

How to Collect and Analyze Command Response Timing Data

8 July 2008

Eric Hamilton

Contents

1	Test Methodology	3
2	Format of Data in the Appendices	3
A	CommaOut.L	4
B	FixRaw.L	6
C	GetTimes.L	8
D	GetTimP.L	10
E	HGetTime.L	12
F	HJusTime.L	14
G	JustTime.L	16
H	JustTimP.L	18
I	MakeD3.L	20
J	NoColon.L	26
K	OnlyTime.L	27
L	Procit.bat	28
M	ProcitP.bat	30
N	HProcit.bat	32
O	ReDo.bat	34

¹\$Header: d:/Binder2/Timings/RCS/TimeHowT.tex,v 1.5 2008-07-08 11:50:51-07 Hamilton Exp Hamilton \$

P	ReDoL.bat	35
Q	Saveum.bat	36

1 Test Methodology

1. The GlassKeyboard, rev 2.36A, was used for all command generation.
2. The only commands sent were pan and tilt commands and these were sent in an approximately random manner. The sending consisted of moving the cursor rapidly around on the Glass Keyboard's simulated joy stick field until over 40 Kbytes of data were recorded.
3. There was a data capture PC monitoring the data flow both to and from the Spectra. It was set up to record all communications data to it disk. The on-disk data included a source id (DTE/DCE), time tag and the actual communications byte.
4. Usually when over 40 Kbytes² of data were recorded, the test was stopped and the data capture software was told to export the communications data. This resulted in a .TXT format data file of about 1.7 Mbytes.
5. The captured data was then analyzed using several programs and the results are shown in the appendices. The "raw" data is available should there be a reason to examine it.
6. The only times that were used, was the time from the recorded start of the last byte of a command to the recorded time of the first byte of the reply. This was done so that the transmission time of the bytes in the command and the transmission time of the bytes in the reply would not adversely affect the response time of a Spectra to commands. (D Protocol permits up a break of up to 249 ms between bytes in a command or reply. In all the testing done here there was effectively no inter-byte delay detected.)

2 Format of Data in the Appendices

1. **Typical response times:** This is a listing of the first few lines of a file that contains all response times. It is included to give an idea of what the reply times are. It is important to note that there is no obvious pattern to the times and they all tend to be similar. In other printouts the extremes, and modes are listed.

The two columns are: the record number and then the time in seconds.

This file is in CSV format.

2. **The minimum response times:** This is a listing of the results of a sort of all the "Typical response times" showing some of the smallest times.

The two columns are: the record number and then the time in seconds.

This file is in CSV format.

3. **The maximum response times:** This is a listing of the results of a sort of all the "Typical response times" showing some of the highest times.

The two columns are: the record number and then the time in seconds.

This file is in CSV format.

²There were some exceptions to this quantity of data collection.

4. **The most common response times:** This is a listing of sorting all of the “Typical response times”, deleting the record number column and then making a count of all the identical lines (FREQ). The results were then sorted and here there are several lines that showed the highest number of identical times. Typically about two thirds of the total number of times are represented by these few lines.

The two columns are: the count of the identical time and then the time in seconds.

This file is in CSV format.

5. **Typical raw data:** This is a sample of the output of the data capture software.

The six columns are:

- 5.1 The source ID, this will be either DCE or DTE. Depending on how the system is wired either one may be from the Spectra or from the Head End. Spectra replies are typically four bytes long and D Protocol commands are seven bytes long.
- 5.2 A sequential record number. (The very first column comes from the software that prints out all of these listings.)
- 5.3 The date that the record was recorded on.
- 5.4 The time that the record was recorded on.
- 5.5 An indicator if this was in the morning or the afternoon.
- 5.6 The data byte as a hexadecimal number.

6. **Typical data with fixed times and delta times:** This listing is the output of the MAKESECS program that takes the data in “Typical raw data” and massages it into a more useful format.

The seven columns are:

- 6.1 A message number within each source.
- 6.2 A sequential record number for records within each source.
- 6.3 The source ID indicator.
- 6.4 A sequential record number for all records.
- 6.5 A time in seconds from the start of the data capture.
- 6.6 A time in seconds from the preceding byte captured.
- 6.7 The data byte in hexadecimal.

7. **When the test was run:** This is information that was written by the data capture software to indicate some overall statistics on the communications capture.

8. **Run Statistics:** This was written by MAKESECS as a result of its processing of the capture file.

A CommaOut.L

```

1  %{
2  // "DTE",      33, 12,11/2006 6,:44:36,.753803, "AM","50"
3  // $Header: d:/Binder2/Timings/RCS/CommaOut.L,v 1.2 2007-02-09 07:03:42-08 Hamilton Exp Hamilton $
4  // CommaOut is used to dump unneeded commas from the output of FixRaw.
5  #include <stdio.h>
6  #include <stdlib.h>
7  #undef yywrap
8  %}

```

```

 9  %%
10  ",:"      fprintf(yyout,":");
11  ",/"      fprintf(yyout,"/");
12
13  " "      ;
14
15  .|\n      ECHO;
16  %%
17  // yywrap is for End Of File processing, 1 = done
18  int yywrap(void)
19  {
20      return 1;
21  }
22
23  main(int argc, char *argv[])
24  {
25      fprintf(stderr,"$Id: CommaOut.L,v 1.2 2007-02-09 07:03:42-08 Hamilton Exp Hamilton $\n");
26
27      // Do we have to deal with stdin/stdout?
28      // 3 = one for the program name and one each for input and output names
29      if (argc == 3) {
30          // Nope
31          // Get over the program, etc., name
32          argc--;
33          argv++;
34          yyin = fopen(argv[0],"r");
35          if (yyin == NULL) {
36              fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
37                      argv[0]);
38              exit(EXIT_FAILURE);
39          }
40
41          // Now get over the input file name and get to the output file name
42          argc--;
43          argv++;
44          yyout = fopen(argv[0],"w");
45          if (yyout == NULL) {
46              fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
47                      argv[0]);
48              exit(EXIT_FAILURE);
49          }
50      }
51      else {
52          // Woops we are using std... type io
53          yyin = stdin;
54          yyout = stdout;
55      }
56
57      yylex();
58      exit(EXIT_SUCCESS);
59  }
60
61  // $Log: CommaOut.L,v $
62  // Revision 1.2 2007-02-09 07:03:42-08 Hamilton
63  // Normal end of day data saving
64  //
65

