D File Formats

Please note

The information presented in this appendix is intended to be used by experienced Pascal programmers. It is not intended to be used by the casual programmer and is therefore not as detailed as would be necessary for such a programmer. These file structures are useful in custom data processing applications such as uploading data files to remote computer systems other than an HP 1000 RTE 6/VM system.

Every attempt has been made to insure the accuracy of the information presented here, however this does not imply that HP in any way guarantees this. Although every attempt will be made to preserve these file structures, HP reserves the right to modify them at any time.

Mass spectral data files

Overall structure

Header Record 512 bytes
Normalization Records 10 at 24 bytes each
Spectral Records N-Variable Length Records
Directory Records N at 12 bytes each

Header record

Byte Offset	
0	File Number : String [3]
2	
4	File String : String [19]
	⋮
22	
24	Data Name : String [61]
	⋮
84	
86	Misc Info : String [61]
	⋮
146	
148	Operator : String [29]
	⋮
176	
178	Date-Time : String [29]
	⋮
206	
208	Instrument Model : String [9]
	⋮
216	
218	Inlet : String [9]

	⋮
226	
228	Method File : String [19]
	⋮
246	
248	File Type : Integer
250	
252	Seq Index : Shortint
254	ALS Bottle : Shortint
256	Replicate : Shortint
258	Directory Entry Type : Shortint
260	Directory Offset : Integer (in Words) [* 1 *]
262	
264	Data Offset : Integer (in Words) [* 2 *]
266	
268	Run Table Offset : Integer (in Words) [Unused]
270	
272	Normalization Records Offset : Integer (in Words) [* 3 *]
274	
276	Extra Records : Shortint
278	Number of Data Records : Integer
280	
282	Starting Retention Time : Integer (in milliseconds)

284	
286	Ending Retention Time : Integer (in milliseconds)
288	
290	Maximum Signal : Integer
292	
294	Minimum Signal : Integer
296	
298	Unused
	. . .
510	

Normalization record

0	Normalization Mass : Real
	. . .
6	
8	Slope : Real
	. . .
14	
16	Intercept : Real
	. . .
22	

Spectral records

Byte Offset	
0	Number of Words : Shortint (inclusive)
2	Retention Time : Integer (in milliseconds)
4	
6	Number of Words : Shortint (inclusive)
8	Data Type : Shortint
10	Status Word : Shortint
12	Number of Peaks : Shortint
14	Base Peak : Unsigned Shortint (mass*20)
16	Base Peak Abundance : Shortint
18	Mass : Unsigned Shortint (mass*20)
20	Abundance : Shortint
22	
	.
	.
	.
	Mass : Unsigned Shortint (mass*20)
	Abundance : Shortint

Directory records

Byte Offset	
0	Spectrum Offset : Integer (in Words)
2	
4	Retention Time : Integer (milliseconds)
6	
8	Total Signal : Integer
10	

