S HVPS Commanded Step and Monitor Voltage.

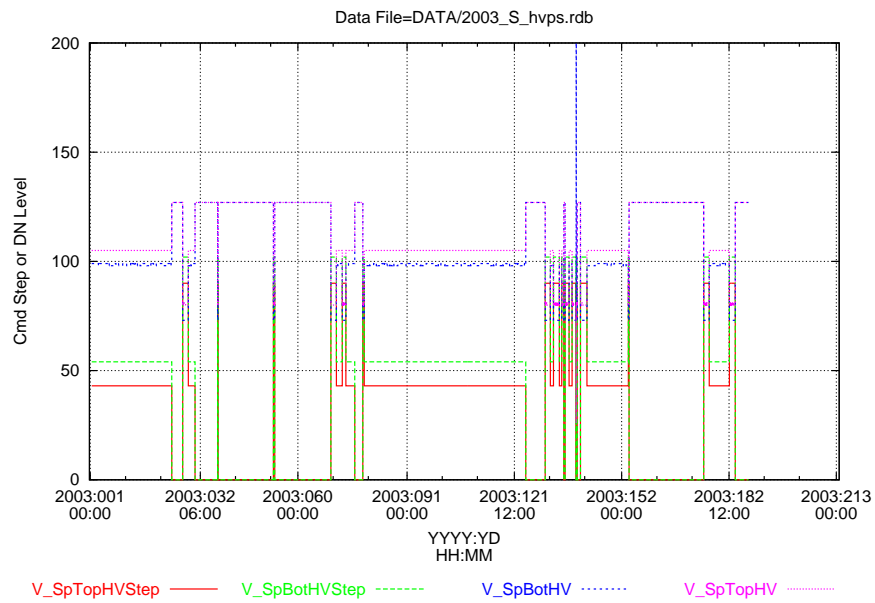


Figure 22: 2003 S HVPS Commanded Step and Monitor Voltage.

4.3.3 Shld HVPS Commanded Step Levels vs. Time

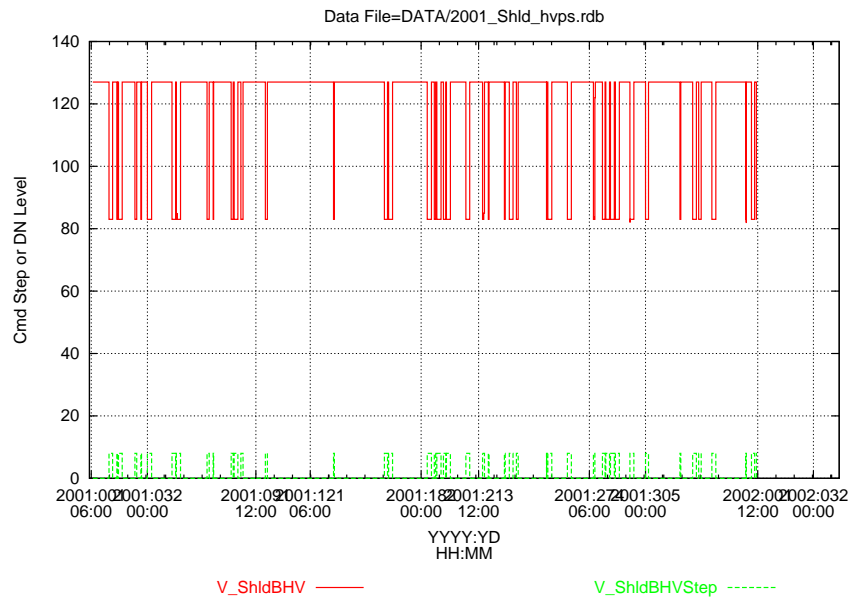


Figure 25: 2001 Shld HVPS Commanded Step and Monitor Voltage.

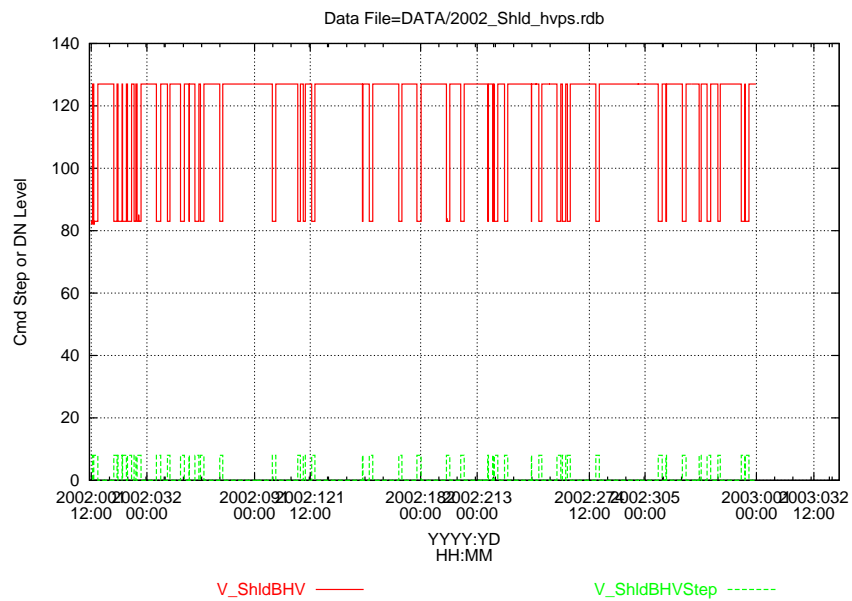


Figure 26: 2002 Shld HVPS Commanded Step and Monitor Voltage.

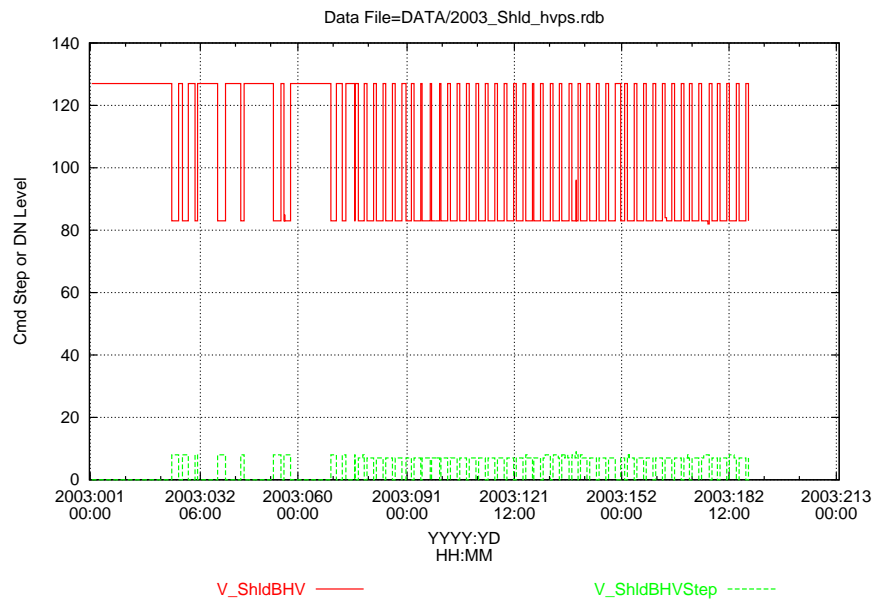


Figure 27: 2003 Shld HVPS Commanded Step and Monitor Voltage.

A Terms, Abbreviations and Definitions

A.1 Acronyms

This section follows the sections containing the content for the rolled-out section.

The abbreviations and acronyms section contains an alphabetized list of the definitions for abbreviations and acronyms used in this document.