```

B FixRaw.L

```

1  %{
2  // 123456789 123456789 123456789 123456789 123456789
3  // DTE      31 12/11/2006 6:44:36.715122 AM 6e
4  // DTE      32 12/11/2006 6:44:36.716164 AM 24
5  // DTE      33 12/11/2006 6:44:36.753803 AM 50
6  // 123456789 123456789 123456789 123456789 123456789
7  // $Header: d:/Binder2/Timings/RCS/FixRaw.L,v 1.2 2007-02-09 07:03:42-08 Hamilton Exp Hamilton $
8  // FixRaw is used to convert raw FTS files into something that ExSel can
9  // understand.
10 #include <stdio.h>
11 #include <stdlib.h>
12 #undef yywrap
13 %}
14 %%
15 "DCE"          |
16 "DTE"          fprintf(yyout,"%s\\",yytext);
17
18 ". "[0-9]{6}    fprintf(yyout,"%s",yytext);
19
20 "/"[0-9]{1,2}"/"[0-9]{4}" "    fprintf(yyout,"%s",yytext);
21 ":"[0-9]{1,2}":"[0-9]{1,2}    fprintf(yyout,"%s",yytext);
22
23 " "[0-9]{1,6}   fprintf(yyout,"%s",yytext);
24
25 "A"            fprintf(yyout,"\\A");
26 "P"            fprintf(yyout,"\\P");
27 "M "           fprintf(yyout,"M");
28 "[0-9a-f]{2}$  fprintf(yyout,"\\","\\s\\n",yytext);
29
30 \\n            ;
31
32 %%
33 //.\n          ECHO;
34 // yywrap is for End Of File processing, 1 = done
35 int yywrap(void)
36 {
37     return 1;
38 }
39
40 main(int argc, char *argv[])
41 {
42     fprintf(stderr,"$Id: FixRaw.L,v 1.2 2007-02-09 07:03:42-08 Hamilton Exp Hamilton $\n");
43
44     // Do we have to deal with stdin/stdout?
45     // 3 = one for the program name and one each for input and output names
46     if (argc == 3) {
47         // Nope
48         // Get over the program, etc., name
49         argc--;
50         argv++;
51         yyin = fopen(argv[0],"r");
52         if (yyin == NULL) {
53             fprintf(stderr,"\\aCould not open \\\"%s\\\" for reading, quitting",
54                     argv[0]);
55             exit(EXIT_FAILURE);
56         }
57
58         // Now get over the input file name and get to the output file name
59         argc--;
60         argv++;
61         yyout = fopen(argv[0],"w");
62         if (yyout == NULL) {
63             fprintf(stderr,"\\aCould not open \\\"%s\\\" for writing, quitting",
64                     argv[0]);
65             exit(EXIT_FAILURE);
66         }
67     }
68     else {
69         // Woops we are using std... type io

```

```
70         yyin = stdin;
71         yyout = stdout;
72     }
73
74     yylex();
75     exit(EXIT_SUCCESS);
76 }
77
78 // $Log: FixRaw.L,v $
79 // Revision 1.2  2007-02-09 07:03:42-08  Hamilton
80 // Normal end of day data saving
81 //
82
```

C GetTimes.L

```

1  %{
2  //      1,      1: DCE      1      0.000000  0.000000 ff
3  //      1,      2: DCE      2      0.001204  0.001204 01
4  //      1,      3: DCE      3      0.002239  0.001035 00
5  //      1,      4: DCE      4      0.003286  0.001047 0a
6  //      1,      5: DCE      5      0.004328  0.001042 19
7  //      1,      6: DCE      6      0.005376  0.001048 15
8  //      1,      7: DCE      7      0.006419  0.001043 39
9  //
10 //      1,      1: DTE      8      0.007760  0.001043 ff
11 //      1,      2: DTE      9      0.008802  0.001042 01
12 //      1,      3: DTE     10      0.009853  0.001051 00
13 //      1,      4: DTE     11      0.010878  0.001025 39
14 //123456789 123456789 123456789 123456789 123456789 123456789
15 //
16 // $Header: d:/Binder2/Timings/RCS/GetTimes.L,v 1.3 2007-02-09 07:03:42-08 Hamilton Exp Hamilton $
17 // GETTIMES is used to extract the start of a reply time delta from a
18 // MAKESECS processed capture file.
19 //
20 #include <stdio.h>
21 #include <stdlib.h>
22 #undef yywrap
23 %}
24 %%
25 .*: DTE ".*"ff"\n      ECHO;
26
27 .|\n      ;
28 %%
29 // yywrap is for End Of File processing, 1 = done
30 int yywrap(void)
31 {
32     return 1;
33 }
34
35 main(int argc, char *argv[])
36 {
37     fprintf(stderr,"$Id: GetTimes.L,v 1.3 2007-02-09 07:03:42-08 Hamilton Exp Hamilton $\n");
38
39     // Do we have to deal with stdin/stdout?
40     // 3 = one for the program name and one each for input and output names
41     if (argc == 3) {
42         // Nope
43         // Get over the program, etc., name
44         argc--;
45         argv++;
46         yyin = fopen(argv[0],"r");
47         if (yyin == NULL) {
48             fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
49                     argv[0]);
50             exit(EXIT_FAILURE);
51         }
52
53         // Now get over the input file name and get to the output file name
54         argc--;
55         argv++;
56         yyout = fopen(argv[0],"w");
57         if (yyout == NULL) {
58             fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
59                     argv[0]);
60             exit(EXIT_FAILURE);
61         }
62     }
63     else {
64         // Woops we are using std... type io
65         yyin = stdin;
66         yyout = stdout;
67     }
68
69     yylex();

```



```
70         exit(EXIT_SUCCESS);
71     }
72
73     // $Log: GetTimes.L,v $
74     // Revision 1.3  2007-02-09 07:03:42-08  Hamilton
75     // Normal end of day data saving
76     //
77
```

D GetTimP.L

```

1  %{
2  //      1,      1: DCE      1      0.000000  0.000000 a0
3  //      1,      2: DCE      2      0.001035  0.001035 01
4  //      1,      3: DCE      3      0.002077  0.001042 00
5  //      1,      4: DCE      4      0.003126  0.001049 0a
6  //      1,      5: DCE      5      0.004161  0.001035 07
7  //      1,      6: DCE      6      0.005210  0.001049 11
8  //      1,      7: DCE      7      0.006250  0.001040 af
9  //      1,      8: DCE      8      0.007300  0.001050 12
10 //
11 //      1,      1: DTE      9      0.008621  0.001050 a2
12 //123456789 123456789 123456789 123456789 123456789 123456789
13 //
14 // $Header: d:/Binder2/Timings/RCS/GetTimP.L,v 1.2 2007-02-09 07:03:42-08 Hamilton Exp Hamilton $
15 // GETTIMES is used to extract the start of a reply time delta from a
16 //      MAKESECS processed capture file of P Protocol data.
17 //
18 #include <stdio.h>
19 #include <stdlib.h>
20 #undef yywrap
21 %}
22 %%
23 .*: DTE ".*"a2"\n      ECHO;
24
25 .|\n      ;
26 %%
27 // yywrap is for End Of File processing, 1 = done
28 int yywrap(void)
29 {
30     return 1;
31 }
32
33 main(int argc, char *argv[])
34 {
35     fprintf(stderr,"$Id: GetTimP.L,v 1.2 2007-02-09 07:03:42-08 Hamilton Exp Hamilton $\n");
36
37     // Do we have to deal with stdin/stdout?
38     // 3 = one for the program name and one each for input and output names
39     if (argc == 3) {
40         // Nope
41         // Get over the program, etc., name
42         argc--;
43         argv++;
44         yyin = fopen(argv[0],"r");
45         if (yyin == NULL) {
46             fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
47                     argv[0]);
48             exit(EXIT_FAILURE);
49         }
50
51         // Now get over the input file name and get to the output file name
52         argc--;
53         argv++;
54         yyout = fopen(argv[0],"w");
55         if (yyout == NULL) {
56             fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
57                     argv[0]);
58             exit(EXIT_FAILURE);
59         }
60     }
61     else {
62         // Woops we are using std... type io
63         yyin = stdin;
64         yyout = stdout;
65     }
66
67     yylex();
68     exit(EXIT_SUCCESS);
69 }

```

```
70
71 // $Log: GetTimP.L,v $
72 // Revision 1.2  2007-02-09 07:03:42-08  Hamilton
73 // Normal end of day data saving
74 //
75
```