Access philosophy

To illustrate how to access information in the mass spectral data files, the following program asks for the name of a mass spectral data file, and the name of the file that the tabulations will be sent to. The program then continually asks for a spectrum number and proceeds to tabulate the spectrum to the output file. No checks for limits are performed, so some caution should be exercised to avoid file access errors.

```
0 $SYSPROG ON$   (for SIZEOF function)
0 $DEBUG ON$
0 $LINES 43$

0 {-----------------------------------------------------------}
0 {  NOTE: Sysprog is used for the SIZEOF extension.           }
0 {-----------------------------------------------------------}

0 PROGRAM MS_Data_Files (INPUT, OUTPUT);

1 {-----------------------------------------------------------}
1 {----------------- Type Declarations ---------------------}
1 {-----------------------------------------------------------}
1 TYPE

1   Byte = 0 .. 255;
1   String80 = String [80];  {For File Names}
1   Shortint = -32768 .. 32767;
1   Unsigned_Shortint = 0 .. 65535;
1
1   {*** This type will be used to access the data file ***}
1   Int_File_Type = FILE OF Shortint;

1   {*** A 'general' type for data (and other) file headers}
1   Header_info_type = RECORD
1        File_num_str   : String [3];
1        File_str       : String [19]; {e.g. GC/MS DATA FILE}
1        Data_name      : String [61]; {User input name}
1        Misc_info      : String [61]; {User input name}
1        Operator       : String [29];
1        Date_time      : String [29];
1        Inst_model     : String [9];  {e.g. 5970      }
1        Inlet          : String [9];  {e.g. GC,LC, ETC.}
1        Method_file    : String [19];
1        File_type      : Integer;
1        Seq_index      : Shortint;
1        Als_bottle     : Shortint;
1        Replicate      : Shortint;   {per ALS index   }
1        Dir_ent_type   : Shortint;   {type of dir ents }
1        Dir_offset     : Integer;
1        Data_offset    : Integer;
1        Run_Tbl_offset : Integer;
1        Norm_offset    : Integer;
```

```
             Extra_records  : Shortint;    {Number of 256 BYTE
1                                           Records following}
1            Num_records    : Integer;     {Number of data recs}
1            Start_rtime    : Integer;     {Starting ret time }
1            End_rtime      : Integer;     {Last retention time}
1            Max_signal     : Integer;     {Maximum dir signal }
1            Min_signal     : Integer      {Minimum dir signal }
                          END;

    {*** This type is used to fill out the rest of the header
1            type so that it will always be 512 bytes long ***}
1   File_header_type = RECORD
1                 Info : Header_info_type;
1               Filler : PACKED ARRAY
1                        [1 .. (512 - SIZEOF (Header_info_type))]
1                        OF Byte;
1                    END;

1   {*** This is the normalization record stuff ***}
1   Norm_record_type = RECORD
1              Norm_Mass : Real;
1                  Slope : Real;
1              Intercept : Real;
1                        END;

1   Norm_record_array_type = ARRAY [1 .. 10]
1                                  OF Norm_record_type;

1   {*** Abundance is in special packed format (powers of 8) ***}
1   Abundance_Rec = PACKED RECORD
1                         Scale : 0 .. 3;      {x1, x8, x64, x512}
1                      Mantissa : 0 .. 16383;
1                              END;

1   Mass_Abund_Type = PACKED RECORD
1                            Mass : Unsigned_Shortint;
1                           Abund : Abundance_Rec;
1                                END;

1   {*** This is the primary Mass Spectral data record ***}
    {     Note: The array [1 .. 2000] of Mass_Abund_Type
                in no way is meant to place a limit on the
                number of mass spectral peaks any one spectrum
```

```
 87:S                        can contain,  this is simply a convenient
 88:S                        number for this example.  The real guides
 89:S                        to the number of mass spectral peaks should
 90:D        1               be Num_Peaks.                             }
 91:D        1   Spectral_Rec = PACKED RECORD
 92:D        1                 Num_Words : Shortint;
 93:D        1                  Ret_Time : Integer;
 94:D        1              Words_Less_3 : Shortint;
 95:D        1                 Data_Type : Shortint;
 96:D        1                    Status : Shortint;
 97:D        1                 Num_Peaks : Shortint;
 98:D        1                 Base_Peak : Unsigned_Shortint;
 99:D        1                Base_Abund : Shortint;
100:D        1                      Data : PACKED ARRAY [1 .. 2000]
101:D        1                                          OF Mass_Abund_Type;
102:D        1                          END;
103:S
104:S            {*** This is what PASCAL calls a 'Variant Record' it is
105:S                 used to get around PASCAL's tendency to want to
106:S                 access files as fixed length records.  Spectral
107:S                 records are by nature variable length.  In order
108:S                 to access them from the file, the data is read as
109:D        1        Shortints and later accessed as the Spectral record ***}
110:D        1   Spectral_Variant = PACKED RECORD
111:D        1                       CASE Boolean OF
112:D        1                          True : (Spec : Spectral_Rec);
113:D        1                         False : (Int : PACKED ARRAY
114:D        1                             [1 .. SIZEOF (Spectral_Rec) DIV 2]
115:D        1                                  OF Shortint);
116:D        1                              END;
117:S
118:S            {*** This is the directory record.  Directory entries are
119:S                 used to provide pseudo-random access to the variable
120:S                 length spectral records.  Each spectrum will have one
121:S                 directory entry.  Directory entries are also used to
122:D        1        reconstruct the total ion chromatogram very quickly ***}
123:D        1   Directory_Entry = PACKED RECORD
124:D        1                       Offset : Integer;
125:D        1                     Ret_Time : Integer;
126:D        1                      Tot_Sig : Integer;
127:D        1                             END;
128:S
129:S            {*** This type is used in the exact same way as the Spectral
```

```
 130:D        1       variant, but in this case for the directory entries ***)
 131:D        1  Directory_Variant = PACKED RECORD
 132:D        1                        CASE Boolean OF
 133:D        1                          True : (Dir : Directory_Entry);
 134:D        1                         False : (Int : PACKED ARRAY
 135:D        1                           [1 .. SIZEOF (Directory_Entry) DIV 2]
 136:D        1                                OF Shortint);
 137:D        1                             END;
 138:S
 139:D        1 {-----------------------------------------------------}
 140:D        1 {-------------- Variable Declarations -----------------}
 141:D        1 {-----------------------------------------------------}
 142:D        1 VAR
 143:D   -512 1            Header : File_Header_Type;
 144:D  -8530 1          Spectrum : Spectral_Rec;
 145:D  -8534 1          Spec_Num : Integer;
 146:D  -9198 1         Data_File : Int_File_Type;
 147:D  -9280 1         File_Name : String80;
 148:D  -9944 1       Output_File : Text;
 149:D -10026 1  Output_File_Name : String80;
 150:S
 151:D -10026 1 {-----------------------------------------------------}
