How to Collect and Analyze

Command Response Timing Data

8 July 2008

Eric Hamilton

Contents

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

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 Kbytes[2] 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.

[2] There were some exceptions to this quanity 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 %}
```

```
%%
",:"     fprintf(yyout,":");
",/"     fprintf(yyout,"/");

" "      ;

.|\n     ECHO;
%%
// yywrap is for End Of File processing, 1 = done
int yywrap(void)
{
        return 1;
}

main(int argc, char *argv[])
{
        fprintf(stderr,"$Id: CommaOut.L,v 1.2 2007-02-09 07:03:42-08 Hamilton Exp Hamilton $\n");

        // Do we have to deal with stdin/stdout?
        // 3 = one for the program name and one each for input and output names
        if (argc == 3) {
            // Nope
            // Get over the program, etc., name
            argc--;
            argv++;
            yyin = fopen(argv[0],"r");
            if (yyin == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }

            // Now get over the input file name and get to the output file name
            argc--;
            argv++;
            yyout = fopen(argv[0],"w");
            if (yyout == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }
        }
        else {
            // Woops we are using std... type io
            yyin  = stdin;
            yyout = stdout;
        }

        yylex();
        exit(EXIT_SUCCESS);
}

// $Log: CommaOut.L,v $
// Revision 1.2  2007-02-09 07:03:42-08  Hamilton
// Normal end of day data saving
//

```

B FixRaw.L

```
%{
// 123456789 123456789 123456789 123456789 123456789
// DTE     31 12/11/2006 6:44:36.715122 AM 6e
// DTE     32 12/11/2006 6:44:36.716164 AM 24
// DTE     33 12/11/2006 6:44:36.753803 AM 50
// 123456789 123456789 123456789 123456789 123456789
// $Header: d:/Binder2/Timings/RCS/FixRaw.L,v 1.2 2007-02-09 07:03:42-08 Hamilton Exp Hamilton $
// FixRaw is used to convert raw FTS files into something that ExSel can
//        understand.
#include <stdio.h>
#include <stdlib.h>
#undef yywrap
%}
%%
"DCE"         |
"DTE"         fprintf(yyout,"\"%s\",",yytext);

"."[0-9]{6}   fprintf(yyout,"%s,",yytext);

"/"[0-9]{1,2}"/"[0-9]{4}" "    fprintf(yyout,"%s,",yytext);
":"[0-9]{1,2}":"[0-9]{1,2}     fprintf(yyout,"%s,",yytext);

" "[0-9]{1,6}   fprintf(yyout,"%s,",yytext);

"A"           fprintf(yyout,"\"A");
"P"           fprintf(yyout,"\"P");
"M "          fprintf(yyout,"M");
[0-9a-f]{2}$ fprintf(yyout,"\",\"%s\"\n",yytext);

\n            ;

%%
//.|\n     ECHO;
// yywrap is for End Of File processing, 1 = done
int yywrap(void)
{
        return 1;
}

main(int argc, char *argv[])
{
        fprintf(stderr,"$Id: FixRaw.L,v 1.2 2007-02-09 07:03:42-08 Hamilton Exp Hamilton $\n");

        // Do we have to deal with stdin/stdout?
        // 3 = one for the program name and one each for input and output names
        if (argc == 3) {
            // Nope
            // Get over the program, etc., name
            argc--;
            argv++;
            yyin = fopen(argv[0],"r");
            if (yyin == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }

            // Now get over the input file name and get to the output file name
            argc--;
            argv++;
            yyout = fopen(argv[0],"w");
            if (yyout == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }
        }
        else {
            // 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,       1: DCE       1     0.000000   0.000000 ff
//     1,       2: DCE       2     0.001204   0.001204 01
//     1,       3: DCE       3     0.002239   0.001035 00
//     1,       4: DCE       4     0.003286   0.001047 0a
//     1,       5: DCE       5     0.004328   0.001042 19
//     1,       6: DCE       6     0.005376   0.001048 15
//     1,       7: DCE       7     0.006419   0.001043 39
//
//     1,       1:  DTE      8     0.007760   0.001043 ff
//     1,       2:  DTE      9     0.008802   0.001042 01
//     1,       3:  DTE     10     0.009853   0.001051 00
//     1,       4:  DTE     11     0.010878   0.001025 39
//123456789 123456789 123456789 123456789 123456789 123456789
//
// $Header: d:/Binder2/Timings/RCS/GetTimes.L,v 1.3 2007-02-09 07:03:42-08 Hamilton Exp Hamilton $
// GETTIMES is used to extract the start of a reply time delta from a
//          MAKESECS processed capture file.
//
#include <stdio.h>
#include <stdlib.h>
#undef yywrap
%}
%%
.*":  DTE ".*"ff"\n       ECHO;