E HGetTime.L

```

1  %{
2  //      1,      1: DCE      1      0.000000  0.000000 ff
3  //      1,      2: DCE      2      0.001204  0.001204 01
4  //      1,      3: DCE      3      0.002239  0.001035 00
5  //      1,      4: DCE      4      0.003286  0.001047 0a
6  //      1,      5: DCE      5      0.004328  0.001042 19
7  //      1,      6: DCE      6      0.005376  0.001048 15
8  //      1,      7: DCE      7      0.006419  0.001043 39
9  //
10 //      1,      1: DTE      8      0.007760  0.001043 ff
11 //      1,      2: DTE      9      0.008802  0.001042 01
12 //      1,      3: DTE     10      0.009853  0.001051 00
13 //      1,      4: DTE     11      0.010878  0.001025 39
14 //123456789 123456789 123456789 123456789 123456789 123456789
15 //
16 // $Header: d:/Binder2/Timings/RCS/HGetTime.L,v 1.2 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $
17 // HETTIME is used to extract the start of a reply time delta from a
18 // MAKESECS processed capture file. This is designed to run on a
19 // Hernis Protocol capture file.
20 //
21 #include <stdio.h>
22 #include <stdlib.h>
23 #undef yywrap
24 %}
25 %%
26 .*: DTE ".*"50"\n      ECHO;
27
28 .|\n      ;
29 %%
30 // yywrap is for End Of File processing, 1 = done
31 int yywrap(void)
32 {
33     return 1;
34 }
35
36 main(int argc, char *argv[])
37 {
38     fprintf(stderr,"$Id: HGetTime.L,v 1.2 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $\n");
39
40     // Do we have to deal with stdin/stdout?
41     // 3 = one for the program name and one each for input and output names
42     if (argc == 3) {
43         // Nope
44         // Get over the program, etc., name
45         argc--;
46         argv++;
47         yyin = fopen(argv[0],"r");
48         if (yyin == NULL) {
49             fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
50                 argv[0]);
51             exit(EXIT_FAILURE);
52         }
53
54         // Now get over the input file name and get to the output file name
55         argc--;
56         argv++;
57         yyout = fopen(argv[0],"w");
58         if (yyout == NULL) {
59             fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
60                 argv[0]);
61             exit(EXIT_FAILURE);
62         }
63     }
64     else {
65         // Woops we are using std... type io
66         yyin = stdin;
67         yyout = stdout;
68     }
69

```

```
70         yylex();
71         exit(EXIT_SUCCESS);
72     }
73
74     // $Log: HGetTime.L,v $
75     // Revision 1.2  2007-02-09 07:03:43-08  Hamilton
76     // Normal end of day data saving
77     //
78
```

F HJusTime.L

```

1  %{
2  //123456789 123456789 123456789 123456789 123456789 123456789
3  //      1,      1: DTE      15      0.160269      0.001047 ff
4  //      2,      5: DTE      26      0.307559      0.001043 ff
5  //      2,      9: DTE      30      0.311722      0.001041 ff
6  //      4,     24: DTE      59      0.454575      0.001042 ff
7  //      5,     28: DTE      70      0.558371      0.001041 ff
8  //      6,     32: DTE      81      0.672061      0.000881 ff
9  //      7,     36: DTE     106      0.994849      0.001039 ff
10 //      8,     40: DTE     117      1.107305      0.001034 ff
11 //      9,     44: DTE     142      1.436274      0.001042 ff
12 //     10,     48: DTE     153      1.542113      0.001048 ff
13 //123456789 123456789 123456789 123456789 123456789 123456789
14 //
15 // $Header: d:/Binder2/Timings/RCS/HJusTime.L,v 1.2 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $
16 // JustTime is used to remove everything but the record number and the
17 //      delta time from capture files processed by MAKESECS and GetTimes.
18 //      This is for use with Hernis Protocol captures.
19 #include <stdio.h>
20 #include <stdlib.h>
21 #undef yywrap
22 %}
23 %%
24 ~" *[0-9]*", " fprintf(yyout,"%s",yytext);
25 .{10}" 50"$ {fprintf(yyout,"%c%c%c%c%c%c%c%c%c%c\n",
26                      yytext[0],yytext[1],yytext[2],
27                      yytext[3],yytext[4],yytext[5],
28                      yytext[6],yytext[7],yytext[8],yytext[9]);
29                      }
30
31 .|\n      ;
32
33 %%
34 // yywrap is for End Of File processing, 1 = done
35 int yywrap(void)
36 {
37     return 1;
38 }
39
40 main(int argc, char *argv[])
41 {
42     fprintf(stderr,"$Id: HJusTime.L,v 1.2 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $\n");
43
44     // Do we have to deal with stdin/stdout?
45     // 3 = one for the program name and one each for input and output names
46     if (argc == 3) {
47         // Nope
48         // Get over the program, etc., name
49         argc--;
50         argv++;
51         yyin = fopen(argv[0],"r");
52         if (yyin == NULL) {
53             fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
54                     argv[0]);
55             exit(EXIT_FAILURE);
56         }
57
58         // Now get over the input file name and get to the output file name
59         argc--;
60         argv++;
61         yyout = fopen(argv[0],"w");
62         if (yyout == NULL) {
63             fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
64                     argv[0]);
65             exit(EXIT_FAILURE);
66         }
67     }
68     else {
69         // Woops we are using std... type io

```

```
70         yyin = stdin;
71         yyout = stdout;
72     }
73
74     yylex();
75     exit(EXIT_SUCCESS);
76 }
77
78 // $Log: HJusTime.L,v $
79 // Revision 1.2  2007-02-09 07:03:43-08  Hamilton
80 // Normal end of day data saving
81 //
82
```

G JustTime.L

```

1  %{
2  //123456789 123456789 123456789 123456789 123456789 123456789
3  //      1,      1: DTE      15      0.160269      0.001047 ff
4  //      2,      5: DTE      26      0.307559      0.001043 ff
5  //      2,      9: DTE      30      0.311722      0.001041 ff
6  //      4,     24: DTE      59      0.454575      0.001042 ff
7  //      5,     28: DTE      70      0.558371      0.001041 ff
8  //      6,     32: DTE      81      0.672061      0.000881 ff
9  //      7,     36: DTE     106      0.994849      0.001039 ff
10 //      8,     40: DTE     117      1.107305      0.001034 ff
11 //      9,     44: DTE     142      1.436274      0.001042 ff
12 //     10,     48: DTE     153      1.542113      0.001048 ff
13 //123456789 123456789 123456789 123456789 123456789 123456789
14 //
15 // $Header: d:/Binder2/Timings/RCS/JustTime.L,v 1.3 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $
16 // JustTime is used to remove everything but the record number and the
17 //      delta time from capture files processed by MAKESECS and GetTimes.
18 #include <stdio.h>
19 #include <stdlib.h>
20 #undef yywrap
21 %}
22 %%
23 ~" "[0-9]*", " fprintf(yyout,"%s",yytext);
24 .{10}" ff"$      {fprintf(yyout,"%c%c%c%c%c%c%c%c%c%c\n",
25                      yytext[0],yytext[1],yytext[2],
26                      yytext[3],yytext[4],yytext[5],
27                      yytext[6],yytext[7],yytext[8],yytext[9]);
28                      }
29
30 .|\n      ;
31
32 %%
33 // yywrap is for End Of File processing, 1 = done
34 int yywrap(void)
35 {
36     return 1;
37 }
38
39 main(int argc, char *argv[])
40 {
41     fprintf(stderr,"$Id: JustTime.L,v 1.3 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $\n");
42
43     // Do we have to deal with stdin/stdout?
44     // 3 = one for the program name and one each for input and output names
45     if (argc == 3) {
46         // Nope
47         // Get over the program, etc., name
48         argc--;
49         argv++;
50         yyin = fopen(argv[0],"r");
51         if (yyin == NULL) {
52             fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
53                     argv[0]);
54             exit(EXIT_FAILURE);
55         }
56
57         // Now get over the input file name and get to the output file name
58         argc--;
59         argv++;
60         yyout = fopen(argv[0],"w");
61         if (yyout == NULL) {
62             fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
63                     argv[0]);
64             exit(EXIT_FAILURE);
65         }
66     }
67     else {
68         // Woops we are using std... type io
69         yyin = stdin;