 152:D -10026 1 {------------------ Procedures ----------------------}
 153:D -10026 1 {-----------------------------------------------------}
 154:D        1 PROCEDURE Read_Data_Header (      File_Name : String80;
 155:D    -82 2                              VAR Data_Header : File_Header_Type);
 156:S
 157:D    -82 2 VAR
 158:D  -1256 2   Header_File : FILE OF File_Header_Type;
 159:S
 160:C        2 BEGIN {Read_Data_Header}
 161:C        2
 162:S           (*** NOTE: Because the header starts at word 0 of
 163:S                      the file, the file can be opened for
 164:S                      access as a 'FILE OF File_Header_Type'
 165:C        2             and only the first record read.  ***)
 166*C        2   OPEN (Header_File, File_Name);
 167*C        2   READ (Header_File, Data_Header);
 168*C        2   CLOSE (Header_File);
 169:S
 170*C        2 END;  {Read_Data_Header}
 171:S
 172:D -10026 1 {-----------------------------------------------------}
```

```
Pascal [Rev 3.2  8/29/87] ReadSpec.TEXT          2-May-88 11:27:52 Page 5

  173:D        1 PROCEDURE Get_Spectrum (VAR       Data_File : Int_File_Type;
  174:D        2                                Spectrum_Num : Integer;
  175:D        2                                   Dir_Start : Integer;
  176:D        2                        ANYVAR      Spectrum : Spectral_Variant);
  177:S
  178:D        2 VAR
  179:D    -4  2               I : Integer;
  180:D   -16  2       Dir_Entry : Directory_Variant;
  181:D   -20  2   Length_In_Words : Integer;
  182:D   -24  2   This_Dir_Offset : Integer;
  183:S
  184:C        2 BEGIN (Get_Spectrum)
  185:S
  186:S            {*** Calculate the start of the directory
  187:S                 entry for this spectrum.  Note, SIZEOF
  188:S                 returns bytes, but words (16 bits) are
  189:C        2         needed, therefore the 'DIV 2' ***}
  190*C        2   This_Dir_Offset := Dir_Start +
  191:C        2                      ((Spectrum_Num - 1) *
  192:C        2                      (SIZEOF (Directory_Entry) DIV 2));
  193:S
  194:S            {*** Read in the directory entry for this spectrum.
  195:S                 Note that the 'Int' portion of the record is used
  196:S                 for this access to satisfy PASCAL's type matching
  197:C        2         requirements, the 'Dir' portion will be used later ***}
  198*C        2   SEEK (Data_File, This_Dir_Offset);
  199*C        2   FOR I := 1 TO (SIZEOF (Directory_Entry) DIV 2) DO
  200*C        3     READ (Data_File, Dir_Entry.Int [I]);
  201:S
  202:S            {*** Now seek to the beginning of this spectrum record
  203:S                 as given by the directory entry and read in the
  204:C        2         first word to determine the length of this spectrum ***}
  205*C        2   SEEK (Data_File, Dir_Entry.Dir.Offset);
  206*C        2   READ (Data_File, Length_In_Words);
  207:S
  208:S            {*** Seek back to the beginning of this spectrum record
  209:C        2         and read the entire spectrum ***}
  210*C        2   SEEK (Data_File, Dir_Entry.Dir.Offset);
  211*C        2   FOR I := 1 TO Length_In_Words DO
  212*C        3     READ (Data_File, Spectrum.Int [I]);
  213:S
  214*C        2 END;  (Next_Spectrum)
  215:S
```

```
216:D-10026  1 {--------------------------------------------------------------}
217:D        1 PROCEDURE Tab_Spectrum (VAR  Spectrum : Spectral_Rec;
218:D        2                        VAR Text_File : Text);
219:S
220:D        2 VAR
221:D    -4  2              I : Integer;
222:D    -8  2          Index : Integer;
223:D   -12  2      Abundance : Integer;
224:D   -16  2  Mass_Peak_Num : Integer;
225:S
226:C        2 BEGIN {Tab_Spectrum}
227:S
228*C        2   WRITELN (Text_File);
229*C        2   WRITELN (Text_File, 'Spectrum Retention Time : ':40,
230:C        2                       (Spectrum.Ret_Time / 60000):0:3);
231*C        2   WRITELN (Text_File, 'Number of Peaks : ':40,
232:C        2                       Spectrum.Num_Peaks:0);
233*C        2   WRITELN (Text_File, 'Base Peak : ':40,
234:C        2                       (Spectrum.Base_Peak / 20):0:2);
235*C        2   WRITELN (Text_File);
236:S
237*C        2   FOR Index := 1 TO 4 DO
238*C        3     WRITE (Text_File, '       Mass   Abund');
239*C        2   WRITELN (Text_File);
240:S
241:S            {*** Spectra are stored high mass to
242:C        2          low so print in reverse order ***}
243*C        2   Mass_Peak_Num := 0;
244*C        2   FOR Index := Spectrum.Num_Peaks DOWNTO 1 DO BEGIN
245*C        3     WITH Spectrum.Data [Index] DO BEGIN
246*C        4       Abundance := Abund.Mantissa;
247*C        4       FOR I := 1 TO Abund.Scale DO
248*C        5         Abundance := Abundance * 8;
249*C        4       WRITE (Text_File, (Mass / 20):11:2, (Abundance):8);
250:C        4     END;  {With Spectrum}
251:S
252:S              {*** If four mass abundance pairs have
253:C        3          been printed, then skip to a new line ***}
254*C        3     Mass_Peak_Num := Mass_Peak_Num + 1;
255*C        3     IF ((Mass_Peak_Num MOD 4) = 0) THEN
256*C        4       WRITELN (Text_File);
257:C        3   END;  {For Index}
258*C        2   WRITELN (Text_File);
```

```
  259*C        2   WRITELN (Text_File);
  260:S
  261*C        2 END;  {Tab_Spectrum}
  262:S
  263:D-10026  1 {--------------------------------------------------------}
  264:D-10026  1 {-------------------- Main Program ---------------------}
  265:D-10026  1 {--------------------------------------------------------}
  266:C        1 BEGIN {Main}
  267:S
  268:C        1   {*** Ask the user for the data file name ***}
  269*C        1   WRITE (OUTPUT, 'Data File Name? >> ');
  270*C        1   READLN (INPUT, File_Name);
  271:S
  272:C        1   {*** Ask the user for the output file name ***}
  273*C        1   WRITE (OUTPUT, 'Output File Name? ("CONSOLE:", ',
  274:C        1                  '"PRINTER:", or disc file name >> ');
  275*C        1   READLN (INPUT, Output_File_Name);
  276:S
  277:S            {*** Read the file header in order to
  278:C        1        get the directory offset ***}
  279*C        1   Read_Data_Header (File_Name, Header);
  280:S
  281:S            {*** Open the data file as a 'FILE OF Shortint'
  282:C        1        for the Read_Spectrum procedure ***}
  283*C        1   OPEN (Data_File, File_Name);
  284:S
  285:C        1   {*** Open the output file for write access ***}
  286*C        1   REWRITE (Output_File, Output_File_Name);
  287:S
  288:C        1   REPEAT
  289:C        2
  290:S              {*** Ask the user for the spectrum
  291:C        2          number that is to be tabulated ***}
  292*C        2     WRITE (OUTPUT, 'Spectrum number to tabulate?  >>');
  293*C        2     READLN (INPUT, Spec_Num);
  294:S
  295*C        2     IF (Spec_Num > 0) THEN BEGIN
  296:S
  297:C        3       {*** Read the desired spectrum from the file ***}
  298*C        3       Get_Spectrum (Data_File, Spec_Num,
  299:C        3                     Header.Info.Dir_Offset, Spectrum);
  300:S
  301:C        3       {*** Tabulate the spectrum to the output file ***}
```

```
Pascal [Rev 3.2  8/29/87] ReadSpec.TEXT            2-May-88 11:27:52 Page 8

  302*C       3       Tab_Spectrum (Spectrum, Output_File);
  303:S
  304:C       3     END;  {If Spec_Num}
  305:S
  306*C       2   UNTIL (Spec_Num <= 0);
  307:S
  308*C       1   CLOSE (Data_File);
  309*C       1   CLOSE (Output_File, 'SAVE');
  310:S
  311*C       1 END.  {Main Program}
  312:S
  313:S

No errors. No warnings.
              ***** Nonstandard language features enabled *****
```