ACRONYM	DESCRIPTION
HVPS	High Voltage Power Supply
L2D	Launch-to-Date

Last Terms, Abbreviations and Definitions Page

B Processing Scripts and Code

B.1 doit.master

```

1: #!/bin/ksh
2:
3:
4:
5:
6: print "### extract the primary data sets"
7: for year in $( seq 1999 2003 )
8: do
9:   CMD=" extract_data.ksh $year"
10:  print $CMD
11:  #XXX $CMD
12: done
13: print
14:
15:
16:
17:
18: date
19: print "### do global post processing"
20: #
21: # input:
22: #   yyyy_hvps.rdb
23: # output:
24: #   yyyy_hvps.sdt.rdb   - csdt rdb file
25: #   yyyy_hvps.csdt.f1.rdb - csdt w/ 0 continuity events removed
26:
27: print " global_post_process_data.ksh"
28: #XXX global_post_process_data.ksh
29: print
30:
31:
32:
33: date
34: print "### Generate the cumulative hvps state changes for each year"
35: for year in $( seq 1999 2003 )
36: do
37:
38:  print " gen-cum_hvps_sc.pl DATA/${year}_hvps.csdt.f1.rdb > DATA/${year}_
_cum_hvps_sc.rdb"
39:  #XXX gen-cum_hvps_sc.pl DATA/${year}_hvps.csdt.f1.rdb > DATA/${year}_cum
_hvps_sc.rdb
40: done
41: print
42:
43:
44:
45:
46:
47: date
48: print "### Generate the cumulative hvps state changes for the launch-to-da
te "
49: for year in $( seq 1999 2003 )
50: do
51:   FILES="$FILES DATA/${year}_hvps.csdt.f1.rdb"
52: done
53: print " gen-cum_hvps_sc.pl $FILES > DATA/ltd_cum_hvps_sc.rdb"
54: #XXX gen-cum_hvps_sc.pl $FILES > DATA/ltd_cum_hvps_sc.rdb
55: print
56:
57:
58: print "### Generate the graphical products"
59: for year in $( seq 1999 2003 )
60: do
61:  print " csdtrdb2gp -ylabel 'Cumulative State Changes' DATA/${year}_cum_
hvps_sc.rdb"
62:  #XXX csdtrdb2gp -ylabel 'Cumulative State Changes' DATA/${year}_cum_hvps
_sc.rdb
63: done
64: print " csdtrdb2gp -ylabel 'Cumulative State Changes' DATA/ltd_cum_hvps_s
c.rdb"
65: #XXX csdtrdb2gp -ylabel 'Cumulative State Changes' DATA/ltd_cum_hvps_sc.
rdb
66: print
67:
68:
69:
70:
71:

```

```

72: print "### Generate the cumulative hvps sec on at max voltage for each year"
r"
73: for year in $( seq 1999 2003 )
74: do
75:   print " gen-cum_hvps_sec_at_step.pl DATA/${year}_hvps.csdt.f1.rdb > DATA/${year}_cum_hvps_sec.rdb"
76:   gen-cum_hvps_sec_at_step.pl DATA/${year}_hvps.csdt.f1.rdb > DATA/${year}_cum_hvps_sec.rdb
77:
78:   print "csdtrdb2gp -ylabel 'Cumulative Time (sec)' -every 100 DATA/${year}_cum_hvps_sec.rdb"
79:   csdtrdb2gp -ylabel 'Cumulative Time (sec)' -every 100 DATA/${year}_cum_hvps_sec.rdb
80:
81:   print
82: done
83: print
84:
85:
86:
87: FILES="";
88: print "### Generate the cumulative hvps sec at max voltage for the launch-to-date "
89: for year in $( seq 1999 2003 )
90: do
91:   FILES="$FILES DATA/${year}_hvps.csdt.f1.rdb"
92: done
93: print " gen-cum_hvps_sec_at_step.pl $FILES > DATA/ltd_cum_hvps_sec.rdb"
94: gen-cum_hvps_sec_at_step.pl $FILES > DATA/ltd_cum_hvps_sec.rdb
95:
96: print "csdtrdb2gp -ylabel 'Cumulative Time (sec)' -every 100 DATA/ltd_cum_hvps_sec.rdb"
97: csdtrdb2gp -ylabel 'Cumulative Time (sec)' -every 100 DATA/ltd_cum_hvps_sec.rdb
98:
99:
100:
101:
102:
103: print "### Generate the hvps step for each year"
104: for year in $( seq 1999 2003 )
105: do
106:   print "ncol csdt V_ImTopHVStep V_ImBotHVStep V_ImBotHV V_ImTopHV < DATA/${year}_hvps.csdt.f1.rdb > DATA/${year}_I_hvps.rdb"
107:   ncol csdt V_ImTopHVStep V_ImBotHVStep V_ImBotHV V_ImTopHV < DATA/${year}_hvps.csdt.f1.rdb > DATA/${year}_I_hvps.rdb
108:
109:   print "ncol csdt V_SpTopHVStep V_SpBotHVStep V_SpBotHV V_SpTopHV < DATA/${year}_hvps.csdt.f1.rdb > DATA/${year}_S_hvps.rdb"
110:   ncol csdt V_SpTopHVStep V_SpBotHVStep V_SpBotHV V_SpTopHV < DATA/${year}_hvps.csdt.f1.rdb > DATA/${year}_S_hvps.rdb
111:
112:   print "ncol csdt V_ShldBHV V_ShldBHVStep < DATA/${year}_hvps.csdt.f1.rdb > DATA/${year}_Shld_hvps.rdb"
113:   ncol csdt V_ShldBHV V_ShldBHVStep < DATA/${year}_hvps.csdt.f1.rdb > DATA/${year}_Shld_hvps.rdb
114:
115:   print "csdtrdb2gp -ylabel 'Cmd Step | DN Level' -every 1000 DATA/${year}_I_hvps.rdb"
116:   csdtrdb2gp -ylabel 'Cmd Step or DN Level' -every 1000 DATA/${year}_I_hvps.rdb
117:
118:   print "csdtrdb2gp -ylabel 'Cmd Step | DN Level' -every 1000 DATA/${year}_S_hvps.rdb"
119:   csdtrdb2gp -ylabel 'Cmd Step or DN Level' -every 1000 DATA/${year}_S_hvps.rdb
120:
121:   print "csdtrdb2gp -ylabel 'Cmd Step | DN Level' -every 1000 DATA/${year}_Shld_hvps.rdb"
122:   csdtrdb2gp -ylabel 'Cmd Step or DN Level' -every 1000 DATA/${year}_Shld_hvps.rdb
123:
124:   print
125: done
126: print
127:
128:
129:
130:
131:
132:
133: print
134:
135:

```

136:

B.2 extract_data.ksh

```

1: #!/bin/ksh
2:
3:
4: if [ ! $1 ]
5: then
6:   print "usage: ./extract_data yyyy"
7:   exit 1
8: fi
9:
10: YY=$1
11:
12: ### set the msid definition file
13: MSID_FILE=hvps
14:
15: ### set the major frame sample rate (ie.every n MF)
16: MFMOD=20
17:
18: ### set the file and directory names
19: AD=/ARCHIVE/ftm
20: GTV_DIR=/d2/hrc/near1
21: DD=/d2/hrc/ANA/hvps/DATA
22: DF="{YY}_{MSID_FILE}.rdb"
23:
24:
25:
26: COLS="Day HH MM SS Sec vcdv MF mf fmt
27: S_ImHvps
28: S_SpHvps
29: S_ShldA
30: S_ShldB
31: V_ImTopHVStep V_ImBotHVStep V_ImBotHV V_ImTopHV
32: V_SpTopHVStep V_SpBotHVStep V_SpBotHV V_SpTopHV
33: V_ShldAHV
34: V_ShldAHVStep
35: V_ShldBHV
36: V_ShldBHVStep
37: "
38:
39: ### get a list of the input .ftm files
40: FTM_FILES=$( find $AD/$YY/ -name '*.tm.gz' -print )
41:
42: ### check for data dir
43: if [ ! -d $DD ]
44: then
45:   mkdir $DD
46: fi
47:
48:
49: ### remove the data file
50: if [ -f $DD/$DF ]
51: then
52:   rm -i $DD/$DF
53: fi
54:
55:
56: #cd $GTV_DIR
57:
58: ### loop thru each input file and extract the requested msids
59: FTT=1
60: for FTM_FILE in $FTM_FILES
61: do
62:   # get the rdb header on the first file
63:   if (( FTT == 1 ))
64:   then
65:     #gzcat $FTM_FILE | $GTV_DIR/gtv -year $YY -d $GTV_DIR/DB -u $MFMOD -p
66:     #MSID_FILE | grep -v '#' | head -2 > $DD/$DF
67:     #ad -2 >> $DD/$DF
68:     #FTT=0
69:     #fi
70:     #CSDT=$( date +%Y:%j:%H:%M:%S' )
71:     #print "$CSDT - $SECONDS - $FTM_FILE"
72:     #gzcat $FTM_FILE |
73:     #ftm2ssdrdb -b -h -c |
74:     #ncol $COLS |
75:     #egrep -v '(#|vcdv|--)' >> $DD/$DF
76:
77: done
78:
79:

```

80:

B.3 global_post_process_data.ksh

```

1: #!/bin/ksh
2:
3:
4:
5: DD=DATA
6: YEARS="1999 2003"
7:
8:
9: ### convert the ftmrdb into csdt rdb formatted file
10: print "Convert the ftmrdb into csdt rdb formatted file"
11: for year in $( seq $YEARS )
12: do
13:   print ".tmdb2csdtrdb $year < $DD/${year}_hvps.rdb > $DD/${year}_hvps.c
sdt.rdb"
14:   ./tmdb2csdtrdb $year < $DD/${year}_hvps.rdb > $DD/${year}_hvps.csdt.rdb
15:
16: done
17: print
18: print
19: print
20:
21:
22:
23:
24:
25:
26:
27: ### Data Filter 1
28: ### remove the zero events from the telemetry - continuity 0's
29: print "remove the zero events from the telemetry - continuity 0's"
30: for year in $( seq $YEARS )
31: do
32:   if (( $year == 2001 ))
33:   then
34:     print "Nrow '(V_ShldAHV != 0)' < $DD/${year}_hvps.csdt.rdb | egrep -v
"2001:351" > $DD/${year}_hvps.csdt.f1.rdb"
35:     Nrow '(V_ShldAHV != 0)' < $DD/${year}_hvps.csdt.rdb | egrep -v "2001:
351" > $DD/${year}_hvps.csdt.f1.rdb
36:   else
37:     print "Nrow '(V_ShldAHV != 0)' < $DD/${year}_hvps.csdt.rdb > $DD/${yea
r}_hvps.csdt.f1.rdb"
38:     Nrow '(V_ShldAHV != 0)' < $DD/${year}_hvps.csdt.rdb > $DD/${year}_hvps
.csdt.f1.rdb
39:     fi
40:
41:     #print "Nrow '(V_ImBotHV != 0.00)' < $DD/${year}_hvps.csdt.rdb > $DD/${y
ear}_hvps.csdt.f1.rdb"
42:     #Nrow '(V_ImBotHV != 0.00)' < $DD/${year}_hvps.csdt.rdb > $DD/${year}_hv
ps.csdt.f1.rdb
43:   done
44:   print
45:   print
46:   print
47:
48:
49:
50:
51:
52:
53:
54:
55:

```

B.4 gen-cum_hvps_sc.pl

```

1: #!/usr/bin/perl -w
2:
3: #=====
==
4: #
5: #   J. Chappell (jhc)
6: #
7: #   Program: template1.pl
8: #   Purpose: perl template file
9: #   external files:
10: #   external programs:
11: #
12: #   Copyright: 2001 jhc
13: #   You may do anything you like with this file except remove
14: #   this copyright.
15: #   jhc makes no representations about the suitability of this
16: #   software for any purpose. It is provided "as is" without
17: #   express or implied warranty.
18: #
19: #
20: # === RCS Information ===
21: # $Revision: 1.1 $
22: # $Date: 2001/09/03 17:36:51 $
23: # -----
24: # $Header: /home/jhc/near1/proj/CM/Templates/PerlScripts/RCS/template1.pl,
v 1.1 2001/09/03 17:36:51 jhc Exp $
25: # $Log: template1.pl,v $
26: # Revision 1.1 2001/09/03 17:36:51 jhc
27: # Initial revision
28: #
29: # =====
30: #
31: #
32: #=====
==
33:
34:
35: use strict;
36: use FileHandle;
37: use Getopt::Long;
38:
39:
40:
41: { ### start of main program
42:
43:   ### define variables for main program
44:   my $RCS = '$Revision: 1.1 $';
45:   my %IP;
46:   my $md5sum;
47:   my $P;
48:   my @Files;
49:   my @a;
50:   my $step;
51:   my $Time;
52:   my $rv;
53:
54:
55:
56:   ### set the default runtime values
57:   $IP{Prog} = 'bin_data1.pl';
58:   $IP{Description} = 'Bin Solar Array Angle Time';
59:   $IP{RCS} = $RCS;
60:   $IP{Debug} = 0;
61:   $IP{PS} = '1-5';
62:   $IP{PSMax} = 5;
63:   $IP{BinSize} = 15;
64:   $IP{Start} = 1;
65:   $IP{Stop} = 365;
66:
67:
68:   ### get the command line options
69:   if( ! GetOptions(
70:     "ps=s" => \$IP{PS},
71:     "d|debug=s" => \$IP{Debug},
72:     "h|help=s" => \$IP{Help},
73:     "binsize=s" => \$IP{BinSize},
74:     "start=s" => \$IP{Start},
75:     "stop=s" => \$IP{Stop},
76:   ) ){
77:     $rv = PrnUsage(\@ARGV, \%IP);
78:     exit(1);

```

```

79: }
80:
81: ### process the command line options
82: # get the process steps
83: $IP{PSSSteps} = GetProcessSteps(%IP);
84:
85:
86: ### Print program header
87: #PrnHeader(\@ARGV, \%IP);
88:
89:
90: ### Print md5 checksums of the input data files
91: #push( @Files, $IP{Prog} );
92: #push( @Files, $ARGV[0] );
93: #PrnMD5( @Files );
94:
95:
96:
97: ### loop thru each process step
98: $Time = 'date';
99: @a = split(" ", $IP{PSSSteps});
100: foreach $step (@a){
101:     $Time = 'date';
102:
103:     ### process step 1
104:     if( $step == 1 ){
105:         Step_1(\%IP);
106:
107:     }
108:
109: }
110:
111:
112: exit(0);
113:
114:
115: } ### end of main program
116:
117:
118:
119:
120: #-----
====
121: sub Step_1
122: {
123:
124:     my ($ip) = @_;
125:     my $yyyy;
126:     my $doy;
127:     my $hh;
128:     my $mm;
129:     my $ss;
130:     my @col;
131:     my %IP;
132:     my $last_bin;
133:     my $last_sec;
134:     my $csdt;
135:     my $saa;
136:     my $sec;
137:     my $start_sec;
138:     my $stop_sec;
139:     my $disec;
140:     my $dssec;
141:     my $bin;
142:     my %binsum;
143:     my $p;
144:     my $ll;
145:     my $ul;
146:     my $sum_sc_i;
147:     my $sum_sc_s;
148:     my $sum_sc_shlda;
149:     my $sum_sc_shldb;
150:     my $sum_sec_i = 0;
151:     my $sum_sec_s = 0;
152:     my $sum_sec_shlda = 0;
153:     my $sum_sec_shldb = 0;
154:     my $S_ImHvps;
155:     my $S_SpHvps;
156:     my $S_ShldA;
157:     my $S_ShldB;
158:     my $last_S_ImHvps;
159:     my $last_S_SpHvps;
160:     my $last_S_ShldA;
161:     my $last_S_ShldB;