```



```
70         yyout = stdout;
71     }
72
73     yylex();
74     exit(EXIT_SUCCESS);
75 }
76
77 // $Log: JustTime.L,v $
78 // Revision 1.3  2007-02-09 07:03:43-08  Hamilton
79 // Normal end of day data saving
80 //
81
```

H JustTimP.L

```

1  %{
2  //123456789 123456789 123456789 123456789 123456789 123456789
3  //      1,      1: DTE      9      0.008621  0.001050 a2
4  //      2,      2: DTE     18      0.120798  0.001374 a2
5  //      3,      3: DTE     27      0.227408  0.001042 a2
6  //      4,      4: DTE     36      0.339792  0.001459 a2
7  //123456789 123456789 123456789 123456789 123456789 123456789
8  //
9  // $Header: d:/Binder2/Timings/RCS/JustTimP.L,v 1.2 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $
10 // JustTimP is used to remove everything but the record number and the
11 //      delta time from capture files processed by MAKESECS and GetTimP
12 //      for P Protocol data.
13 #include <stdio.h>
14 #include <stdlib.h>
15 #undef yywrap
16 %}
17 %%
18 ~" "[0-9]*", " fprintf(yyout,"%s",yytext);
19 .{10}" a2"$ {fprintf(yyout,"%c%c%c%c%c%c%c%c%c%c\n",
20                      yytext[0],yytext[1],yytext[2],
21                      yytext[3],yytext[4],yytext[5],
22                      yytext[6],yytext[7],yytext[8],yytext[9]);
23                      }
24
25 .|\n      ;
26
27 %%
28 // yywrap is for End Of File processing, 1 = done
29 int yywrap(void)
30 {
31     return 1;
32 }
33
34 main(int argc, char *argv[])
35 {
36     fprintf(stderr,"$Id: JustTimP.L,v 1.2 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $\n");
37
38     // Do we have to deal with stdin/stdout?
39     // 3 = one for the program name and one each for input and output names
40     if (argc == 3) {
41         // Nope
42         // Get over the program, etc., name
43         argc--;
44         argv++;
45         yyin = fopen(argv[0],"r");
46         if (yyin == NULL) {
47             fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
48                     argv[0]);
49             exit(EXIT_FAILURE);
50         }
51
52         // Now get over the input file name and get to the output file name
53         argc--;
54         argv++;
55         yyout = fopen(argv[0],"w");
56         if (yyout == NULL) {
57             fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
58                     argv[0]);
59             exit(EXIT_FAILURE);
60         }
61     }
62     else {
63         // Woops we are using std... type io
64         yyin = stdin;
65         yyout = stdout;
66     }
67
68     yylex();
69     exit(EXIT_SUCCESS);

```

```
70 }  
71 // $Log: JustTimP.L,v $  
72 // Revision 1.2  2007-02-09 07:03:43-08  Hamilton  
73 // Normal end of day data saving  
74 //  
75
```

I MakeD3.L

```

1  %{
2  // 123456789 123456789 123456789 123456789 123456789 123456789
3  //      1,      1: DCE      1      0.000000  0.000000 ff
4  //      1,      2: DCE      2      0.001031  0.001031 01
5  //      1,      3: DCE      3      0.002078  0.001047 00
6  //      1,      4: DCE      4      0.003121  0.001043 08
7  //      1,      5: DCE      5      0.004163  0.001042 00
8  //      1,      6: DCE      6      0.005205  0.001042 09
9  //      1,      7: DCE      7      0.006298  0.001093 12
10 //
11 //      1,      1: DTE      8      0.007929  0.001093 ff
12 //      1,      2: DTE      9      0.008965  0.001036 01
13 //      1,      3: DTE     10      0.010005  0.001040 00
14 //      1,      4: DTE     11      0.011047  0.001042 12
15 // 123456789 123456789 123456789 123456789 123456789 123456789
16 //
17 // $Header: d:/Binder2/Timings/RCS/MakeD3.L,v 1.4 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $
18 // MakeD is used to process D Protocol messages that have been previously
19 //      processed by MAKESECS. The result is a more intellegent version
20 //      of the data.
21 #include <stdio.h>
22 #include <stdlib.h>
23 #include "dproto.h"
24
25 #undef yywrap
26
27 char string[100];
28 char bytess[100];
29 char savestart[1000];
30 char hexbyte[10];
31
32 void DecodeIt(void);
33 unsigned char MakeHex(char byte);
34 void addtwo(void);
35
36 int dcecount = 0;
37 int dtcount = 0;
38 int doingdce = 0;
39 int doingdte = 0;
40
41 %}
42 %%
43 .*: DCE ".*\n {strncpy(savestart,yytext,55);
44                hexbyte[0] = ' ';
45                hexbyte[1] = yytext[53];
46                hexbyte[2] = yytext[54];
47                hexbyte[3] = '\0';
48                dcecount++;
49                doingdte = 0;
50                if (strcmp(" ff",hexbyte) == 0)
51                {
52                    if (doingdce == 1)
53                        fprintf(yyout,"\n");
54                    dcecount = 0;
55                    fprintf(yyout,"%s",savestart);
56                    strcpy(string,hexbyte);
57                }
58                else
59                {
60                    if (doingdte == 1)
61                    {
62                        fprintf(yyout,"%s",savestart);
63                    }
64                    else
65                    {
66                        fprintf(yyout,"%s",hexbyte);
67                        strcat(string,hexbyte);
68                        if (dcecount >= 6)
69                        {

```

```

70             DecodeIt();
71             doingdce = 1;
72             BEGIN 0;
73         }
74     }
75 }
76 }
77
78 .*: DTE".*\n {strncpy(savestart,yytext,55);
79             hexbyte[0] = ' ';
80             hexbyte[1] = yytext[53];
81             hexbyte[2] = yytext[54];
82             hexbyte[3] = '\0';
83             dtecount++;
84             doingdce = 0;
85             if (strcmp(" ff",hexbyte) == 0)
86             {
87                 if (doingdte == 1)
88                     fprintf(yyout,"\n");
89                 dtecount = 0;
90                 fprintf(yyout,"%s",savestart);
91                 strcpy(string,hexbyte);
92             }
93             else
94             {
95                 if (doingdce == 1)
96                 {
97                     fprintf(yyout,"%s",savestart);
98                 }
99                 else
100                 {
101                     fprintf(yyout,"%s",hexbyte);
102                     strcat(string,hexbyte);
103                     if (dtecount >= 3)
104                     {
105                         doingdte = 1;
106                         fprintf(yyout,"\n");
107                         BEGIN 0;
108                     }
109                 }
110             }
111         }
112
113
114 \n          ECHO;
115 .          ECHO;
116 %%
117 // This routine is used to add two bytes from the input line onto the
118 // main string it also echos the bytes to the output file.
119 void addtwo(void)
120 {
121     sprintf(bytess,"%c%c ",yytext[46],yytext[47]);
122     strcat(string,bytess);
123     fprintf(yyout,"%s",bytess);
124 }
125
126
127 // This routine converts an AsciiHex byte into a Hex byte.
128 //
129 unsigned char MakeHex(char byte)
130 {
131     unsigned char temp;
132
133     temp = byte - 0x30;          // 0 = 0x30, 9 = 0x39, a = 0x61, f = 0x66
134     if (temp >= 0x30)           // temp = 0x00 --> 0x36
135         temp -= (0x31 - 0x0a);  // temp = 0x00 --> 0x0f
136     return temp;
137 }
138
139
140 // Used to interpret a D Protocol command
141 //