An example As an example, this program was run on the data file DEMOSCAN.D and two spectra (#1 and the top of the first peak, #24) were tabulated, producing the following output.

```
                  Spectrum Retention Time : 4.630
                          Number of Peaks : 36
                                Base Peak : 55.60

    Mass    Abund       Mass    Abund       Mass    Abund       Mass    Abund
   41.20       21      43.90       20      44.70       14      46.80       23
   47.70       20      49.90       16      51.00       26      51.60       13
   52.80       15      53.40       15      55.60       30      56.10       23
   56.70       13      57.70       19      58.40       17      59.10       15
   59.80       19      62.70       15      63.20       14      64.80       23
   67.70       15      68.20       17      70.50       19      72.10       13
   76.50       12      76.90       14      78.10       13      78.40       13
   82.15       19      82.65       13      84.95       18      86.85       13
  103.35       17     105.85       16     118.75       11     131.25       12
```

```
                    Spectrum Retention Time : 4.803
                           Number of Peaks : 55
                                 Base Peak : 57.00

     Mass    Abund         Mass    Abund         Mass    Abund         Mass    Abund
    41.00     1584        42.10      575        43.00     2283        44.00      107
    44.00       16        45.50       18        46.40       28        50.90       27
    53.00      138        54.00       92        55.00      602        56.10      629
    57.00     2819        58.00      129        59.60       20        61.90       14
    64.20       17        65.10       23        66.90       40        67.20       43
    67.90       45        69.00      249        70.00      394        71.00     1500
    72.10       94        72.90       17        75.70       20        76.30       12
    76.50       12        78.95       21        79.75       12        80.95       14
    82.05       30        83.15       98        84.15      274        85.05      701
    86.05       78        87.25       28        92.05       13        95.95       31
    97.05       44        98.05      183        99.15      174       100.05       13
   107.15       13       111.05       21       112.05      140       112.95       93
   126.15       84       127.05       91       140.05       25       141.15       52
   142.05       15       170.15      184       171.05       30
```