```

```

162: my $start_ImHvps;
163: my $stop_ImHvps;
164: my $start_SpHvps;
165: my $stop_SpHvps;
166:
167: %IP = %$ip;
168:
169:
170: ### loop thru the input file
171: $p = 0;
172: $sum_sc_i = 0;
173: $sum_sc_s = 0;
174: $sum_sc_shlda = 0;
175: $sum_sc_shldb = 0;
176: $sum_sec_i = 0;
177: $sum_sec_s = 0;
178: $last_S_ImHvps = -1;
179: $last_S_SpHvps = -1;
180: $last_S_ShldA = -1;
181: $last_S_ShldB = -1;
182: $start_ImHvps = 0;;
183: $stop_ImHvps = 0;;
184: $start_SpHvps = 0;;
185: $stop_SpHvps = 0;;
186:
187: print "csdt\t";
188: print "sum_sc_i\t";
189: print "sum_sc_s\t";
190: print "sum_sc_shldb\n";
191:
192: print "---\t";
193: print "---\t";
194: print "---\t";
195: print "---\n";
196:
197: while(<>){
198:
199:   # skip the rdb header
200:   if( $_ =~ m/csdt/ ){next;}
201:   if( $_ =~ m/^\-/\ ){next;}
202:   if( $_ =~ m/^\#/ ){next;}
203:
204:   # split the row
205:   chomp();
206:   @col = split("\t");
207:
208:   # split the CSDT field
209:   #( $yyyy, $doy, $hh, $mm, $ss ) = split(":",$col[0]);
210:
211:   $S_ImHvps = $col[11];
212:   $S_SpHvps = $col[12];
213:   $S_ShldA = $col[13];
214:   $S_ShldB = $col[14];
215:
216:
217:   ### check for a I state change
218:   if( $S_ImHvps != $last_S_ImHvps ){
219:     $sum_sc_i++;
220:     $p = 1;
221:   }
222:
223:
224:
225:   ### check for a S state change
226:   if( $S_SpHvps != $last_S_SpHvps ){
227:     $sum_sc_s++;
228:     $p = 1;
229:   }
230:
231:
232:
233:   ### check for a shldB state change
234:   if( $S_ShldB != $last_S_ShldB ){
235:     $sum_sc_shldb++;
236:     $p = 1;
237:   }
238:
239:
240:   ### save the last state values
241:   $last_S_ImHvps = $S_ImHvps;
242:   $last_S_SpHvps = $S_SpHvps;
243:   $last_S_ShldB = $S_ShldB;
244:
245:   if( $p == 1 ){

```

```

246:     printf "%s\t%d\t%d\t%d\t%d\n",
247:         $col[0],
248:         $sum_sc_i,
249:         $sum_sc_s,
250:         $sum_sc_shldb
251:     ;
252:     $p = 0;
253: }
254: }
255:
256: }
257:
258: #-----
====
259: sub GetProcessSteps
260: {
261:     my (%IP) = @_;
262:
263:     ### define local vars
264:     my @a;
265:     my @b;
266:     my @c;
267:     my $i;
268:     my $j;
269:     my $maxps;
270:     my $rv;
271:
272:
273:     $maxps = $IP{PSMax};
274:
275:     # cmd line fmt: 1,2,4-5,6-,-8
276:
277:     # split the command line arg on ','s
278:     @a = split(",", $IP{PS});
279:
280:     # loop thru each comma group:
281:     for($i=0; $i<=$#a; $i++){
282:
283:
284:         # check if the comma group has a dash, if not push it in the list
285:         if( $a[$i] =~ /\-\/ ){
286:
287:             # split the comma group on a -
288:             @b = split("-", $a[$i]);
289:
290:             if( $a[$i] =~ /\-\/ ){
291:                 # dash at start
292:                 for($j=1; $j<=$b[1]; $j++){ push(@c, $j); }
293:
294:             }elseif( $a[$i] =~ /\-$\/ ){
295:                 # dash at end
296:                 for($j=1; $j<=$maxps; $j++){ push(@c, $j); }
297:
298:             }else{
299:                 # dash at mid
300:                 for($j=$b[0]; $j<=$b[1]; $j++){ push(@c, $j); }
301:
302:             }
303:
304:         }else{
305:             push(@c, $a[$i]);
306:         }
307:     }
308:
309:     return (@c);
310: }
311:
312:
313:
314:
315: #-----
====
316: sub PrnMD5
317: {
318:     my ( @Files ) = @_;
319:     my $Time;
320:     my $file;
321:     my $md5sum;
322:     my $rv;
323:
324:     ### set the default values
325:     $Time = '';
326:     $file = '';
327:

```

```

328: $rv = 0;
329:
330: ### calculate the md5 checksum of the input data files
331: $Time = 'date';
332: chomp( $Time );
333: print "##### $Time\n";
334: print "# Calculating Electronic Signature (md5sum) of the input files:\n
";
335:
336:
337: ### calculate the md5 checksum of the cmd delog file
338: foreach $file (@Files){
339:     if( -e $file ){
340:         $md5sum = `md5sum $file`;
341:         chomp($md5sum);
342:         print "# $md5sum\n";
343:     }else{
344:         print "# File does not exist: $file\n";
345:     }
346: }
347:
348:
349: }
350:
351:
352:
353: #####
====
354: sub ReadFile
355: {
356:
357:     my ( $Time, $time,$sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst )
;
358:     my ($RDBFILE) = @_;
359:     my @a;
360:
361:     open( FD, $RDBFILE ) || die "Can't open OR file:$RDBFILE\n";
362:     @a = <FD>;
363:     close(FD);
364:
365:     return(@a);
366: }
367:
368: #####
====
369: sub PrnHeader
370: {
371:
372:     my ($argv, $ip) = @_;
373:
374:     my %IP;
375:     my @ARGV;
376:     my $md5sum;
377:     my $P;
378:     my @a;
379:     my $PRCS;
380:     my $Time;
381:     my $host;
382:     my $cwd;
383:
384:     %IP = %$ip;
385:     @ARGV = @$argv;
386:
387:     ### get the starting info
388:     $Time = 'date';
389:     $host = 'hostname';
390:     $cwd = 'pwd';
391:
392:
393:
394:     ### print the program header and ext progs info
395:     print "#####\n";
396:     print "# Generating XXX Products\n";
397:     print "## Cmdline: $IP{Prog} @ARGV\n";
398:     print "# Run Time: $Time";
399:     print "# Machine: $host";
400:     print "# Runtime Directory: $cwd";
401:     print "# Command Line: @ARGV\n";
402:     print "#\n";
403:     print "#\n";
404:     print "# Program(s) Location and Version:\n" ;
405:     print "#   Program - Version\n" ;
406:     print "# -----\n";
407:     # OS

```

```

408: $P = "OS";
409: chomp($P);
410: $PRCS = 'cat /etc/issue';
411: chomp($PRCS);
412: $PRCS =~ s/\n/ /g;
413: print "# $P - $PRCS\n";
414:
415: # perl
416: $P = 'which perl';
417: chomp($P);
418: $PRCS = 'perl --version | egrep '(version|^This)'';
419: chomp($PRCS);
420: @a = split(' ', $PRCS);
421: print "# $P - $a[3]\n";
422:
423: # md5sum
424: $P = 'which md5sum';
425: chomp($P);
426: $PRCS = 'md5sum --version | grep md5sum';
427: chomp($PRCS);
428: @a = split(' ', $PRCS);
429: print "# $P - $a[2]\n";
430:
431: # this prog
432: $P = 'which $IP{Prog}';
433: chomp($P);
434: print "# $P - $IP{RCS}\n";
435: print "#####\n";
;
436: print "#\n";
437:
438: }
439:
440: #####
====
441: sub PrnUsage
442: {
443:   my ($argv, $ip) = @_;
444:   my %IP;
445:   my @ARGV;
446:
447:   %IP = %$ip;
448:   @ARGV = @$argv;
449:
450:   print "-----\n";
451:   print "$IP{Prog} ($IP{RCS}) $IP{Description}\n";
452:   print "-----\n";
453:   print "usage: $IP{Prog} [Options] file.rdb\n";
454:   print "Options: [defaults in brackets after descriptions]\n";
455:   print "  -ps steps   Set the process steps to be executed; \n";
456:   print "           examples: (note no spaces allowed in argument field)
\n";
457:   print "           -ps 1,4-6,8,9   process steps 1,4,5,6,8,9 \n";
458:   print "           -ps 4-         process steps 4,5,6,...,$IP{PSMax}
\n";
459:   print "           -ps 3,6       process steps 3,6 \n";
460:   print "  -dlddebug   Set the program debug level [$IP{Debug}] \n";
461:   print "  -v|version  Report the program version [$IP{RCS}] \n";
462:   print "  -h|help    Print this message\n";
463:   print "  -binsize   Set the SAA bin size angle (deg)[$IP{BinSize}] \n";
464:   print "  -start     Set the DOY start time [$IP{Start}] \n";
465:   print "  -stop      Set the DOY stop time  [$IP{Stop}] \n";
466:   print "\n\n";
467:   print "  To generate the total time vs solar array angle (saa) over a sle
cted time slice [year doyl]:\n";
468:   print "\n\n";
469:   print "$IP{Prog} DATA/2000_400.msids.rdb\n";
470:   print "$IP{Prog} < DATA/2000_400.msids.rdb\n";
471:   print "\n\n";
472:   print "\n\n";
473:
474:
475:   return(0);
476:
477: }
478:
479: #####
====
480: sub csdtminus
481: {
482:   my ($start, $stop) = @_;
483:
484:   my $yyyy;
485:   my $doy;