.|\n     ;
%%
// yywrap is for End Of File processing, 1 = done
int yywrap(void)
{
        return 1;
}

main(int argc, char *argv[])
{
        fprintf(stderr,"$Id: GetTimes.L,v 1.3 2007-02-09 07:03:42-08 Hamilton Exp Hamilton $\n");

        // Do we have to deal with stdin/stdout?
        // 3 = one for the program name and one each for input and output names
        if (argc == 3) {
            // Nope
            // Get over the program, etc., name
            argc--;
            argv++;
            yyin = fopen(argv[0],"r");
            if (yyin == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }

            // Now get over the input file name and get to the output file name
            argc--;
            argv++;
            yyout = fopen(argv[0],"w");
            if (yyout == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }
        }
        else {
            // Woops we are using std... type io
            yyin  = stdin;
            yyout = stdout;
        }

        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,       1: DCE       1     0.000000   0.000000 a0
//     1,       2: DCE       2     0.001035   0.001035 01
//     1,       3: DCE       3     0.002077   0.001042 00
//     1,       4: DCE       4     0.003126   0.001049 0a
//     1,       5: DCE       5     0.004161   0.001035 07
//     1,       6: DCE       6     0.005210   0.001049 11
//     1,       7: DCE       7     0.006250   0.001040 af
//     1,       8: DCE       8     0.007300   0.001050 12
//
//     1,       1:  DTE      9     0.008621   0.001050 a2
//123456789 123456789 123456789 123456789 123456789 123456789
//
// $Header: d:/Binder2/Timings/RCS/GetTimP.L,v 1.2 2007-02-09 07:03:42-08 Hamilton Exp Hamilton $
// GETTIMES is used to extract the start of a reply time delta from a
//          MAKESECS processed capture file of P Protocol data.
//
#include <stdio.h>
#include <stdlib.h>
#undef yywrap
%}
%%
.*":  DTE ".*"a2"\n       ECHO;

.|\n     ;
%%
// yywrap is for End Of File processing, 1 = done
int yywrap(void)
{
        return 1;
}

main(int argc, char *argv[])
{
        fprintf(stderr,"$Id: GetTimP.L,v 1.2 2007-02-09 07:03:42-08 Hamilton Exp Hamilton $\n");

        // Do we have to deal with stdin/stdout?
        // 3 = one for the program name and one each for input and output names
        if (argc == 3) {
            // Nope
            // Get over the program, etc., name
            argc--;
            argv++;
            yyin = fopen(argv[0],"r");
            if (yyin == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }

            // Now get over the input file name and get to the output file name
            argc--;
            argv++;
            yyout = fopen(argv[0],"w");
            if (yyout == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }
        }
        else {
            // Woops we are using std... type io
            yyin  = stdin;
            yyout = stdout;
        }

        yylex();
        exit(EXIT_SUCCESS);
}
```

```
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,       1: DCE       1     0.000000   0.000000 ff
//     1,       2: DCE       2     0.001204   0.001204 01
//     1,       3: DCE       3     0.002239   0.001035 00
//     1,       4: DCE       4     0.003286   0.001047 0a
//     1,       5: DCE       5     0.004328   0.001042 19
//     1,       6: DCE       6     0.005376   0.001048 15
//     1,       7: DCE       7     0.006419   0.001043 39
//
//     1,       1:  DTE      8     0.007760   0.001043 ff
//     1,       2:  DTE      9     0.008802   0.001042 01
//     1,       3:  DTE     10     0.009853   0.001051 00
//     1,       4:  DTE     11     0.010878   0.001025 39
//123456789 123456789 123456789 123456789 123456789 123456789
//
// $Header: d:/Binder2/Timings/RCS/HGetTime.L,v 1.2 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $
// HETTIME is used to extract the start of a reply time delta from a
//          MAKESECS processed capture file. This is designed to run on a
//          Hernis Protocol capture file.
//
#include <stdio.h>
#include <stdlib.h>
#undef yywrap
%}
%%
.*":  DTE ".*"50"\n       ECHO;