Integration results file

Overall structure

Header Record 512 bytes
Processed Peak Records 3 Rec/Pk at 40 bytes each
Channel Directory Records 34 bytes each

Header record

Header records for integration results files have the same structure as the header records for mass spectral data files. The information contained in the records is changed to reflect the fact that the data have been integrated.

Peak entries

Byte Offsets							
0	Various Peak Codes (see note 1)						
2		Stop	Start	D	A	O	U
4	Record Type (see note 2)						
6	Record Dependent : Real (see note 3)						
	. . .						
12							
14	Record Dependent : Real (see note 3)						
	. . .						
20							
22	Record Dependent : Real (see note 3)						
	. . .						
28							
30	Relative Record Number : Integer						
32							
34	Previous Record : Integer						
36							
38	Next Record : Integer						
40							

Note 1:

Start & Stop Codes;		Other Peak Codes;	
00	Baseline	D	Distorted
01	Valley	A	Aborted
10	Penetration	O	Over
11	Horizontal	U	Under

Note 2:

Record Type Values

0	Extended Peak Information
1	Extended Peak Start Information
2	Extended Peak Stop Information
3	Area Slice Information
4	Front Shoulder Information
5	Rear Shoulder Information
6	Not used in Processed Peak Information
7	Not used in Processed Peak Information
8	Normal Peak Information
9	Solvent Peak Information
10	Tangent Peak Information
11	Negative Peak Information
12	Area Sum Peak Information
13	Not used in Processed Peak Information
14	Not used in Processed Peak Information

Note 3:

Type	Real 1	Real 2	Real 3
0	Width	Symmetry	Baseline
1-2	Pk Strt/Stp Time	Pk Strt/Stp Level	Baseline Strt/Stp
3	Slice Start Time	Slice Width	Slice Area
4-5	Shoulder Time	Shoulder Height	Unused
8-12	Retention Time	Peak Area	Peak Height

Channel directories

Byte Offset			
0		Ch Type	Unused
2	Channel Dependent : Real		
	. . .		
8			
10	Channel Dependent : Real		
	. . .		
16			
18	Unused		
20			
22	Relative Entry Number : Integer		
24			
26	Previous Directory Entry : Integer		
28			
30	Next Directory Entry : Integer		
32			