```

```

142 // Calling sequence:
143 //   string has a D Protocol command in AsciiHex.
144 //   ff aa cc cc dd dd cc
145 //   0123456789 123456789
146 //
147 void DecodeIt(void)
148 {
149   int  thisbyte;
150   char decodestring[1000];
151   char temp[100];
152   char errorstring[100];
153
154   // Sync
155   strcpy(decodestring, ":");
156
157   // Address
158   //   temp[0] = string[4];
159   //   temp[1] = string[5];
160   temp[2] = '\0';
161   //   strcat(decodestring,temp);
162   //   strcat(decodestring," ");
163
164   // See if this is a value coded command or is a bit encoded command
165   thisbyte = MakeHex(string[10]) * 16;
166   thisbyte += MakeHex(string[11]);
167
168   // Check for a stop all command
169   if (thisbyte == 0x00)
170   {
171     strcat(decodestring,"Stop All Motion  ");
172   }
173
174   // All odd command byte 2s are value encoded commands
175   else if (thisbyte & 0x01)
176   {
177     switch(thisbyte)
178     {
179       case EC_SET_PRESET:      strcat(decodestring,"SET_PRESET      "); break;
180       case EC_CLEAR_PRESET:   strcat(decodestring,"CLEAR_PRESET   "); break;
181       case EC_MOVE_PRESET:    strcat(decodestring,"MOVE_PRESET    "); break;
182       case EC_SET_AUX:        strcat(decodestring,"SET_AUX        "); break;
183       case EC_CLEAR_AUX:      strcat(decodestring,"CLEAR_AUX      "); break;
184       case EC_DUMMY_1:        strcat(decodestring,"DUMMY_1        "); break;
185       case EC_RESET:          strcat(decodestring,"RESET          "); break;
186       case EC_ZONE_START:     strcat(decodestring,"ZONE_START     "); break;
187       case EC_ZONE_END:       strcat(decodestring,"ZONE_END       "); break;
188       case EC_WRITE_CHAR:     strcat(decodestring,"WRITE_CHAR     "); break;
189       case EC_CLEAR_SCREEN:   strcat(decodestring,"CLEAR_SCREEN   "); break;
190       case EC_ALARM_ACK:      strcat(decodestring,"ALARM_ACK      "); break;
191       case EC_ZONE_ON:        strcat(decodestring,"ZONE_ON        "); break;
192       case EC_ZONE_OFF:       strcat(decodestring,"ZONE_OFF       "); break;
193       case EC_START_RECORD:    strcat(decodestring,"START_RECORD   "); break;
194       case EC_END_RECORD:     strcat(decodestring,"END_RECORD     "); break;
195       case EC_START_PLAY:     strcat(decodestring,"START_PLAY     "); break;
196       case EC_ZOOM_SPEED:      strcat(decodestring,"ZOOM_SPEED     "); break;
197       case EC_FOCUS_SPEED:    strcat(decodestring,"FOCUS_SPEED    "); break;
198       case EC_CAMERA_RESET:   strcat(decodestring,"CAMERA_RESET   "); break;
199       case EC_AUTO_FOCUS:     strcat(decodestring,"AUTO_FOCUS     "); break;
200       case EC_AUTO_IRIS:      strcat(decodestring,"AUTO_IRIS      "); break;
201       case EC_AGC:            strcat(decodestring,"AGC            "); break;
202       case EC_BLC:            strcat(decodestring,"BLC            "); break;
203       case EC_AWB:            strcat(decodestring,"AWB            "); break;
204       case EC_DEVICE_PHASE:    strcat(decodestring,"DEVICE_PHASE    "); break;
205       case EC_SHUTTER_SPEED:   strcat(decodestring,"SHUTTER_SPEED   "); break;
206       case EC_ADJUST_PHASE:    strcat(decodestring,"ADJUST_PHASE    "); break;
207       case EC_ADJUST_RB_WB:    strcat(decodestring,"ADJUST_RB_WB   "); break;
208       case EC_ADJUST_MG_WB:    strcat(decodestring,"ADJUST_MG_WB   "); break;
209       case EC_ADJUST_GAIN:     strcat(decodestring,"ADJUST_GAIN     "); break;
210       case EC_ADJUST_AI_LEVEL: strcat(decodestring,"ADJUST_AI_LEVEL "); break;
211       case EC_ADJUST_AI_PEAK:  strcat(decodestring,"ADJUST_AI_PEAK "); break;
212       case EC_QUERY:          strcat(decodestring,"QUERY          "); break;
213       case EC_PRESET_SCAN:     strcat(decodestring,"PRESET_SCAN     "); break;

```

```

214         case EC_SET_ZERO:          strcat(decodestring,"SET_ZERO          "); break;
215         case EC_SET_PAN:            strcat(decodestring,"SET_PAN            "); break;
216         case EC_SET_TILT:           strcat(decodestring,"SET_TILT           "); break;
217         case EC_SET_ZOOM:           strcat(decodestring,"SET_ZOOM           "); break;
218         case EC_QUERY_PAN:          strcat(decodestring,"QUERY_PAN          "); break;
219         case EC_QUERY_TILT:         strcat(decodestring,"QUERY_TILT         "); break;
220         case EC_QUERY_ZOOM:         strcat(decodestring,"QUERY_ZOOM         "); break;
221         case EC_DOWNLOAD:           strcat(decodestring,"DOWNLOAD           "); break;
222         case EC_PAN_RESP:           strcat(decodestring,"PAN_RESP           "); break;
223         case EC_TILT_RESP:          strcat(decodestring,"TILT_RESP          "); break;
224         case EC_ZOOM_RESP:          strcat(decodestring,"ZOOM_RESP          "); break;
225         case EC_SET_MAG:            strcat(decodestring,"SET_MAG            "); break;
226         case EC_QUERY_MAG:          strcat(decodestring,"QUERY_MAG          "); break;
227         case EC_MAG_RESP:           strcat(decodestring,"MAG_RESP           "); break;
228         case EC_ECHO_MODE:          strcat(decodestring,"ECHO_MODE          "); break;
229         case EC_SET_BAUD:           strcat(decodestring,"SET_BAUD           "); break;
230         case EC_START_DOWNLOAD:     strcat(decodestring,"START_DOWNLOAD     "); break;
231         case EC_QUERY_DEV_TYPE:     strcat(decodestring,"QUERY_DEV_TYPE     "); break;
232         case EC_QUERY_DEV_TYPE_RESP: strcat(decodestring,"QUERY_DEV_TYPE_RESP"); break;
233         case EC_QUERY_DIAG_INFO:    strcat(decodestring,"QUERY_DIAG_INFO    "); break;
234         case EC_QUERY_DIAG_INFO_RESP: strcat(decodestring,"QUERY_DIAG_INFO_RESP"); break;
235         case EC_VERSION_INFO:       strcat(decodestring,"VERSION_INFO     "); break;
236         case EC_EVEREST:            strcat(decodestring,"EVEREST            "); break;
237         default:
238             strcat(decodestring,"error          ");
239             break;
240     }
241 }
242 else
243 // All even command byte 2s are bit encoded commands
244 {
245     // Command 1
246     thisbyte = MakeHex(string[7]) * 16;
247     thisbyte += MakeHex(string[8]);
248
249     if (thisbyte & 0x80) strcat(decodestring,"S"); // Sense
250     else                 strcat(decodestring,".");
251
252     if (thisbyte & 0x40) strcat(decodestring,"r"); // reserved
253     else                 strcat(decodestring,".");
254
255     if (thisbyte & 0x20) strcat(decodestring,"r"); // reserved
256     else                 strcat(decodestring,".");
257
258     if (thisbyte & 0x10) strcat(decodestring,"A"); // Auto/manual
259     else                 strcat(decodestring,".");
260
261     if (thisbyte & 0x08) strcat(decodestring,"F"); // camera on/oFf
262     else                 strcat(decodestring,".");
263
264     if (thisbyte & 0x04) strcat(decodestring,"C"); // Close
265     else                 strcat(decodestring,".");
266
267     if (thisbyte & 0x02) strcat(decodestring,"O"); // Open
268     else                 strcat(decodestring,".");
269
270     if (thisbyte & 0x01) strcat(decodestring,"N"); // Near
271     else                 strcat(decodestring,".");
272
273     strcat(decodestring," ");
274
275     // Command 2
276     thisbyte = MakeHex(string[10]) * 16;
277     thisbyte += MakeHex(string[11]);
278     if (thisbyte & 0x80) strcat(decodestring,"F"); // Far
279     else                 strcat(decodestring,".");
280
281     if (thisbyte & 0x40) strcat(decodestring,"W"); // Wide
282     else                 strcat(decodestring,".");
283
284     if (thisbyte & 0x20) strcat(decodestring,"T"); // Tele
285     else                 strcat(decodestring,".");