.|\n     ;
%%
// yywrap is for End Of File processing, 1 = done
int yywrap(void)
{
        return 1;
}

main(int argc, char *argv[])
{
        fprintf(stderr,"$Id: HGetTime.L,v 1.2 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $\n");

        // Do we have to deal with stdin/stdout?
        // 3 = one for the program name and one each for input and output names
        if (argc == 3) {
            // Nope
            // Get over the program, etc., name
            argc--;
            argv++;
            yyin = fopen(argv[0],"r");
            if (yyin == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }

            // Now get over the input file name and get to the output file name
            argc--;
            argv++;
            yyout = fopen(argv[0],"w");
            if (yyout == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }
        }
        else {
            // Woops we are using std... type io
            yyin  = stdin;
            yyout = stdout;
        }

```

```
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

```
%{
//123456789 123456789 123456789 123456789 123456789 123456789
//     1,       1:  DTE     15     0.160269   0.001047 ff
//     2,       5:  DTE     26     0.307559   0.001043 ff
//     2,       9:  DTE     30     0.311722   0.001041 ff
//     4,      24:  DTE     59     0.454575   0.001042 ff
//     5,      28:  DTE     70     0.558371   0.001041 ff
//     6,      32:  DTE     81     0.672061   0.000881 ff
//     7,      36:  DTE    106     0.994849   0.001039 ff
//     8,      40:  DTE    117     1.107305   0.001034 ff
//     9,      44:  DTE    142     1.436274   0.001042 ff
//    10,      48:  DTE    153     1.542113   0.001048 ff
//123456789 123456789 123456789 123456789 123456789 123456789
//
// $Header: d:/Binder2/Timings/RCS/HJusTime.L,v 1.2 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $
// JustTime is used to remove everything but the record number and the
//          delta time from capture files processed by MAKESECS and GetTimes.
//          This is for use with Hernis Protocol captures.
#include <stdio.h>
#include <stdlib.h>
#undef yywrap
%}
%%
^" "*[0-9]*", " fprintf(yyout,"%s",yytext);
.{10}" 50"$    {fprintf(yyout,"%c%c%c%c%c%c%c%c%c\n",
                               yytext[0],yytext[1],yytext[2],
                               yytext[3],yytext[4],yytext[5],
                               yytext[6],yytext[7],yytext[8],yytext[9]);
               }

.|\n            ;

%%
// yywrap is for End Of File processing, 1 = done
int yywrap(void)
{
        return 1;
}

main(int argc, char *argv[])
{
        fprintf(stderr,"$Id: HJusTime.L,v 1.2 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $\n");

        // Do we have to deal with stdin/stdout?
        // 3 = one for the program name and one each for input and output names
        if (argc == 3) {
            // Nope
            // Get over the program, etc., name
            argc--;
            argv++;
            yyin = fopen(argv[0],"r");
            if (yyin == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }

            // Now get over the input file name and get to the output file name
            argc--;
            argv++;
            yyout = fopen(argv[0],"w");
            if (yyout == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }
        }
        else {
            // 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

```
%{
//123456789 123456789 123456789 123456789 123456789 123456789
//     1,       1:  DTE     15     0.160269   0.001047 ff
//     2,       5:  DTE     26     0.307559   0.001043 ff
//     2,       9:  DTE     30     0.311722   0.001041 ff
//     4,      24:  DTE     59     0.454575   0.001042 ff
//     5,      28:  DTE     70     0.558371   0.001041 ff
//     6,      32:  DTE     81     0.672061   0.000881 ff
//     7,      36:  DTE    106     0.994849   0.001039 ff
//     8,      40:  DTE    117     1.107305   0.001034 ff
//     9,      44:  DTE    142     1.436274   0.001042 ff
//    10,      48:  DTE    153     1.542113   0.001048 ff
//123456789 123456789 123456789 123456789 123456789 123456789
//
// $Header: d:/Binder2/Timings/RCS/JustTime.L,v 1.3 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $
// JustTime is used to remove everything but the record number and the
//          delta time from capture files processed by MAKESECS and GetTimes.
#include <stdio.h>
#include <stdlib.h>
#undef yywrap
%}
%%
^" "*[0-9]*", " fprintf(yyout,"%s",yytext);
.{10}" ff"$    {fprintf(yyout,"%c%c%c%c%c%c%c%c%c%c\n",
                               yytext[0],yytext[1],yytext[2],
                               yytext[3],yytext[4],yytext[5],
                               yytext[6],yytext[7],yytext[8],yytext[9]);
               }