Access philosophy To illustrate how to access information in the integration results data files, the following program asks for the name of a integration data file and the name of the file that the tabulations will be sent to. The program then continually prints peak records from the integration file to the output file. No checks for limits are performed, so some caution should be exercised to avoid file access errors.

```
Pascal [Rev 3.2  8/29/87] IntPrt.TEXT           2-May-88 11:27:04 Page 1

         0 $SYSPROG ON$
         0 $DEBUG ON$
         0 $LINES 43$

         0 PROGRAM Int_Prt (INPUT, OUTPUT);

         1 TYPE

         1       Byte = 0 .. 255;
         1   String80 = String [80];
         1   Shortint = -32768 .. 32767;

         1   Flag_Peak_Type =
         1        (Stop_Code_A,  { Defines, 2 and 2 bits separately    }
         1         Stop_Code_B,  { any of 4 peak start and stop types  }
         1         Start_Code_A, { 00=B=Baseline,     01=V=Valley      }
         1         Start_Code_B, { 10=P=penetration,  11=H=Horizontal  }
         1         Distorted,    { set if spiked, missed readings, etc }
         1         Aborted,      { peak is aborted for whatever reason }
         1         Under,        { Peak encountered A/D underrange     }
         1         Over);        { Peak encountered A/D overrange      }

         1    Flag_Normal_Peak = SET OF Flag_Peak_Type;

         1    {-------------------------------------------------------}
         1    { For all peak types the tag byte is a constant, and    }
         1    { assumes the following values:                         }
         1    {       00 = Extended peak record;                      }
         1    {       01 = Extended peak start info;                  }
         1    {       02 = Extended peak end info;                    }
         1    {       03 = Area slice record;                         }
         1    {       04 = Front shoulder record;                     }
         1    {       05 = Rear shoulder record;                      }
         1    {       06 = Area reject record;                        }
         1    {       07 = Baseline record;                           }
         1    {       08 = Normal peak info;                          }
         1    {       09 = Solvent peak info;                         }
         1    {       10 = Tangent skimmed peak info;                 }
         1    {       11 = Negative peaks enabled info;               }
         1    {       12 = Area Summation info;                       }
         1    {       13 = Header_Info;                               }
         1    {       14 = Initialize_Info                            }
         1    {-------------------------------------------------------}
```

```
Pascal [Rev 3.2  8/29/87] IntPrt.TEXT                2-May-88 11:27:04 Page 2

   44:D         1
   45:D         1    Descriptor_Type = (Extended_Peak_Info,
   46:D         1                        Extended_Peak_Start_Info,
   47:D         1                        Extended_Peak_End_Info,
   48:D         1                        Slice_Info,
   49:D         1                        Front_Shoulder_Info,
   50:D         1                        Rear_Shoulder_Info,
   51:D         1                        Area_Reject_Info,
   52:D         1                        Baseline_Info,
   53:D         1                        Normal_Peak_Info,
   54:D         1                        Solvent_Peak_Info,
   55:D         1                        Tangent_Skim_Peak_Info,
   56:D         1                        Negative_Peak_Info,
   57:D         1                        Area_Sum_Peak_Info,
   58:D         1                        Header_Info,
   59:D         1                        Initialize_Info);
   60:D         1
   61:D         1 TGCResponseErrors = (IntegOrIdentAborted,   {bit 15}
   62:D         1                      NoSummedPeaks,         {bit 14}
   63:D         1                      NoReferencePeakFound,  {bit 13}
   64:D         1                      NoStandardPeakFound,   {bit 12}
   65:D         1                      AnalysisAborted,       {bit 11}
   66:D         1                      NumberOfPeaks_GT_Max,  {bit 10}
   67:D         1                      NoPeaksIntegrated,     {bit 9 }
   68:D         1                      EndNotOnBaseline,      {bit 8 }
   69:D         1                      Error7,                {bit 7 }
   70:D         1                      CardPoints_GT_Max,     {bit 6 }
   71:D         1                      ExcessNegativeInput,   {bit 5 }
   72:D         1                      ReadingsMissed,        {bit 4 }
   73:D         1                      ADOverrange,           {bit 3 }
   74:D         1                      ControlEventsAborted,  {bit 2 }
   75:D         1                      RunAborted,            {bit 1 }
   76:D         1                      ADProblem );           {bit 0 }
   77:D         1
   78:D         1 TGCErrorSet = Set of TGCResponseErrors;
   79:D         1
   80:D         1 TGA_Peak_Info_Record = RECORD
   81:D         1          Flag_Peak : Flag_Normal_Peak; { Should occupy 2 bytes }
   82:D         1        { Dummy : Bint; Invoked if the tag does not use 2 bytes }
   83:S
   84:D         1        CASE Peak_Type : Descriptor_Type OF
   85:S
   86:D         1          Extended_Peak_Info : (Width : Real;
  87:D        1                         Symmetry : Real;
  88:D        1                         Baseline : Real); (height units)
  89:S
  90:D        1           Extended_Peak_Start_Info : (Time_Peak_Start : Real;
  91:D        1                                       Level_Peak_Start : Real;
  92:D        1                                    Baseline_Peak_Start : Real);
  93:S
  94:D        1           Extended_Peak_End_Info : (Time_Peak_End : Real;
  95:D        1                                     Level_Peak_End : Real;
  96:D        1                                  Baseline_Peak_End : Real);
  97:S
  98:D        1           Slice_Info : (Start_Time : Real;
  99:D        1                        Slice_Width : Real;
 100:D        1                         Slice_Area : Real);
 101:S
 102:D        1           Front_Shoulder_Info : (F_Shoulder_Time : Real;
 103:D        1                                  F_Shoulder_Height : Real);
 104:S
 105:D        1           Rear_Shoulder_Info : (R_Shoulder_Time : Real;
 106:D        1                                 R_Shoulder_Height : Real);
 107:S
 108:D        1           Normal_Peak_Info : (Retention_Time : Real;
 109:D        1                                         Area : Real;
 110:D        1                                       Height : Real);
 111:S
 112:D        1           Solvent_Peak_Info : (Retention_Time_S : Real;
 113:D        1                                          Area_S : Real;
 114:D        1                                        Height_S : Real);
 115:S
 116:D        1           Tangent_Skim_Peak_Info : (Retention_Time_T : Real;
 117:D        1                                               Area_T : Real;
 118:D        1                                             Height_T : Real);
 119:S
 120:D        1           Negative_Peak_Info : (Retention_Time_N : Real;
 121:D        1                                           Area_N : Real;
 122:D        1                                         Height_N : Real);
 123:S
 124:D        1           Area_Sum_Peak_Info : (Retention_Time_A : Real;
 125:D        1                                           Area_A : Real;
 126:D        1                                         Height_A : Real);
 127:S
 128:D        1           Header_Info : (No_Records : Integer;
 129:D        1                            Run_Time : Real;
```

```
                             Errors : TGCErrorSet;
                             Number : Shortint)
       END;

  Mode_Type = (Null_Ch, Sim, Bim, Tim, Complex_Ch);

  Channel =  PACKED RECORD
             Ch_Atten : Boolean;
              Ch_Type : Mode_Type;
               L_Mass : Real;
               H_Mass : Real;
               Unused : Integer;
                     END;

  (*** Processed Peak Record ***)
  Proc_Pk_Type =   RECORD
              Pk_Info : TGA_Peak_Info_Record;
          Rel_Rec_Num : Integer;
             Prev_Rec : Integer;
             Next_Rec : Integer;
                     END;
  Proc_Pk_File_Type = FILE OF Proc_Pk_Type;

  (*** Processed Peak Channel Directory ***)
  Directory_Record_Type = RECORD
                Ch_Desc : Channel;
            Rel_Rec_Num : Integer;
              First_Rec : Integer;
               Last_Rec : Integer;
                Num_Rec : Integer;
                       END;

Directory_File_Type = FILE OF Directory_Record_Type;
Directory_Mem_Ptr = ^Directory_Mem_Type;
Directory_Mem_Type = RECORD
              Dir_Info : Directory_Record_Type;
              Next_Dir : Directory_Mem_Ptr;
                       END;

(*** File Header Types ***)
Header_info_type = RECORD
         File_num_str   : String [3];
         File_str       : String [19]; (e.g. GC/MS DATA FILE)
```

```
 173:D       1         Data_name      : String [61]; {User input name}
 174:D       1         Misc_info      : String [61]; {User input name}
 175:D       1         Operator       : String [29];
 176:D       1         Date_time      : String [29];
 177:D       1         Inst_model     : String [9];  {e.g. 5970      }
 178:D       1         Inlet          : String [9];  {e.g. GC,LC, ETC.}
 179:D       1         Method_file    : String [19];
 180:D       1         File_type      : Integer;
 181:D       1         Seq_index      : Shortint;
 182:D       1         Als_bottle     : Shortint;
 183:D       1         Replicate      : Shortint;   {per ALS index   }
 184:D       1         Dir_ent_type   : Shortint;   {type of dir ents }
 185:D       1         Dir_offset     : Integer;
 186:D       1         Data_offset    : Integer;
 187:D       1         Run_Tbl_offset : Integer;
 188:D       1         Norm_offset    : Integer;
 189:S                 Extra_records  : Shortint;   {Number of 256 BYTE
 190:D       1                                       Records following}
 191:D       1         Num_records    : Integer;    {Number of data recs}
 192:D       1         Start_rtime    : Integer;    {Starting ret time }
 193:D       1         End_rtime      : Integer;    {Last retention time}
 194:D       1         Max_signal     : Integer;    {Maximum dir signal }
 195:D       1         Min_signal     : Integer     {Minimum dir signal }
 196:D       1                      END;
 197:S
 198:D       1     Header_Rec_Type = PACKED RECORD
 199:D       1                   Info : Header_info_type;
 200:D       1                 Filler : PACKED ARRAY
 201:D       1                          [1 .. (512 -
 202:D       1                         SIZEOF (Header_info_type))] OF Byte;
 203:D       1                       END;
 204:D       1     Header_File_Type = FILE OF Header_Rec_Type;
 205:S
 206:D       1     {*** 'Full' Peak record, i.e. all possible information ***}
 207:D       1     Peak_Rec =             RECORD
 208:S
 209:D       1            {*** Channel Information ***}
 210:D       1                 Ch_Str : String [9];
 211:D       1               Low_Mass : Real;
 212:D       1              High_Mass : Real;
 213:S
 214:D       1         {*** Normal Peak Information ***}
 215:D       1               Ret_Time : Real;
```

```
216:D        1                       Area : Real;
217:D        1                     Height : Real;
218:D        1                      Codes : String [5];
219:S
220:D        1               {*** Extended Information ***}
221:D        1                      Width : Real;
222:D        1                   Symmetry : Real;
223:D        1                   Baseline : Real;
224:S
225:D        1            {*** Extended Start Information ***}
226:D        1                 Start_Time : Real;
227:D        1             Start_Baseline : Real;
228:D        1                Start_Level  : Real;
229:S
230:D        1            {*** Extended Stop Information ***}
231:D        1                   End_Time : Real;
232:D        1               End_Baseline : Real;
233:D        1                  End_Level  : Real;
234:D        1                           END;
235:S
236:D        1   Marker_Type = (Base_Line, Valley, Penetration, Horizontal);
237:D        1
238:D        1   Boundary_Type = (Start, Stop);
239:S
240:D        1   Descriptor_Set = SET OF Descriptor_Type;
241:S
242:D        1 CONST
243:S
244:D        1   Primary_Peak_Info = Descriptor_Set [Normal_Peak_Info,
245:D        1                                       Solvent_Peak_Info,
246:D        1                                       Tangent_Skim_Peak_Info,
247:D        1                                       Area_Sum_Peak_Info];
248:D        1
249:D        1 VAR
250:S
251:D  -512  1             Header : Header_Rec_Type;
252:D  -516  1            Ch_Head : Directory_Mem_Ptr;
253:D  -520  1            Ch_Tail : Directory_Mem_Ptr;
254:D  -602  1           Temp_Str : String80;
255:D  -606  1           Next_Pos : Integer;
256:D  -610  1        Current_Dir : Directory_Mem_Ptr;
257:S
258:D  -738  1               Peak : Peak_Rec;
```

```
259:D   -742  1             Index : Integer;
260:D   -784  1           Proc_Pk : Proc_Pk_Type;
261:D  -1448  1         Text_File : Text;
262:D  -1449  1        First_Time : Boolean;
263:D  -1454  1       Current_Rec : Integer;
264:D  -2158  1      Proc_Pk_File : Proc_Pk_File_Type;
265:D  -2240  1     Int_File_Name : String80;
266:D  -2322  1  Output_File_Name : String80;
267:S
268:D  -2322  1 {---------------------------------------------------------}
269:D         1 PROCEDURE Add_Dir_to_List (Dir_Ptr : Directory_Mem_Ptr);
270:S
271:C         2 BEGIN {Add_Dir_to_List}
272:S
273:C         2   {*** Initialize Entry ***}
274*C         2   Dir_Ptr^.Next_Dir := NIL;
275:C         2
276:C         2   {*** Check if First Entry ***}
277*C         2   IF (Ch_Head <> NIL) THEN BEGIN
278:C         3
279:C         3     {*** Attach End of Directory List to New Entry ***}
280*C         3     Ch_Tail^.Next_Dir := Dir_Ptr;
281:C         3
282:C         3     {*** Adjust Directory List Tail Ptr ***}
283*C         3     Ch_Tail := Ch_Tail^.Next_Dir;
284:C         3
285:C         3   END ELSE BEGIN
286:C         3
287*C         3     Ch_Head := Dir_Ptr;
288*C         3     Ch_Tail := Dir_Ptr;
289:C         3
290:C         3   END;  {First Entry}
291:C         2
292*C         2 END;  {Add_to_Dir_to_List}
293:S
294:D  -2322  1 {---------------------------------------------------------}
295:D         1 PROCEDURE Set_Up_File (VAR Proc_Pk_File : Proc_Pk_File_Type;
296:D         2                        VAR    File_Name : String);
297:D         2
298:D         2 VAR
299:S
300:D     -4  2     PP_Start_Rec : Integer;
301:D     -8  2    Dir_Start_Rec : Integer;
```