```

```

286
287     if (thisbyte & 0x10) strcat(decodestring,"D"); // Down
288     else                    strcat(decodestring,".");
289
290     if (thisbyte & 0x08) strcat(decodestring,"U"); // Up
291     else                    strcat(decodestring,".");
292
293     if (thisbyte & 0x04) strcat(decodestring,"L"); // Left
294     else                    strcat(decodestring,".");
295
296     if (thisbyte & 0x02) strcat(decodestring,"R"); // Right
297     else                    strcat(decodestring,".");
298
299     if (thisbyte & 0x01) strcat(decodestring,"E"); // Error always "0"
300     else                    strcat(decodestring,".");
301
302     strcat(decodestring," ");
303 }
304
305 // Data 1, pan speed
306 temp[0] = string[13];
307 temp[1] = string[14];
308 strcat(decodestring,temp);
309 strcat(decodestring," ");
310
311 // Data 2, tilt speed
312 temp[0] = string[16];
313 temp[1] = string[17];
314 strcat(decodestring,temp);
315 strcat(decodestring," ");
316
317 // // Checksum
318 // temp[0] = string[19];
319 // temp[1] = string[20];
320 // strcat(decodestring,temp);
321
322 fprintf(yyout,"%s",decodestring);
323 }
324
325 // yywrap is for End Of File processing, 1 = done
326 int yywrap(void)
327 {
328     return 1;
329 }
330
331 main(int argc, char *argv[])
332 {
333     fprintf(stderr,"$Id: MakeD3.L,v 1.4 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $\n");
334
335     // Do we have to deal with stdin/stdout?
336     // 3 = one for the program name and one each for input and output names
337     if (argc == 3) {
338         // Nope
339         // Get over the program, etc., name
340         argc--;
341         argv++;
342         yyin = fopen(argv[0],"r");
343         if (yyin == NULL) {
344             fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
345                     argv[0]);
346             exit(EXIT_FAILURE);
347         }
348
349         // Now get over the input file name and get to the output file name
350         argc--;
351         argv++;
352         yyout = fopen(argv[0],"w");
353         if (yyout == NULL) {
354             fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
355                     argv[0]);
356             exit(EXIT_FAILURE);
357         }

```



```
358     }
359     else {
360         // Woops we are using std... type io
361         yyin = stdin;
362         yyout = stdout;
363     }
364
365     yylex();
366     exit(EXIT_SUCCESS);
367 }
368
369 // $Log: MakeD3.L,v $
370 // Revision 1.4  2007-02-09 07:03:43-08  Hamilton
371 // Normal end of day data saving
372 //
373
```

J NoColon.L

```

1  %{
2  // $Header: d:/Binder2/Timings/RCS/NoColon.L,v 1.2 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $
3  // NOCOLON is used to change all colons ":" into commas ",", ".
4  #include <stdio.h>
5  #include <stdlib.h>
6  #undef yywrap
7  %}
8  %%
9  ":"      fprintf(yyout, ",");
10
11  .|\n     ECHO;
12  %%
13  // yywrap is for End Of File processing, 1 = done
14  int yywrap(void)
15  {
16      return 1;
17  }
18
19  main(int argc, char *argv[])
20  {
21      fprintf(stderr, "$Id: NoColon.L,v 1.2 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $\n");
22
23      // Do we have to deal with stdin/stdout?
24      // 3 = one for the program name and one each for input and output names
25      if (argc == 3) {
26          // Nope
27          // Get over the program, etc., name
28          argc--;
29          argv++;
30          yyin = fopen(argv[0], "r");
31          if (yyin == NULL) {
32              fprintf(stderr, "\aCould not open \"%s\" for reading, quitting",
33                      argv[0]);
34              exit(EXIT_FAILURE);
35          }
36
37          // Now get over the input file name and get to the output file name
38          argc--;
39          argv++;
40          yyout = fopen(argv[0], "w");
41          if (yyout == NULL) {
42              fprintf(stderr, "\aCould not open \"%s\" for writing, quitting",
43                      argv[0]);
44              exit(EXIT_FAILURE);
45          }
46      }
47      else {
48          // Woops we are using std... type io
49          yyin = stdin;
50          yyout = stdout;
51      }
52
53      yylex();
54      exit(EXIT_SUCCESS);
55  }
56
57
58  // $Log: NoColon.L,v $
59  // Revision 1.2  2007-02-09 07:03:43-08  Hamilton
60  // Normal end of day data saving
61  //
62

```

K OnlyTime.L

```

1  %{
2  //123456789 123456789
3  //      1,   0.001093
4  //      2,   0.001037
5  //      3,   0.000882
6  //      4,   0.001041
7  //123456789 123456789
8  // $Header: d:/Binder2/Timings/RCS/OnlyTime.L,v 1.3 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $
9  // OnlyTime is used to extract just the time data from files processed by
10 //      JustTime.
11 #include <stdio.h>
12 #include <stdlib.h>
13 #undef yywrap
14 %}
15 %%
16 .{10}$      fprintf(yyout,"%s%\n",yytext);
17
18 .|\n      ;
19 %%
20 // yywrap is for End Of File processing, 1 = done
21 int yywrap(void)
22 {
23     return 1;
24 }
25
26 main(int argc, char *argv[])
27 {
28     fprintf(stderr,"$Id: OnlyTime.L,v 1.3 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $\n");
29
30     // Do we have to deal with stdin/stdout?
31     // 3 = one for the program name and one each for input and output names
32     if (argc == 3) {
33         // Nope
34         // Get over the program, etc., name
35         argc--;
36         argv++;
37         yyin = fopen(argv[0],"r");
38         if (yyin == NULL) {
39             fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
40                     argv[0]);
41             exit(EXIT_FAILURE);
42         }
43
44         // Now get over the input file name and get to the output file name
45         argc--;
46         argv++;
47         yyout = fopen(argv[0],"w");
48         if (yyout == NULL) {
49             fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
50                     argv[0]);
51             exit(EXIT_FAILURE);
52         }
53     }
54     else {
55         // Woops we are using std... type io
56         yyin = stdin;
57         yyout = stdout;
58     }
59
60     yylex();
61     exit(EXIT_SUCCESS);
62 }
63
64 // $Log: OnlyTime.L,v $
65 // Revision 1.3 2007-02-09 07:03:43-08 Hamilton
66 // Normal end of day data saving
67 //
68