.|\n            ;

%%
// yywrap is for End Of File processing, 1 = done
int yywrap(void)
{
        return 1;
}

main(int argc, char *argv[])
{
        fprintf(stderr,"$Id: JustTime.L,v 1.3 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $\n");

        // Do we have to deal with stdin/stdout?
        // 3 = one for the program name and one each for input and output names
        if (argc == 3) {
            // Nope
            // Get over the program, etc., name
            argc--;
            argv++;
            yyin = fopen(argv[0],"r");
            if (yyin == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }

            // Now get over the input file name and get to the output file name
            argc--;
            argv++;
            yyout = fopen(argv[0],"w");
            if (yyout == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }
        }
        else {
            // Woops we are using std... type io
            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

```
%{
//123456789 123456789 123456789 123456789 123456789 123456789
//     1,       1:  DTE      9     0.008621   0.001050 a2
//     2,       2:  DTE     18     0.120798   0.001374 a2
//     3,       3:  DTE     27     0.227408   0.001042 a2
//     4,       4:  DTE     36     0.339792   0.001459 a2
//123456789 123456789 123456789 123456789 123456789 123456789
//
// $Header: d:/Binder2/Timings/RCS/JustTimP.L,v 1.2 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $
// JustTimP is used to remove everything but the record number and the
//          delta time from capture files processed by MAKESECS and GetTimP
//          for P Protocol data.
#include <stdio.h>
#include <stdlib.h>
#undef yywrap
%}
%%
^" "*[0-9]*", " fprintf(yyout,"%s",yytext);
.{10}" a2"$    {fprintf(yyout,"%c%c%c%c%c%c%c%c%c%c\n",
                               yytext[0],yytext[1],yytext[2],
                               yytext[3],yytext[4],yytext[5],
                               yytext[6],yytext[7],yytext[8],yytext[9]);
               }

.|\n            ;

%%
// yywrap is for End Of File processing, 1 = done
int yywrap(void)
{
        return 1;
}

main(int argc, char *argv[])
{
        fprintf(stderr,"$Id: JustTimP.L,v 1.2 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $\n");

        // Do we have to deal with stdin/stdout?
        // 3 = one for the program name and one each for input and output names
        if (argc == 3) {
            // Nope
            // Get over the program, etc., name
            argc--;
            argv++;
            yyin = fopen(argv[0],"r");
            if (yyin == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
                               argv[0]);
                exit(EXIT_FAILURE);
            }

            // Now get over the input file name and get to the output file name
            argc--;
            argv++;
            yyout = fopen(argv[0],"w");
            if (yyout == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
                               argv[0]);
                exit(EXIT_FAILURE);
            }
        }
        else {
            // Woops we are using std... type io
            yyin  = stdin;
            yyout = stdout;
        }

        yylex();
        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

```
%{
// 123456789 123456789 123456789 123456789 123456789 123456789
//      1,       1: DCE       1     0.000000   0.000000 ff
//      1,       2: DCE       2     0.001031   0.001031 01
//      1,       3: DCE       3     0.002078   0.001047 00
//      1,       4: DCE       4     0.003121   0.001043 08
//      1,       5: DCE       5     0.004163   0.001042 00
//      1,       6: DCE       6     0.005205   0.001042 09
//      1,       7: DCE       7     0.006298   0.001093 12
//
//      1,       1:  DTE      8     0.007929   0.001093 ff
//      1,       2:  DTE      9     0.008965   0.001036 01
//      1,       3:  DTE     10     0.010005   0.001040 00
//      1,       4:  DTE     11     0.011047   0.001042 12
// 123456789 123456789 123456789 123456789 123456789 123456789
//
// $Header: d:/Binder2/Timings/RCS/MakeD3.L,v 1.4 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $
// MakeD is used to process D Protocol messages that have been previously
//       processed by MAKESECS. The result is a more intellegent version
//       of the data.
#include <stdio.h>
#include <stdlib.h>
#include "dproto.h"

#undef yywrap

char string[100];
char bytess[100];
char savestart[1000];
char hexbyte[10];

void DecodeIt(void);
unsigned char MakeHex(char byte);
void addtwo(void);

int dcecount = 0;
int dtecount = 0;
int doingdce = 0;
int doingdte = 0;