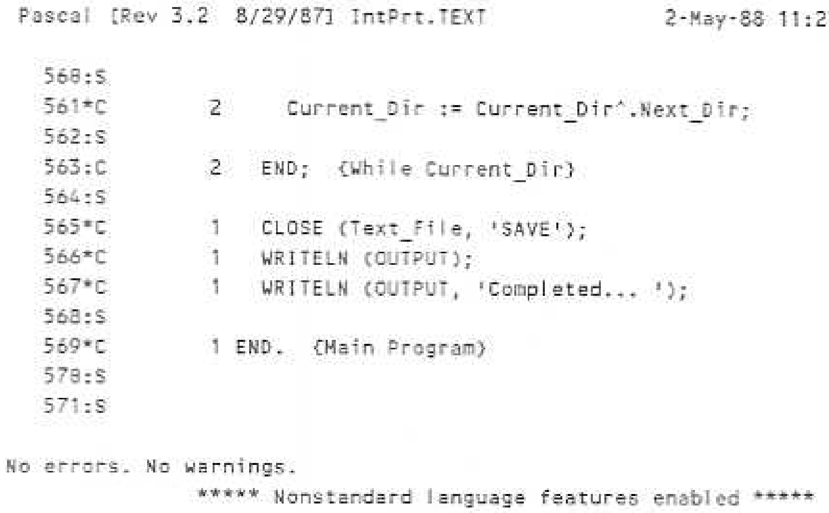
```
302:S
303:D   -46  2          Dir_Rec : Directory_Record_Type;
304:D   -50  2          Dir_Ptr : Directory_Mem_Ptr;
305:D   -54  2         Next_Pos : Integer;
306:D -1228  2      Header_File : Header_File_Type;
307:D -1928  2   Directory_File : Directory_File_Type;
308:D -1928  2
309:C        2 BEGIN {Set_Up_File}
310:S
311*C        2   Ch_Head := NIL;
312*C        2   Ch_Tail := NIL;
313:S
314:C        2   {*** Open File as Header, Extract Record Positions ***}
315*C        2   OPEN (Header_File, File_Name);
316*C        2   READ (Header_File, Header);
317:S
318*C        2   WITH Header.Info DO BEGIN
319*C        3     PP_Start_Rec := Data_Offset;
320*C        3     Dir_Start_Rec := Dir_Offset;
321:C        3   END;  {With Header}
323*C        2   CLOSE (Header_File, 'SAVE');
324:S
325*C        2   OPEN (Directory_File, File_Name);
326*C        2   SEEK (Directory_File, Dir_Start_Rec);
327:S
328*C        2   WHILE NOT EOF (Directory_File) DO BEGIN
329*C        3     READ (Directory_File, Dir_Rec);
330*C        3     NEW (Dir_Ptr);
331*C        3     WITH Dir_Ptr^ DO BEGIN
332*C        4       Dir_Info := Dir_Rec;
333*C        4       WITH Dir_Info.Ch_Desc DO BEGIN
334*C        5         L_Mass := (ROUND (L_Mass * 20)) / 20;
335*C        5         H_Mass := (ROUND (H_Mass * 20)) / 20;
336:C        5       END;  {With Dir_Info}
337:C        4     END;  {With Dir_Ptr}
338*C        3     Add_Dir_to_List (Dir_Ptr);
339:C        3   END;  {While Not EOF}
340:S
341*C        2   CLOSE (Directory_File);
342*C        2   OPEN (Proc_Pk_File, File_Name);
343:S
344*C        2 END;  {Set_Up_File}
```

```
345:S
346:D -2322  1 {----------------------------------------------------------}
347:D        1 FUNCTION Boundary_Condition (Flag : Flag_Normal_Peak;
348:D        2                          Boundary : Boundary_Type)
349:D        2                                   : Marker_Type;
350:S
351:S
352:C        2 BEGIN {Boundary_Condition}
353:S
354*C        2   CASE Boundary OF
355*C        3     Start : IF (Start_Code_A IN Flag) AND
356:C        4                (Start_Code_B IN Flag) THEN
357*C        4               Boundary_Condition := Horizontal
358:C        4             ELSE
359*C        4               IF (Start_Code_A IN Flag) THEN
360*C        5                 Boundary_Condition := Penetration
361:C        5               ELSE
362*C        5                 IF (Start_Code_B IN Flag) THEN
363*C        6                   Boundary_Condition := Valley
364:C        6                 ELSE
365*C        6                   Boundary_Condition := Base_Line;
366:C        3
367*C        3     Stop : IF (Stop_Code_A IN Flag) AND
368:C        4               (Stop_Code_B IN Flag) THEN
369*C        4              Boundary_Condition := Horizontal
370:C        4            ELSE
371*C        4              IF (Stop_Code_A IN Flag) THEN
372*C        5                Boundary_Condition := Penetration
373:C        5              ELSE
374*C        5                IF (Stop_Code_B IN Flag) THEN
375*C        6                  Boundary_Condition := Valley
376:C        6                ELSE
377*C        6                  Boundary_Condition := Base_Line;
378:C        3
379:C        3   END;
380:C        2
381*C        2 END;  {Boundary_Condition}
382:S
383:D -2322  1 {----------------------------------------------------------}
384:D        1 PROCEDURE Extract_Codes (VAR Proc_Pk : Proc_Pk_Type;
385:D        2                          VAR Pk_Type : String);
386:S
387:D        2 VAR
```

```
388:S
389:D     -2  2   Index : Boundary_Type;
390:S
391:C         2 BEGIN {Extract_Pk_Info}
392:C         2
393*C         2   WITH Proc_Pk.Pk_Info DO
394*C         3     IF Peak_Type IN Primary_Peak_Info THEN BEGIN
395:S
396*C         4       Pk_Type := '';
397*C         4       FOR Index := Start TO Stop DO
398*C         5         CASE Boundary_Condition (Flag_Peak, Index) OF
399*C         6           Base_Line : Pk_Type := Pk_Type + 'B';
400*C         6           Valley : Pk_Type := Pk_Type + 'V';
401*C         6           Penetration : Pk_Type := Pk_Type + 'P';
402*C         6           Horizontal  : Pk_Type := Pk_Type + 'H';
403:C         6         END;
404:C         4
405*C         4       CASE Peak_Type OF
406*C         5         Normal_Peak_Info : Pk_Type := Pk_Type + ' ';
407*C         5         Solvent_Peak_Info : Pk_Type := Pk_Type + 'S';
408*C         5         Tangent_Skim_Peak_Info : Pk_Type := Pk_Type + 'T';
409*C         5         Area_Sum_Peak_Info : Pk_Type := Pk_Type + '+';
410:C         5         OTHERWISE
411:C         5       END;  {Case Peak_Type}
412:S
413:C         4     END;  {If_Pk_Type}
414:S
415*C         2 END;  {Extract_Pk_Info}
416:S
417:D -2322   1 {------------------------------------------------------------}
418:D         1 PROCEDURE Write_Peak_to_File (VAR Text_File : Text;
419:D         2                               VAR      Peak : Peak_Rec);
420:S
421:C         2 BEGIN {Write_Peak_to_File}
422:S
423*C         2   WITH Peak DO BEGIN
424*C         3     WRITELN (Text_File, Ret_Time:0:3, Ch_Str:10, ' ',
425:C         3              Low_Mass:0:2, ' to ', High_Mass:0:2,
426:C         3              '  Codes: ', Codes:5,
427:C         3              '  A: ', Area:0:0, '  Ht: ', Height:0:0);
428:S
429:S                 {*** These features are not currently enabled
430:S                      they will contain values of zero, therefore
431:C        3               don't print them ***}
432:S                   {***      Width:5:3,
433:S                          Symmetry:5:3,
434:C        3                 Baseline:8:0,   ***}
435:S
436*C        3     WRITELN (Text_File,
437:C        3             '  Start Time: ', Start_Time:8:3,
438:C        3       '    Start Baseline: ', Start_Baseline:8:0,
439:C        3          '    Start Level: ', Start_Level:8:0);
440:S
441*C        3     WRITELN (Text_File,
442:C        3             '    End Time: ', End_Time:8:3,
443:C        3       '      End Baseline: ', End_Baseline:8:0,
444:C        3          '      End Level: ', End_Level:8:0);
445:S
446:C        3   END;  {With Peak}
447:S
448*C        2   WRITELN (Text_File);
449:S
450*C        2 END;  {Write_Peak_to_File}
451:S
452:D  -2322  1 {-----------------------------------------------------------}
453:D  -2322  1 {------------------  Main Program  -------------------------}
454:D  -2322  1 {-----------------------------------------------------------}
455:C        1 BEGIN {Main Program}
456:C        1
457:C        1   {*** Ask the user for the integration results file ***}
458*C        1   WRITE (OUTPUT, 'Integration results file? >> ');
459*C        1   READLN (INPUT, Int_File_Name);
460:S
461:C        1   {*** Ask the user for the output destination file ***}
462*C        1   WRITE (OUTPUT, 'Output file name? ',
463:C        1                  '("CONSOLE:", "PRINTER:", or file name) >> ');
464*C        1   READLN (INPUT, Output_File_Name);
465*C        1   REWRITE (Text_File, Output_File_Name);
466:S
467*C        1   Index := 0;
468*C        1   Set_Up_File (Proc_Pk_File, Int_File_Name);
469*C        1   First_Time := True;
470*C        1   Current_Dir := Ch_Head;
471:S
472*C        1   WHILE (Current_Dir <> NIL) DO BEGIN
473:S
```

```
2      WITH Current_Dir^.Dir_Info.Ch_Desc DO BEGIN
3        CASE Ch_Type OF
4          TIM : Peak.Ch_Str := 'TOTAL ION';
4          BIM : Peak.Ch_Str := 'ION RANGE';
4          SIM : Peak.Ch_Str := 'SINGL ION';
4          OTHERWISE Peak.Ch_Str := '          ';
4        END;  {With Current_Dir}
3        Peak.Low_Mass := L_Mass;
3        Peak.High_Mass := H_Mass;
3      END;  {With Current_Dir}

2      Current_Rec := Current_Dir^.Dir_Info.First_Rec;

2      WHILE (Current_Rec > 0) DO BEGIN

3        SEEK (Proc_Pk_File, Current_Rec);
3        READ (Proc_Pk_File, Proc_Pk);

3        WITH Proc_Pk.Pk_Info DO

4          CASE Peak_Type OF
5            Extended_Peak_Info :
5                 BEGIN
             {*** This record type is not currently
5                 enabled, these values will be zero ***}
5                   Peak.Width := Width;
5                   Peak.Symmetry := Symmetry;
5                   Peak.Baseline := Baseline;
5                 END;

5            Extended_Peak_Start_Info :
5                 BEGIN
5                   Peak.Start_Time := Time_Peak_Start;
5                   Peak.Start_Baseline := Baseline_Peak_Start;
5                   Peak.Start_Level := Level_Peak_Start;
5                 END;

5            Extended_Peak_End_Info :
5                 BEGIN
5                   Peak.End_Time := Time_Peak_End;
5                   Peak.End_Baseline := Baseline_Peak_End;
5                   Peak.End_Level := Level_Peak_End;