```

L Procit.bat

```

1  @echo off
2  rem $Header: d:/Binder2/Timings/RCS/ProcIt.bat,v 1.13 2007-02-12 11:28:49-08 Hamilton Exp Hamilton $
3
4  rem Convert the input file to a useful format and get just the reply times
5  makesecs Original\%1.txt %1.sec
6  gettimes %1.sec %1.gt
7  justtime %1.gt %1.tim
8
9  rem Count the number of times in the file
10 lc %1.tim | tail -1 > %1.lin
11
12 rem Save the "typical times"
13 head -25 %1.tim > %1.typ
14
15 rem Now sort um by reply times
16 qsort %1.tim %1.srt /8:15 /1:7
17
18 rem Get the shortest times
19 head -12 %1.srt > %1.min
20
21 rem I don't know why this has to be done twice, but it works
22 head -12 %1.srt > %1.min
23
24 rem Get the longest times
25 tail -12 %1.srt > %1.max
26
27 rem Strip out the replys only
28 onlytime <%1.srt | efreq > %1.frq
29
30 rem Now get the most common reply times
31 qsort %1.frq %1.tmp
32 nocolon %1.tmp %1.ncl
33 tail -12 %1.ncl > %1.mde
34 del %1.tmp
35 del %1.ncl
36
37 rem Now generate a gnuplot control file for durations vs. reply number
38 echo set xlabel "Reply Number"> %1.gp
39 echo set ylabel "Duration" >> %1.gp
40 echo set term latex >> %1.gp
41 echo set output "%1.inc" >> %1.gp
42 echo set timestamp >> %1.gp
43 echo set grid >> %1.gp
44 echo set key left top >> %1.gp
45 echo plot "%1.tim" using 1:2 title "%1.tim response times for pan/tilt commands" with lines >> %1.gp
46 gnuplot %1.gp
47
48 rem Now make another gnuplot control file for durations vs. number of that duration
49 echo set ylabel "Number" > %1d.gp
50 echo set xlabel "Duration" >> %1d.gp
51 echo set term latex >> %1d.gp
52 echo set output "%1d.inc" >> %1d.gp
53 echo set timestamp >> %1d.gp
54 echo set grid >> %1d.gp
55 echo set key left top >> %1d.gp
56 echo plot "%1.frq" using 2:1 title "%1.frq response times for pan/tilt commands" with lines >> %1d.gp
57 gnuplot %1d.gp
58
59 rem Find out when the test was run
60 head -5 %1.sec | tail -2 > %1.whn
61
62 rem List out statistics about the run
63 tail -8 %1.sec | head -7 > %1.dat
64
65 rem Get typical raw data
66 head -38 Original\%1.txt | tail -33 >%1.raw
67
68 rem Get typical output of MAKESECS
69 head -42 %1.sec | tail -36 >%1.cok

```

```
70
71 dir %1*.*
72
73 mv %1.cok SavedFiles
74 mv %1.dat SavedFiles
75 mv %1.frq SavedFiles
76 mv %1.gp SavedFiles
77 mv %1d.gp SavedFiles
78 mv %1.gt SavedFiles
79 mv %1.inc SavedFiles
80 mv %1d.inc SavedFiles
81 mv %1.lin SavedFiles
82 mv %1.max SavedFiles
83 mv %1.mde SavedFiles
84 mv %1.min SavedFiles
85 mv %1.raw SavedFiles
86 mv %1.sec SavedFiles
87 mv %1.srt SavedFiles
88 mv %1.tim SavedFiles
89 mv %1.typ SavedFiles
90 mv %1.whn SavedFiles
91
92 echo Make sure that all directory entries have something in them.
93 pause
94
```

M ProcitP.bat

```

1  @echo off
2  rem $Header: d:/Binder2/Timings/RCS/ProcitP.bat,v 1.4 2007-02-09 09:04:45-08 Hamilton Exp Hamilton $
3
4  rem Convert the input file to a useful format and get just the reply times
5  makesecs Original\%1.txt %1.sec
6  gettimp %1.sec %1.gt
7  justtimp %1.gt %1.tim
8
9  rem Count the number of times in the file
10 lc %1.tim | tail -1 > %1.lin
11
12 rem Save the "typical times"
13 head -25 %1.tim > %1.typ
14
15 rem Now sort um by reply times
16 qsort %1.tim %1.srt /8:15 /1:7
17
18 rem Get the shortest times
19 head -10 %1.srt > %1.min
20
21 rem I don't know why this has to be done twice, but it works
22 head -10 %1.srt > %1.min
23
24 rem Get the longest times
25 tail -10 %1.srt > %1.max
26
27 rem Strip out the replys only
28 onlytime <%1.srt | efreq > %1.frq
29
30 rem Now get the most common reply times
31 qsort %1.frq %1.tmp
32 nocolon %1.tmp %1.ncl
33 tail -10 %1.ncl > %1.mde
34 del %1.tmp
35 del %1.ncl
36
37 rem Now generate a gnuplot control file for durations vs. reply number
38 echo set xlabel "Reply Number"> %1.gp
39 echo set ylabel "Duration" >> %1.gp
40 echo set term latex >> %1.gp
41 echo set output "%1.inc" >> %1.gp
42 echo set timestamp >> %1.gp
43 echo set grid >> %1.gp
44 echo set key left top >> %1.gp
45 echo plot "%1.tim" using 1:2 title "%1 response times for pan/tilt commands" with lines >> %1.gp
46 gnuplot %1.gp
47
48 rem Now make another gnuplot control file for durations vs. number of that duration
49 echo set ylabel "Number" > %1d.gp
50 echo set xlabel "Duration" >> %1d.gp
51 echo set term latex >> %1d.gp
52 echo set output "%1d.inc" >> %1d.gp
53 echo set timestamp >> %1d.gp
54 echo set grid >> %1d.gp
55 echo set key left top >> %1d.gp
56 echo plot "%1.frq" using 2:1 title "%1 response times for pan/tilt commands" with lines >> %1d.gp
57 gnuplot %1d.gp
58
59 rem Find out when the test was run
60 head -5 %1.sec | tail -2 > %1.whn
61
62 rem List out statistics about the run
63 tail -8 %1.sec | head -7 > %1.dat
64
65 rem Get typical raw data
66 head -38 Original\%1.txt | tail -33 >%1.raw
67
68 rem Get typical output of MAKESECS
69 head -42 %1.sec | tail -36 >%1.cok

```

```
70 dir %1*.*
71
72 mv %1.cok SavedFiles
73 mv %1.dat SavedFiles
74 mv %1.frq SavedFiles
75 mv %1.gp SavedFiles
76 mv %1d.gp SavedFiles
77 mv %1.gt SavedFiles
78 mv %1.inc SavedFiles
79 mv %1d.inc SavedFiles
80 mv %1.lin SavedFiles
81 mv %1.max SavedFiles
82 mv %1.mde SavedFiles
83 mv %1.min SavedFiles
84 mv %1.raw SavedFiles
85 mv %1.sec SavedFiles
86 mv %1.srt SavedFiles
87 mv %1.tim SavedFiles
88 mv %1.typ SavedFiles
89 mv %1.whn SavedFiles
90
91 echo Make sure that all directory entries have something in them.
92 pause
```