%}
%%
.*": DCE ".*\n   {strncpy(savestart,yytext,55);
                  hexbyte[0] = ' ';
                  hexbyte[1] = yytext[53];
                  hexbyte[2] = yytext[54];
                  hexbyte[3] = '\0';
                  dcecount++;
                  doingdte = 0;
                  if (strcmp(" ff",hexbyte) == 0)
                  {
                      if (doingdce == 1)
                          fprintf(yyout,"\n");
                      dcecount = 0;
                      fprintf(yyout,"%s",savestart);
                      strcpy(string,hexbyte);
                  }
                  else
                  {
                      if (doingdte == 1)
                      {
                          fprintf(yyout,"%s",savestart);
                      }
                      else
                      {
                          fprintf(yyout,"%s",hexbyte);
                          strcat(string,hexbyte);
                          if (dcecount >= 6)
                          {
```

```
                                DecodeIt();
                                doingdce = 1;
                                BEGIN 0;
                            }
                        }
                    }
                   }

.*":  DTE".*\n   {strncpy(savestart,yytext,55);
                    hexbyte[0] = ' ';
                    hexbyte[1] = yytext[53];
                    hexbyte[2] = yytext[54];
                    hexbyte[3] = '\0';
                    dtecount++;
                    doingdce = 0;
                    if (strcmp(" ff",hexbyte) == 0)
                    {
                        if (doingdte == 1)
                            fprintf(yyout,"\n");
                        dtecount = 0;
                        fprintf(yyout,"%s",savestart);
                        strcpy(string,hexbyte);
                    }
                    else
                    {
                        if (doingdce == 1)
                        {
                            fprintf(yyout,"%s",savestart);
                        }
                        else
                        {
                            fprintf(yyout,"%s",hexbyte);
                            strcat(string,hexbyte);
                            if (dtecount >= 3)
                            {
                                doingdte = 1;
                                fprintf(yyout,"\n");
                                BEGIN 0;
                            }
                        }
                    }
                   }


\n       ECHO;
.        ECHO;
%%
// This routine is used to add two bytes from the input line onto the
// main string it also echos the bytes to the output file.
void addtwo(void)
{
    sprintf(bytess,"%c%c ",yytext[46],yytext[47]);
    strcat(string,bytess);
    fprintf(yyout,"%s",bytess);
}


// This routine converts an AsciiHex byte into a Hex byte.
//
unsigned char MakeHex(char byte)
{
unsigned char temp;

    temp = byte - 0x30;         // 0 = 0x30, 9 = 0x39, a = 0x61, f = 0x66
    if (temp >= 0x30)           // temp = 0x00 --> 0x36
        temp -= (0x31 - 0x0a);  // temp = 0x00 --> 0x0f
    return temp;
}


// Used to interpert a D Protocol command
//
```

```
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

```
%{
// $Header: d:/Binder2/Timings/RCS/NoColon.L,v 1.2 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $
// NOCOLON is used to change all colons ":" into commas ",".
#include <stdio.h>
#include <stdlib.h>
#undef yywrap
%}
%%
":"      fprintf(yyout,",");

.|\n     ECHO;
%%
// yywrap is for End Of File processing, 1 = done
int yywrap(void)
{
        return 1;
}

main(int argc, char *argv[])
{
        fprintf(stderr,"$Id: NoColon.L,v 1.2 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $\n");

        // Do we have to deal with stdin/stdout?
        // 3 = one for the program name and one each for input and output names
        if (argc == 3) {
            // Nope
            // Get over the program, etc., name
            argc--;
            argv++;
            yyin = fopen(argv[0],"r");
            if (yyin == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }

            // Now get over the input file name and get to the output file name
            argc--;
            argv++;
            yyout = fopen(argv[0],"w");
            if (yyout == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }
        }
        else {
            // Woops we are using std... type io
            yyin  = stdin;
            yyout = stdout;
        }

        yylex();
        exit(EXIT_SUCCESS);
}


// $Log: NoColon.L,v $
// Revision 1.2  2007-02-09 07:03:43-08  Hamilton
// Normal end of day data saving
//