```

```
 517:S                          {*** Last record for this peak,
 518:C        5                      so print full peak entry ***)
 519*C        5                 Write_Peak_to_File (Text_File, Peak);
 520:C        5               END;
 521:S
 522:C        5          Normal_Peak_Info,
 523:C        5          Solvent_Peak_Info ,
 524:C        5          Tangent_Skim_Peak_Info ,
 525:C        5          Negative_Peak_Info ,
 526:C        5          Area_Sum_Peak_Info :
 527:C        5               BEGIN
 528:S
 529:C        5                 {*** Initialize variables for next peak ***}
 530*C        5                 Peak.Width := 0;
 531*C        5                 Peak.Symmetry := 0;
 532*C        5                 Peak.Baseline := 0;
 533*C        5                 Peak.Start_Time := 0;
 534*C        5                 Peak.Start_Baseline := 0;
 535*C        5                 Peak.Start_Level  := 0;
 536*C        5                 Peak.End_Time := 0;
 537*C        5                 Peak.End_Baseline := 0;
 538*C        5                 Peak.End_Level  := 0;
 539*C        5                 Peak.Codes := '';
 540*C        5                 Peak.Ret_Time := 0;
 541*C        5                 Peak.Area := 0;
 542*C        5                 Peak.Height := 0;
 543:S
 544*C        5                 Extract_Codes (Proc_Pk, Peak.Codes);
 545*C        5                 Peak.Ret_Time := Retention_Time;
 546*C        5                 Peak.Area := Area;
 547*C        5                 Peak.Height := Height;
 548:C        5               END;
 549:S
 550:C        5          OTHERWISE
 551:C        5             {*** Do Nothing ***}
 552:C        5        END;  {Case Flag_Pk}
 553:S
 554*C        3      IF (Current_Rec <> Current_Dir^.Dir_Info.Last_Rec) THEN
 555*C        4        Current_Rec := Current_Rec + 1
 556:C        4      ELSE
 557*C        4        Current_Rec := 0;
 558:S
 559:C        3    END;  {While Current_Rec}
```



```
 Pascal [Rev 3.2  8/29/87] IntPrt.TEXT            2-May-88 11:27:04 Page 14

   560:S
   561*C        2     Current_Dir := Current_Dir^.Next_Dir;
   562:S
   563:C        2   END;  {While Current_Dir}
   564:S
   565*C        1   CLOSE (Text_File, 'SAVE');
   566*C        1   WRITELN (OUTPUT);
   567*C        1   WRITELN (OUTPUT, 'Completed... ');
   568:S
   569*C        1 END.  {Main Program}
   570:S
   571:S

No errors. No warnings.
              ***** Nonstandard language features enabled *****
```

As an example, this program was run on the integration results data file BPU.I producing the following output.

```
6.164 SINGL ION 104.00 to 104.00  Codes:   BB   A: 828  Ht: 32
  Start Time:    6.094    Start Baseline:        9    Start Level:        0
    End Time:    6.391      End Baseline:        9      End Level:        0

6.160 SINGL ION 105.00 to 105.00  Codes:   BB   A: 3210  Ht: 135
  Start Time:    6.080    Start Baseline:        6    Start Level:        0
    End Time:    6.363      End Baseline:        5      End Level:        0
```