```

```
486: my $hh;
487: my $mm;
488: my $ss;
489: my %Start;
490: my %Stop;
491: my $dsec;
492:
493:
494: ( $yyyy, $doy, $hh, $mm, $ss ) = split("\.", $start);
495: $Start{yyyy} = $yyyy;
496: $Start{doy} = $doy;
497: $Start{hh} = $hh;
498: $Start{mm} = $mm;
499: $Start{ss} = $ss;
500:
501:
502: ( $yyyy, $doy, $hh, $mm, $ss ) = split("\.", $stop);
503: $Stop{yyyy} = $yyyy;
504: $Stop{doy} = $doy;
505: $Stop{hh} = $hh;
506: $Stop{mm} = $mm;
507: $Stop{ss} = $ss;
508:
509:
510: $dsec = $Stop{ss} - $Start{ss};
511: $dsec = $dsec + ($Stop{mm} - $Start{mm})*60;
512: $dsec = $dsec + ($Stop{hh} - $Start{hh})*3600;
513: $dsec = $dsec + ($Stop{doy} - $Start{doy})*3600*24;
514:
515:
516:
517: return(abs($dsec));
518: }
```

B.5 gen-cum_hvps_sec_at_step.pl

```

1: #!/usr/bin/perl -w
2:
3: #=====
==
4: #
5: #   J. Chappell (jhc)
6: #
7: #   Program: template1.pl
8: #   Purpose: perl template file
9: #   external files:
10: #   external programs:
11: #
12: #   Copyright: 2001 jhc
13: #   You may do anything you like with this file except remove
14: #   this copyright.
15: #   jhc makes no representations about the suitability of this
16: #   software for any purpose. It is provided "as is" without
17: #   express or implied warranty.
18: #
19: #
20: # === RCS Information ===
21: # $Revision: 1.1 $
22: # $Date: 2001/09/03 17:36:51 $
23: # -----
24: # $Header: /home/jhc/near1/proj/CM/Templates/PerlScripts/RCS/template1.pl,
v 1.1 2001/09/03 17:36:51 jhc Exp $
25: # $Log: template1.pl,v $
26: # Revision 1.1 2001/09/03 17:36:51 jhc
27: # Initial revision
28: #
29: # =====
30: #
31: #
32: #=====
==
33:
34:
35: use strict;
36: use FileHandle;
37: use Getopt::Long;
38:
39:
40:
41: { ### start of main program
42:
43:   ### define variables for main program
44:   my $RCS = '$Revision: 1.1 $';
45:   my %IP;
46:   my $md5sum;
47:   my $P;
48:   my @Files;
49:   my @a;
50:   my $step;
51:   my $Time;
52:   my $rv;
53:
54:
55:
56:   ### set the default runtime values
57:   $IP{Prog} = 'bin_data1.pl';
58:   $IP{Description} = 'Bin Solar Array Angle Time';
59:   $IP{RCS} = $RCS;
60:   $IP{Debug} = 0;
61:   $IP{PS} = '1-5';
62:   $IP{PSMax} = 5;
63:   $IP{BinSize} = 15;
64:   $IP{Start} = 1;
65:   $IP{Stop} = 365;
66:
67:
68:   ### get the command line options
69:   if( ! GetOptions(
70:     "ps=s" => \$IP{PS},
71:     "d|debug=s" => \$IP{Debug},
72:     "h|help=s" => \$IP{Help},
73:     "binsize=s" => \$IP{BinSize},
74:     "start=s" => \$IP{Start},
75:     "stop=s" => \$IP{Stop},
76:   ) ){
77:     $rv = PrnUsage(\@ARGV, \%IP);
78:     exit(1);

```

```

79: }
80:
81: ### process the command line options
82: # get the process steps
83: $IP{PSSSteps} = GetProcessSteps(%IP);
84:
85:
86: ### Print program header
87: #PrnHeader(\@ARGV, \%IP);
88:
89:
90: ### Print md5 checksums of the input data files
91: #push( @Files, $IP{Prog} );
92: #push( @Files, $ARGV[0] );
93: #PrnMD5( @Files );
94:
95:
96:
97: ### loop thru each process step
98: $Time = 'date';
99: @a = split(" ", $IP{PSSSteps});
100: foreach $step (@a){
101:     $Time = 'date';
102:
103:     ### process step 1
104:     if( $step == 1 ){
105:         Step_1(\%IP);
106:
107:     }
108:
109: }
110:
111:
112:
113: exit(0);
114:
115: } ### end of main program
116:
117:
118:
119:
120: #-----
====
121: sub Step_1
122: {
123:
124:     my ($ip) = @_;
125:     my $yyyy;
126:     my $doy;
127:     my $hh;
128:     my $mm;
129:     my $ss;
130:     my @col;
131:     my @CSDT;
132:     my %IP;
133:     my $cur_csd;
134:     my $last_csd;
135:     my $last_sec;
136:     my $csdt;
137:     my $saa;
138:     my $sec;
139:     my $start_sec;
140:     my $stop_sec;
141:     my $dsecI;
142:     my $dsecS;
143:     my $dsecSh;
144:     my $bin;
145:     my %binsum;
146:     my $p;
147:     my $ll;
148:     my $ul;
149:     my $sum_sc_i;
150:     my $sum_sc_s;
151:     my $sum_sc_shlda;
152:     my $sum_sc_shldb;
153:     my $sum_sec_i = 0;
154:     my $sum_sec_s = 0;
155:     my $sum_sec_shlda = 0;
156:     my $sum_sec_shldb = 0;
157:     my $$ImHvps;
158:     my $$SpHvps;
159:     my $$Shlda;
160:     my $$Shldb;
161:     my $last_S_ImHvps;

```

```

162: my $last_S_SpHvps;
163: my $last_S_ShldA;
164: my $last_S_ShldB;
165: my $start_ImHvps;
166: my $stop_ImHvps;
167: my $start_SpHvps;
168: my $stop_SpHvps;
169: my %I;
170: my %S;
171: my %Sh;
172: my $step;
173: my $Istep;
174: my $$step;
175: my $$shstep;
176: my $V_ImTopHVStep;
177: my $V_ImBotHVStep;
178: my $ftt;
179: my $V_SpTopHVStep;
180: my $V_SpBotHVStep;
181: my $V_ShldBHVStep;
182: my $hrs;
183:
184: %IP = %$ip;
185:
186:
187: ### loop thru the input file
188: $p = 0;
189: $sum_sc_i = 0;
190: $sum_sc_s = 0;
191: $sum_sc_shlda = 0;
192: $sum_sc_shldb = 0;
193: $sum_sec_i = 0;
194: $sum_sec_s = 0;
195: $last_S_ImHvps = -1;
196: $last_S_SpHvps = -1;
197: $last_S_ShldA = -1;
198: $last_S_ShldB = -1;
199: $start_ImHvps = 0;;
200: $stop_ImHvps = 0;;
201: $start_SpHvps = 0;;
202: $stop_SpHvps = 0;;
203:
204: print "csdt\t";
205: print "sum_i\t";
206: print "sum_s\t";
207: print "sum_shldb\n";
208:
209: print "---\t";
210: print "---\t";
211: print "---\t";
212: print "---\n";
213:
214: $Istep = 77;
215: $$step = 90;
216: $$shstep = 8;
217:
218: $ftt = 1;
219: $p = 0;
220: while(<>){
221:
222:     # skip the rdb header
223:     if( $_ = " m/csdt/ ){next;}
224:     if( $_ = " m/^-/ ){next;}
225:     if( $_ = " m/^#/ ){next;}
226:     if( $_ = " m/^Step/ ){next;}
227:
228:     # split the row
229:     chomp();
230:     @col = split("\t");
231:     @GSDT = split("\t", $col[0]);
232:
233:
234:     $cur_csdt          = $col[0];
235:     $V_ImTopHVStep    = $col[15];
236:     $V_ImBotHVStep    = $col[16];
237:     $V_SpTopHVStep    = $col[19];
238:     $V_SpBotHVStep    = $col[20];
239:     $V_ShldBHVStep    = $col[26];
240:
241:
242:     if( $ftt == 1 ){
243:         #print "Start Time: $cur_csdt\n";
244:         $ftt = 0;
245:     }