```

K OnlyTime.L

```
%{
//123456789 123456789
//     1,   0.001093
//     2,   0.001037
//     3,   0.000882
//     4,   0.001041
//123456789 123456789
// $Header: d:/Binder2/Timings/RCS/OnlyTime.L,v 1.3 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $
// OnlyTime is used to extract just the time data from files processed by
//          JustTime.
#include <stdio.h>
#include <stdlib.h>
#undef yywrap
%}
%%
.{10}$   fprintf(yyout,"%s%\n",yytext);

.|\n     ;
%%
// yywrap is for End Of File processing, 1 = done
int yywrap(void)
{
        return 1;
}

main(int argc, char *argv[])
{
        fprintf(stderr,"$Id: OnlyTime.L,v 1.3 2007-02-09 07:03:43-08 Hamilton Exp Hamilton $\n");

        // Do we have to deal with stdin/stdout?
        // 3 = one for the program name and one each for input and output names
        if (argc == 3) {
            // Nope
            // Get over the program, etc., name
            argc--;
            argv++;
            yyin = fopen(argv[0],"r");
            if (yyin == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for reading, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }

            // Now get over the input file name and get to the output file name
            argc--;
            argv++;
            yyout = fopen(argv[0],"w");
            if (yyout == NULL) {
                fprintf(stderr,"\aCould not open \"%s\" for writing, quitting",
                                argv[0]);
                exit(EXIT_FAILURE);
            }
        }
        else {
            // Woops we are using std... type io
            yyin  = stdin;
            yyout = stdout;
        }

        yylex();
        exit(EXIT_SUCCESS);
}

// $Log: OnlyTime.L,v $
// Revision 1.3  2007-02-09 07:03:43-08  Hamilton
// Normal end of day data saving
//

```

L Procit.bat

```
@echo off
rem $Header: d:/Binder2/Timings/RCS/ProcIt.bat,v 1.13 2007-02-12 11:28:49-08 Hamilton Exp Hamilton $

rem Convert the input file to a useful format and get just the reply times
makesecs Original\%1.txt %1.sec
gettimes %1.sec %1.gt
justtime %1.gt  %1.tim

rem Count the number of times in the file
lc %1.tim | tail -1 > %1.lin

rem Save the "typical times"
head -25 %1.tim > %1.typ

rem Now sort um by reply times
qsort %1.tim %1.srt /8:15 /1:7

rem Get the shortest times
head -12 %1.srt > %1.min

rem I don't know why this has to be done twice, but it works
head -12 %1.srt > %1.min

rem Get the longest times
tail -12 %1.srt > %1.max

rem Strip out the replys only
onlytime <%1.srt  | efreq > %1.frq

rem Now get the most common reply times
qsort    %1.frq %1.tmp
nocolon  %1.tmp %1.ncl
tail -12 %1.ncl > %1.mde
del %1.tmp
del %1.ncl

rem Now generate a gnuplot control file for durations vs. reply number
echo set xlabel "Reply Number">  %1.gp
echo set ylabel "Duration"    >> %1.gp
echo set term latex           >> %1.gp
echo set output "%1.inc"      >> %1.gp
echo set timestamp            >> %1.gp
echo set grid                 >> %1.gp
echo set key left top         >> %1.gp
echo plot "%1.tim" using 1:2 title "%1.tim response times for pan/tilt commands" with lines >> %1.gp
gnuplot %1.gp

rem Now make another gnuplot control file for durations vs. number of that duration
echo set ylabel "Number"      >  %1d.gp
echo set xlabel "Duration"    >> %1d.gp
echo set term latex           >> %1d.gp
echo set output "%1d.inc"     >> %1d.gp
echo set timestamp            >> %1d.gp
echo set grid                 >> %1d.gp
echo set key left top         >> %1d.gp
echo plot "%1.frq" using 2:1 title "%1.frq response times for pan/tilt commands" with lines >> %1d.gp
gnuplot %1d.gp

rem Find out when the test was run
head -5 %1.sec | tail -2 > %1.whn

rem List out statistics about the run
tail -8 %1.sec | head -7 > %1.dat

rem Get typical raw data
head -38 Original\%1.txt | tail -33 >%1.raw

rem Get typical output of MAKESECS
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