N HProcit.bat

```

1  @echo off
2  rem $Header: d:/Binder2/Timings/RCS/HProcit.bat,v 1.6 2007-02-09 09:04:44-08 Hamilton Exp Hamilton $
3
4  rem Convert the input file to a useful format and get just the reply times,
5  rem Hernis version
6  makesecs Original\%1.txt %1.sec
7  hgettime %1.sec %1.gt
8  hjustime %1.gt %1.tim
9
10 rem Count the number of times in the file
11 lc %1.tim | tail -1 > %1.lin
12
13 rem Save the "typical times"
14 head -25 %1.tim > %1.typ
15
16 rem Now sort um by reply times
17 qsort %1.tim %1.srt /8:15 /1:7
18
19 rem Get the shortest times
20 head -10 %1.srt > %1.min
21
22 rem I don't know why this has to be done twice, but it works
23 head -10 %1.srt > %1.min
24
25 rem Get the longest times
26 tail -10 %1.srt > %1.max
27
28 rem Strip out the replys only
29 onlytime <%1.srt | efreq > %1.frq
30
31 rem Now get the most common reply times
32 qsort %1.frq %1.tmp
33 nocolon %1.tmp %1.ncl
34 tail -10 %1.ncl > %1.mde
35 del %1.tmp
36 del %1.ncl
37
38 rem Now generate a gnuplot control file for durations vs. reply number
39 echo set xlabel "Reply Number"> %1.gp
40 echo set ylabel "Duration" >> %1.gp
41 echo set term latex >> %1.gp
42 echo set output "%1.inc" >> %1.gp
43 echo set timestamp >> %1.gp
44 echo set grid >> %1.gp
45 echo set key left top >> %1.gp
46 echo plot "%1.tim" using 1:2 title "%1 response times for pan/tilt commands" with lines >> %1.gp
47 gnuplot %1.gp
48
49 rem Now make another gnuplot control file for durations vs. number of that duration
50 echo set ylabel "Number" > %1d.gp
51 echo set xlabel "Duration" >> %1d.gp
52 echo set term latex >> %1d.gp
53 echo set output "%1d.inc" >> %1d.gp
54 echo set timestamp >> %1d.gp
55 echo set grid >> %1d.gp
56 echo set key left top >> %1d.gp
57 echo plot "%1.frq" using 2:1 title "%1 response times for pan/tilt commands" with lines >> %1d.gp
58 gnuplot %1d.gp
59
60 rem Find out when the test was run
61 head -5 %1.sec | tail -2 > %1.whn
62
63 rem List out statistics about the run
64 tail -8 %1.sec | head -7 > %1.dat
65
66 rem Get typical raw data
67 head -38 Original\%1.txt | tail -33 >%1.raw
68
69 rem Get typical output of MAKESECS

```



```
70 head -42 %1.sec | tail -36 >%1.cok
71
72 dir %1*.*
73
74 mv %1.cok SavedFiles
75 mv %1.dat SavedFiles
76 mv %1.frq SavedFiles
77 mv %1.gp SavedFiles
78 mv %1d.gp SavedFiles
79 mv %1.gt SavedFiles
80 mv %1.inc SavedFiles
81 mv %1d.inc SavedFiles
82 mv %1.lin SavedFiles
83 mv %1.max SavedFiles
84 mv %1.mde SavedFiles
85 mv %1.min SavedFiles
86 mv %1.raw SavedFiles
87 mv %1.sec SavedFiles
88 mv %1.srt SavedFiles
89 mv %1.tim SavedFiles
90 mv %1.typ SavedFiles
91 mv %1.whn SavedFiles
92
93 echo Make sure that all directory entries have something in them.
94 pause
```

O ReDo.bat

```
1 @echo off
2 rem $Header: d:/Binder2/Timings/RCS/ReDo.bat,v 1.1 2007-01-24 08:53:06-08 Hamilton Exp Hamilton $
3 call procit E3012T
4 call procit E3012T1
5 call procit E3012T2
6 call procit E3012T3
7 call procit E3012TR
8 call procith HTime0H
9 call procit HTime0S
10 call procitp S2P
11 call procit S2Time0
12 call procit S3C1
13 call procitp S3P
14 call procit S3Ran1
15 call procit S3Time0
16 call procit S3Time0h
17 call procit S3Time1
18 call procit S3Time1H
19 call procitp S4p
20 call procit S4Ran1
21 call procit S4Time0
```

P ReDoL.bat

```

1 @echo off
2 rem $Header: d:/Binder2/Timings/RCS/ReDoL.bat,v 1.1 2007-01-24 08:53:11-08 Hamilton Exp Hamilton $
3 call flexit CommaOut
4 call gc CommaOut
5 del CommaOut.c
6 call flexit FixRaw
7 call gc FixRaw
8 del FixRaw.c
9 call flexit GetTimes
10 call gc GetTimes
11 del GetTimes.c
12 call flexit GetTimP
13 call gc GetTimP
14 del GetTimP.c
15 call flexit HGetTime
16 call gc HGetTime
17 del HGetTime.c
18 call flexit HJustTime
19 call gc HJustTime
20 del HJustTime.c
21 call flexit JustTime
22 call gc JustTime
23 del JustTime.c
24 call flexit JustTimP
25 call gc JustTimP
26 del JustTimP.c
27 call flexit MakeD3
28 call gc MakeD3
29 del MakeD3.c
30 call flexit NoColon
31 call gc NoColon
32 del NoColon.c
33 call flexit OnlyTime
34 call gc OnlyTime
35 del OnlyTime.c

```

Q Saveum.bat

```
1 @echo off
2 echo Moving all intermediate files to SavedFiles
3 rem $Header: d:/Binder2/Timings/RCS/Saveum.bat,v 1.2 2007-02-09 07:39:41-08 Hamilton Exp Hamilton $
4 if exist *.cok mv *.cok SavedFiles
5 if exist *.dat mv *.dat SavedFiles
6 if exist *.frq mv *.frq SavedFiles
7 if exist *.gp mv *.gp SavedFiles
8 if exist *.gt mv *.gt SavedFiles
9 if exist *.inc mv *.inc SavedFiles
10 if exist *.lin mv *.lin SavedFiles
11 if exist *.max mv *.max SavedFiles
12 if exist *.mde mv *.mde SavedFiles
13 if exist *.min mv *.min SavedFiles
14 if exist *.raw mv *.raw SavedFiles
15 if exist *.sec mv *.sec SavedFiles
16 if exist *.srt mv *.srt SavedFiles
17 if exist *.tim mv *.tim SavedFiles
18 if exist *.typ mv *.typ SavedFiles
19 if exist *.whn mv *.whn SavedFiles
```

Index

CommaOut.L, 4
CSV, 3, 4

D, 3
D Protocol, 4
DCE, 3, 4
DTE, 3, 4

FixRaw.L, 6

GetTimes.L, 8
GetTimP.L, 10
GlassKeyboard, 3

HGetTime.L, 12
HJusTime.L, 14
HProcit.bat, 32

JustTime.L, 16
JustTimP.L, 18

MakeD3.L, 20
MakeSecs, 4

NoColon.L, 26

OnlyTime.L, 27

Procit.bat, 28
ProcitP.bat, 30

ReDo.bat, 34
ReDoL.bat, 35

Saveum.bat, 36

TXT, 3