```

```

246:
247:   if( ! defined( $last_csdt ) ){
248:     $last_csdt = $cur_csdt;
249:   }
250:
251:   if( ! defined( ${I${Istep}{last_csdt}} ) ){
252:     ${I${Istep}{last_csdt}} = $cur_csdt;
253:   }
254:
255:   if( ! defined( ${S${Sstep}{last_csdt}} ) ){
256:     ${S${Sstep}{last_csdt}} = $cur_csdt;
257:   }
258:
259:   if( ! defined( ${Sh${Shstep}{last_csdt}} ) ){
260:     ${Sh${Shstep}{last_csdt}} = $cur_csdt;
261:   }
262:
263:
264:
265:   if( ! defined( ${I${Istep}{sum}} ) ){
266:     ${I${Istep}{sum}} = 0;
267:   }
268:
269:   if( ! defined( ${S${Sstep}{sum}} ) ){
270:     ${S${Sstep}{sum}} = 0;
271:   }
272:
273:   if( ! defined( ${Sh${Shstep}{sum}} ) ){
274:     ${Sh${Shstep}{sum}} = 0;
275:   }
276:
277:
278:
279:
280:   if( ($V_ImTopHVStep == 77) ){
281:     ### process the I detector
282:     # get the number of seconds since the last reporting peroid
283:     #dsecI = csdtminus($cur_csdt, ${I${Istep}{last_csdt}});
284:     $dsecI = csdtminus($cur_csdt, $last_csdt);
285:
286:     # increment the cumulative time at the current step
287:     if( ! defined( ${I${Istep}{sum}} ) ){
288:       ${I${Istep}{sum}} = 0;
289:     }
290:     ${I${Istep}{sum}} = ${I${Istep}{sum}} + $dsecI;
291:
292:
293:
294:     #printf "I %s\t%.2f\t%d\t%.2f\n",
295:     # $cur_csdt,
296:     # $dsecI,
297:     # $V_ImTopHVStep,
298:     # ${I${V_ImTopHVStep}{sum}},
299:     #;
300:
301:
302:     ### save the last csdt
303:     ${I${Istep}{last_csdt}} = $cur_csdt;
304:
305:     $p = 1;
306:   }
307:
308:
309:
310:
311:
312:   if( ($V_SpTopHVStep == 90) ){
313:     ### process the S detector
314:     # get the number of seconds since the last reporting peroid
315:     #dsecS = csdtminus($cur_csdt, ${S${Sstep}{last_csdt}});
316:     $dsecS = csdtminus($cur_csdt, $last_csdt);
317:
318:     # increment the cumulative time at the current step
319:     if( ! defined( ${S${Sstep}{sum}} ) ){
320:       ${S${Sstep}{sum}} = 0;
321:     }
322:     ${S${Sstep}{sum}} = ${S${Sstep}{sum}} + $dsecS;
323:
324:
325:
326:     #printf "%s\t%.2f\t%d\t%.2f\n",
327:     # $cur_csdt,
328:     # $dsecS,
329:     # $V_SpTopHVStep,

```

```

330:     # ${V_SpTopHVStep}{sum},
331:     #;
332:
333:
334:     ### save the last csdt
335:     ${S${Sstep}}{last_csdt} = $cur_csdt;
336:
337:     $p = 1;
338: }
339:
340:
341:
342:
343: if( ($V_ShldBHVStep >= 7) && ($V_ShldBHVStep <= 8) ){
344:     ### process the Sh detector
345:     # get the number of seconds since the last reporting peroid
346:     $dsecSh = csdtminus($cur_csdt, ${Sh}${Sstep}{last_csdt});
347:     $dsecS = csdtminus($cur_csdt, $last_csdt);
348:
349:     # increment the cumulative time at the current step
350:     if( ! defined( ${Sh}${Sstep}{sum} ) ){
351:         ${Sh}${Sstep}{sum} = 0;
352:     }
353:     ${Sh}${Sstep}{sum} = ${Sh}${Sstep}{sum} + $dsecSh;
354:     ${Sh}{sum} = ${Sh}${Sstep}{sum};
355:
356:
357:
358:     #if( $V_SpTopHVStep != 90 ){
359:     # printf "%s\t%.2f\t%.2f\t%.2f\n",
360:     #     $cur_csdt,
361:     #     $dsecS,
362:     #     $V_SpTopHVStep,
363:     #     ${V_SpTopHVStep}{sum},
364:     #     ;
365:     #}
366:
367:
368:     ### save the last csdt
369:     ${Sh}${Sstep}{last_csdt} = $cur_csdt;
370:
371:     $p = 1;
372: }
373:
374:     $last_csdt = $cur_csdt;
375:
376:
377:     if( $p == 1 ){
378:         printf ("%s\t%.2f\t%.2f\t%.2f\n", $cur_csdt, ${I${Istep}}{sum}, ${S${Sstep}}{sum}, ${Sh}${Sstep}{sum});
379:
380:         $p = 0;
381:     }
382:
383:
384: }
385: exit(0);
386:
387: print "# I Detector: \n";
388: print "step\tcumsec\tcumhrs\n";
389: print "----\t-----\t-----\n";
390: foreach $step (sort(keys(%I))){
391:
392:     $hrs = ${I{$step}}{sum}/3600;
393:     printf ("%d\t%d\t%.2f\n", $step, ${I{$step}}{sum},$hrs) ;
394: }
395:
396: print "\n";
397: print "# S Detector: \n";
398: print "step\tcumsec\tcumhrs\n";
399: print "----\t-----\t-----\n";
400: foreach $step (sort(keys(%S))){
401:
402:     $hrs = ${S{$step}}{sum}/3600;
403:     printf ("%d\t%d\t%.2f\n", $step, ${S{$step}}{sum},$hrs) ;
404: }
405:
406: print "\n";
407: print "# Shield Detector: \n";
408: print "step\tcumsec\tcumhrs\n";
409: print "----\t-----\t-----\n";
410: foreach $step (sort(keys(%Sh))){
411:
412:     $hrs = ${Sh{$step}}{sum}/3600;