```
@echo off
rem $Header: d:/Binder2/Timings/RCS/ProcitP.bat,v 1.4 2007-02-09 09:04:45-08 Hamilton Exp Hamilton $

rem Convert the input file to a useful format and get just the reply times
makesecs Original\%1.txt %1.sec
gettimp  %1.sec %1.gt
justtimp %1.gt  %1.tim

rem Count the number of times in the file
lc %1.tim | tail -1 > %1.lin

rem Save the "typical times"
head -25 %1.tim > %1.typ

rem Now sort um by reply times
qsort %1.tim %1.srt /8:15 /1:7

rem Get the shortest times
head -10 %1.srt > %1.min

rem I don't know why this has to be done twice, but it works
head -10 %1.srt > %1.min

rem Get the longest times
tail -10 %1.srt > %1.max

rem Strip out the replys only
onlytime <%1.srt  | efreq > %1.frq

rem Now get the most common reply times
qsort    %1.frq %1.tmp
nocolon  %1.tmp %1.ncl
tail -10 %1.ncl > %1.mde
del %1.tmp
del %1.ncl

rem Now generate a gnuplot control file for durations vs. reply number
echo set xlabel "Reply Number">  %1.gp
echo set ylabel "Duration"    >> %1.gp
echo set term latex           >> %1.gp
echo set output "%1.inc"      >> %1.gp
echo set timestamp            >> %1.gp
echo set grid                 >> %1.gp
echo set key left top         >> %1.gp
echo plot "%1.tim" using 1:2 title "%1 response times for pan/tilt commands" with lines >> %1.gp
gnuplot %1.gp

rem Now make another gnuplot control file for durations vs. number of that duration
echo set ylabel "Number"      >  %1d.gp
echo set xlabel "Duration"    >> %1d.gp
echo set term latex           >> %1d.gp
echo set output "%1d.inc"     >> %1d.gp
echo set timestamp            >> %1d.gp
echo set grid                 >> %1d.gp
echo set key left top         >> %1d.gp
echo plot "%1.frq" using 2:1 title "%1 response times for pan/tilt commands" with lines >> %1d.gp
gnuplot %1d.gp

rem Find out when the test was run
head -5 %1.sec | tail -2 > %1.whn

rem List out statistics about the run
tail -8 %1.sec | head -7 > %1.dat

rem Get typical raw data
head -38 Original\%1.txt | tail -33 >%1.raw

rem Get typical output of MAKESECS
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

```
@echo off
rem $Header: d:/Binder2/Timings/RCS/HProcit.bat,v 1.6 2007-02-09 09:04:44-08 Hamilton Exp Hamilton $

rem Convert the input file to a useful format and get just the reply times,
rem Hernis version
makesecs Original\%1.txt %1.sec
hgettime %1.sec %1.gt
hjustime %1.gt  %1.tim

rem Count the number of times in the file
lc %1.tim | tail -1 > %1.lin

rem Save the "typical times"
head -25 %1.tim > %1.typ

rem Now sort um by reply times
qsort %1.tim %1.srt /8:15 /1:7

rem Get the shortest times
head -10 %1.srt > %1.min

rem I don't know why this has to be done twice, but it works
head -10 %1.srt > %1.min

rem Get the longest times
tail -10 %1.srt > %1.max

rem Strip out the replys only
onlytime <%1.srt  | efreq > %1.frq

rem Now get the most common reply times
qsort    %1.frq %1.tmp
nocolon  %1.tmp %1.ncl
tail -10 %1.ncl > %1.mde
del %1.tmp
del %1.ncl

rem Now generate a gnuplot control file for durations vs. reply number
echo set xlabel "Reply Number">  %1.gp
echo set ylabel "Duration"    >> %1.gp
echo set term latex           >> %1.gp
echo set output "%1.inc"      >> %1.gp
echo set timestamp            >> %1.gp
echo set grid                 >> %1.gp
echo set key left top         >> %1.gp
echo plot "%1.tim" using 1:2 title "%1 response times for pan/tilt commands" with lines >> %1.gp
gnuplot %1.gp

rem Now make another gnuplot control file for durations vs. number of that duration
echo set ylabel "Number"      >  %1d.gp
echo set xlabel "Duration"    >> %1d.gp
echo set term latex           >> %1d.gp
echo set output "%1d.inc"     >> %1d.gp
echo set timestamp            >> %1d.gp
echo set grid                 >> %1d.gp
echo set key left top         >> %1d.gp
echo plot "%1.frq" using 2:1 title "%1 response times for pan/tilt commands" with lines >> %1d.gp
gnuplot %1d.gp

rem Find out when the test was run
head -5 %1.sec | tail -2 > %1.whn

rem List out statistics about the run
tail -8 %1.sec | head -7 > %1.dat

rem Get typical raw data
head -38 Original\%1.txt | tail -33 >%1.raw

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 HJusTime
19  call gc     HJusTime
20  del         HJusTime.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