```

```

413:     printf ("%d\t%d\t%.2f\n", $step, $Sh{$step}{sum},$hrs) ;
414: }
415:
416: print "\nEnd Time: $cur_csd\t\n";
417: exit(0);
418: }
419:
420: #=====
====
421: sub GetProcessSteps
422: {
423:
424:   my (%IP) = @_;
425:
426:   ### define local vars
427:   my @a;
428:   my @b;
429:   my @c;
430:   my $i;
431:   my $j;
432:   my $maxps;
433:   my $rv;
434:
435:
436:   $maxps = $IP{PSMax};
437:
438:   # cmd line fmt: 1,2,4-5,6,-,-8
439:
440:   # split the command line arg on ', 's
441:   @a = split(",",$IP{PS});
442:
443:   # loop thru each comma group:
444:   for($i=0; $i<=$#a; $i++){
445:
446:
447:     # check if the comma group has a dash, if not push it in the list
448:     if( $a[$i] =~ /\-/ ){
449:
450:       # split the comma group on a -
451:       @b = split("-", $a[$i]);
452:
453:       if( $a[$i] =~ /\^\/ ){
454:         # dash at start
455:         for($j=1; $j<=$b[1]; $j++){ push(@c,$j); }
456:
457:       }elseif( $a[$i] =~ /\-$/ ){
458:         # dash at end
459:         for($j=1; $j<=$maxps; $j++){ push(@c,$j); }
460:
461:       }else{
462:         # dash at mid
463:         for($j=$b[0]; $j<=$b[1]; $j++){ push(@c,$j); }
464:       }
465:
466:     }else{
467:       push(@c,$a[$i]);
468:     }
469:   }
470:
471:   return ("@c");
472: }
473: }
474:
475:
476:
477: #=====
====
478: sub PrnMD5
479: {
480:   my ( @Files ) = @_;
481:   my $Time;
482:   my $file;
483:   my $md5sum;
484:   my $rv;
485:
486:
487:   ### set the default values
488:   $Time = '';
489:   $file = '';
490:   $rv = 0;
491:
492:   ### calculate the md5 checksum of the input data files
493:   $Time = `date`;
494:   chomp( $Time );

```

```

495: print "#===== $Time\n";
496: print "# Calculating Electronic Signature (md5sum) of the input files:\n
";
497:
498:
499: ### calculate the md5 checksum of the cmd delog file
500: foreach $file (@Files){
501:   if( -e $file ){
502:     $md5sum = 'md5sum $file';
503:     chomp($md5sum);
504:     print "# $md5sum\n";
505:   }else{
506:     print "# File does not exist: $file\n";
507:   }
508: }
509:
510:
511: }
512:
513:
514:
515: #=====
=====
516: sub ReadFile
517: {
518:
519:   #my ( $Time, $time,$sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst )
;
520:   my ($RDBFILE) = @_;
521:   my @a;
522:
523:   open( FD, $RDBFILE ) || die "Can't open OR file:$RDBFILE\n";
524:   @a = <FD>;
525:   close(FD);
526:
527:   return(@a);
528: }
529:
530: #=====
=====
531: sub PrnHeader
532: {
533:
534:   my ($argv, $ip) = @_;
535:
536:   my %IP;
537:   my @ARGV;
538:   my $md5sum;
539:   my $P;
540:   my @a;
541:   my $PRCS;
542:   my $Time;
543:   my $host;
544:   my $cwd;
545:
546:   %IP = %$ip;
547:   @ARGV = @$argv;
548:
549:   ### get the starting info
550:   $Time = 'date';
551:   $host = 'hostname';
552:   $cwd = 'pwd';
553:
554:
555:
556:   ### print the program header and ext progs info
557:   print "#+++++++\n";
558:   print "# Generating XXX Products\n";
559:   print "## Cmdline: $IP{Prog} @ARGV\n";
560:   print "## Run Time: $Time";
561:   print "## Machine: $host";
562:   print "## Runtime Directory: $cwd";
563:   print "## Command Line: @ARGV\n";
564:   print "#\n";
565:   print "#\n";
566:   print "## Program(s) Location and Version:\n" ;
567:   print "##   Program - Version\n" ;
568:   print "## -----\n";
569:   # OS
570:   $P = "OS";
571:   chomp($P);
572:   $PRCS = 'cat /etc/issue';
573:   chomp($PRCS);
574:   $PRCS = " s/\n/ /g;

```

```

575: print "# $P - $PRCS\n";
576:
577: # perl
578: $P = 'which perl';
579: chomp($P);
580: $PRCS = 'perl --version | egrep '(version|^This)'';
581: chomp($PRCS);
582: @a = split(' ', $PRCS);
583: print "# $P - $a[3]\n";
584:
585: # md5sum
586: $P = 'which md5sum';
587: chomp($P);
588: $PRCS = 'md5sum --version | grep md5sum';
589: chomp($PRCS);
590: @a = split(' ', $PRCS);
591: print "# $P - $a[2]\n";
592:
593: # this prog
594: $P = 'which $IP{Prog}';
595: chomp($P);
596: print "# $P - $IP{RCS}\n";
597: print "#####\n";
;
598: print "#\n";
599:
600: }
601:
602: #=====
====
603: sub PrnUsage
604: {
605:   my ($argv, $ip) = @_;
606:   my %IP;
607:   my @ARGV;
608:
609:   %IP = %$ip;
610:   @ARGV = @$argv;
611:
612:   print "-----\n";
613:   print "$IP{Prog} ($IP{RCS}) $IP{Description}\n";
614:   print "-----\n";
615:   print "usage: $IP{Prog} [Options] file.rdb\n";
616:   print "Options: [defaults in brackets after descriptions]\n";
617:   print "  -ps steps    Set the process steps to be executed; \n";
618:   print "          examples: (note no spaces allowed in argument field)
\n";
619:   print "          -ps 1,4-6,8,9    process steps 1,4,5,6,8,9 \n";
620:   print "          -ps 4-          process steps 4,5,6,...,$IP{PSMax}
\n";
621:   print "          -ps 3,6          process steps 3,6 \n";
622:   print "  -d|debug      Set the program debug level [$IP{Debug}] \n";
623:   print "  -v|version    Report the program version [$IP{RCS}] \n";
624:   print "  -h|help      Print this message\n";
625:   print "  -binsize     Set the SAA bin size angle (deg)[$IP{BinSize}] \n";
626:   print "  -start       Set the DOY start time [$IP{Start}] \n";
627:   print "  -stop        Set the DOY stop time  [$IP{Stop}] \n";
628:   print "\n\n";
629:   print "  To generate the total time vs solar array angle (saa) over a sle
cted time slice [year doy]:\n";
630:   print "\n\n";
631:   print "$IP{Prog} DATA/2000_400.msids.rdb\n";
632:   print "$IP{Prog} < DATA/2000_400.msids.rdb\n";
633:   print "\n\n";
634:   print "\n\n";
635:
636:
637:   return(0);
638:
639: }
640:
641: #=====
====
642: sub csdtminus
643: {
644:   my ($START, $STOP) = @_;
645:
646:   my $yyyy;
647:   my $doy;
648:   my $hh;
649:   my $mm;
650:   my $ss;
651:   my %Start;
652:   my %Stop;

```

```

653: my $dsec;
654:
655:
656: my $Seconds;
657: my $secA;
658: my $secB;
659: my $start;
660: my $stop;
661: my $sign;
662: my $year;
663: my $days;
664: my $diffYrSec;
665:
666: ( $yyyy, $doy, $hh, $mm, $ss ) = split("\:",$START);
667: $Start{yyyy} = $yyyy;
668: $Start{doy} = $doy;
669: $Start{hh} = $hh;
670: $Start{mm} = $mm;
671: $Start{ss} = $ss;
672:
673:
674: ( $yyyy, $doy, $hh, $mm, $ss ) = split("\:",$STOP);
675: $Stop{yyyy} = $yyyy;
676: $Stop{doy} = $doy;
677: $Stop{hh} = $hh;
678: $Stop{mm} = $mm;
679: $Stop{ss} = $ss;
680:
681:
682:
683: # convert the a string into seconds
684: $secA = ($Start{doy} * 3600 * 24) + ($Start{hh} * 3600) + ($Start{mm}
* 60) + $Start{ss};
685: $secB = ($Stop{doy} * 3600 * 24) + ($Stop{hh} * 3600) + ($Stop{mm} * 6
0) + $Stop{ss};
686:
687:
688: # assign the stop and start year and sign */
689: if( $Start{yyyy} > $Stop{yyyy} ){
690:     $sign = +1;
691:     $start = int($Stop{yyyy});
692:     $stop = int($Start{yyyy});
693: }else{
694:     $sign = -1;
695:     $start = int($Start{yyyy});
696:     $stop = int($Stop{yyyy});
697: }
698:
699:
700: # add up the number of days between the start and stop years */
701: $days = 0;
702: $year = $start;
703: while( $year < $stop ){
704:     if( ($year % 4) == 0 ){
705:         $days = $days + 366;
706:     }else{
707:         $days = $days + 365;
708:     }
709:     $year++;
710: }
711:
712: # calculate the number of seconds in the total year-days */
713: $diffYrSec = ($days * 3600 * 24);
714:
715: # calculate the difference in the year seconds and day-hh-mm-ss second
s */
716: $Seconds = ($sign * $diffYrSec) + ( $secA - $secB );
717:
718:
719:
720:
721: return($Seconds);
722: }

```

B.6 csdtrdb2gp

```

1: #!/usr/bin/perl -w
2:
3: #=====
==
4: #
5: #   J. Chappell (jhc)
6: #
7: #   Program: eedrd2plotdp
8: #   Purpose: Perl script used to generate eed plot dp's from an rdb file
9: #   external files:
10: #   external programs:
11: #
12: #   Copyright: 1998 NEARL
13: #   You may do anything you like with this file except remove
14: #   this copyright.
15: #   NEARL makes no representations about the suitability of this
16: #   software for any purpose. It is provided "as is" without
17: #   express or implied warranty.
18: #
19: # === RCS Information ===
20: # $Revision: 1.1 $
21: # $Log: eedrd2plotdp,v $
22: # Revision 1.1 1999/12/07 19:08:36 jhc
23: # Initial revision
24: #
25: # =====
26: #
27: #=====
==
28:
29:
30: use strict;
31: use FileHandle;
32: use Getopt::Long;
33: {
34:
35: my %IP;
36: my $key;
37: my $prog;
38:
39: ### set the default values
40: $IP{Prog} = 'csdtrdb2gp';
41: $IP{PlotType} = 'ana';
42: $IP{XType} = 'time';
43: $IP{Every} = '1';
44: $IP{RDBFile} = '';
45:
46: ### get the command line options
47: if( ! GetOptions(
48:     "list=s" => \%IP{List},
49:     "title=s" => \%IP{Title},
50:     "ylabel=s" => \%IP{YLabel},
51:     "xlabel=s" => \%IP{XLabel},
52:     "ptype=s" => \%IP{PlotType},
53:     "logscale=s" => \%IP{Log},
54:     "xtype=s" => \%IP{XType},
55:     "xrange=s" => \%IP{XrangeString},
56:     "yrange=s" => \%IP{YrangeString},
57:     "every=s" => \%IP{Every},
58: ) ){
59:     print "IP{Prog}: Generates HRC Data Plots from an HRC TM RDB File\n";
60:     print "usage: $IP{Prog} [-switches] file.rdb\n";
61:     print " -list ['col_names'] Set column names to be plotted\n";
62:     print " -logscale [y] Set logscale y \n";
63:     print " -ptype [bilana] Set plot type\n";
64:     print " -xtype [vcdu|time] Set x-axis type\n";
65:     print " -xrange [ll:ul] Set x-range \n";
66:     print " -yrange [ll:ul] Set y-range \n";
67:     print " -xlabel Set x-label \n";
68:     print " -ylabel Set y-label \n";
69:     print " -title Set title\n";
70:     print " -xterm Set xterm display\n";
71:     print " -every [n] Set plot frequency\n";
72:     print "\n";
73:     exit(1);
74: }
75:
76:
77: ### check if the input rdb file exists
78: $IP{RDBFile} = $ARGV[0];
79: if( ! -f $IP{RDBFile} ){

```

```

80:   print "Input RDB Database file: $IP{RDBFile}; does not exist\n";
81:   exit(1);
82: }
83:
84:
85: MkRdbPlots( %IP );
86:
87: exit(0);
88: }
89:
90: #=====
91: sub MkRdbPlots {
92:
93:   my (%IP) = @_ ;
94:   my @COLS;
95:   my $File;
96:   my $XCol;
97:   my $pv;
98:   my %PV;
99:   my $col;
100:  my $s;
101:  my $i;
102:  my $j;
103:  my $startcol;
104:
105:
106:  ### get the root file name
107:  $File = $IP{RDBFile};
108:  $File =~ s/\.\rdb//;
109:
110:  print "Generating Plot Data Products From: $File.rdb\n";
111:  print "  -> $File.gp\n";
112:  print "  -> $File.ps\n";
113:  print "  -> $File.png\n";
114:
115:
116:  ### open the rdb file and get the col names and col numbers
117:  @COLS = split("\t",`head -250 $IP{RDBFile} | grep -v '#' | head -1`);
118:  chomp(@COLS[@COLS]);
119:  $i = 1;
120:  foreach $s (@COLS){
121:    $PV{$s}{col} = $i++;
122:  }
123:
124:
125:
126:  ### generate the gnuplot file
127:  open(FD,">$File.gp");
128:  print FD "set grid\n";
129:  print FD "set key outside below left\n";
130:  print FD "n = $IP{Every}\n";
131:
132:  if( defined($IP{Title}) ){
133:    print FD "set title '\$IP{Title}: Data File=$IP{RDBFile}\'\n";
134:  }else{
135:    print FD "set title '\Data File=$IP{RDBFile}\'\n";
136:  }
137:
138:  # set the Y-axis parameters
139:  if( defined($IP{YLabel}) ){
140:    print FD "set ylabel '\$IP{YLabel}\'\n";
141:  }
142:
143:  if( ($IP{PlotType} =~ /ana/) && (defined($IP{YrangeString})) ){
144:    print FD "set yrange [$IP{YrangeString}]\n";
145:  }
146:
147:
148:  # set the X-axis parameters
149:  if( $IP{XType} =~ /time/){
150:    print FD "set xdata time\n";
151:    print FD "set timefmt \"%Y:%j:%H:%M:%S\" \n";
152:    print FD "set xlabel \"%YYYY:YD\nHH:MM\" \n";
153:    print FD "set format x \"%Y:%j\n\n%H:%M\" \n";
154:    $XCol = 1;
155:  }else{
156:    print FD "set xlabel 'VCDU'\n";
157:    $XCol = 7;
158:  }
159:
160:  if( defined($IP{Xrange}) ){
161:    print "set xrange [$IP{XrangeString}]\n";
162:    print FD "set xrange [$IP{XrangeString}]\n";
163:  }

```

```

164:
165:
166:   if( defined($IP{Log}) ){
167:     print FD "set logscale $IP{Log}\n";
168:   }
169:
170:
171:
172:   # open the first output file
173:   print FD "set term post color\n";
174:   print FD "set output '$File.ps'\n";
175:
176:
177:   # generate the plot line
178:   $i = 0;
179:   $j = 0;
180:   $startcol = 2;
181:   foreach $col ( @COLS ){
182:     $i++;
183:
184:     if( $i < $startcol ){next;}
185:
186:
187:     if( $IP{PlotType} =~ /ana/ ){
188:       if( $i == $startcol ){
189:         print FD "plot '$IP{RDBFile}' ev n u $XCol:$PV{$col}{col} title '$
col' w st lw 2";
190:       }else{
191:         print FD ", \\n '$IP{RDBFile}' ev n u $XCol:$PV{$col}{col} title
'$col' w st lw 2";
192:       }
193:     }else{
194:       if( $i == $startcol ){
195:         print FD "plot '$IP{RDBFile}' ev n u $XCol:(\\$PV{$col}{col} + $j)
title '$col ($j)' w st lw 2";
196:       }else{
197:         print FD ", \\n '$IP{RDBFile}' ev n u $XCol:(\\$PV{$col}{col} + $
j) title '$col ($j)' w st lw 2";
198:       }
199:       $j = $j + 2;
200:     }
201:
202:
203:   }
204:   print FD "\n";
205:   #if( $IP{Xterm} > 0 ){
206:     # print FD "pause -1\n";
207:   #}
208:
209:   print FD "set term png\n";
210:   print FD "set output '$File.png'\n";
211:   print FD "replot\n";
212:
213:   close(FD);
214:
215:   ### run gnuplot
216:   system("gnuplot $File.gp");
217:
218:
219: }
220:

```