Project Report

Research in Head Protection in the Industrial Environment

Supported in part by
the Safety Equipment Institute

Principal Investigators:

Jacqueline G. Paver and James H. McElhaney
Department of Biomedical Engineering
Duke University
Durham, N.C. 27706

October 1987

Research in Head Protection in the Industrial Environment

Research in Head Protection in the Industrial Environment

A. Summary

The goal of this project was to develop a standard for industrial helmets for use in environments that require a higher level of lateral protection than provided by current industrial helmets. To accomplish this goal and demonstrate feasibility, research and development efforts were performed to develop justification for the various test specifications from the available literature and to develop a helmet test system as described in the proposed standard.

The literature on head and neck injury criteria was assembled and studied to establish test specifications. An extensive review of the relevant literature indicates that 1000 pounds is a reasonable criteria for vertebral body collapse due to axial compression of the cervical spine. A consensus also indicates that the Severity Index (SI) or Head Injury Criteria (HIC) is a reasonable indicator of the potential for head injury due to frontal impacts. The literature that exists on side or rear head impacts is not definitive. It is clear that the side of the head is more susceptible to impact injury than the front. A universal head injury criteria does not exist at this time and its development and validation may take several years. However, the HIC is widely accepted in the automotive industry and it is probably a valid measure of comparative performance. It is therefore the recommended criteria for energy management in the proposed standard. A data acquisition system, which utilizes an IBM or compatible PC, has been developed. The software required to calculate the HIC is provided as part of this report.

A helmet test system as required in the proposed ANSI Z89.2 Standard was also designed and constructed. The test methodology was developed. The proposed test simulates the forces and accelerations from lateral impacts by dropping a helmeted headform in guided fall upon flat and hemispherical steel anvils. The acceleration-time history of

the headform is measured and the HIC calculated. The proposed standard requires that the HIC be less than 1000.

Experimental studies were conducted (1986 Report) to evaluate and compare the performance of various helmet types to impacts in different directions. Two basic impactor shapes were used, flat and hemispherical. The helmet types tested included typical industrial, motorcycle, bicycle, football, baseball, jailai, and jockey helmets.

A majority of the world's helmet standards were reviewed (1986 Report) to document the "state-of-the-art." Two basic impact test methods were recognized. Either an impactor is dropped on a helmeted headform or the helmeted headform is dropped onto a test block. Measured responses are typically transmitted force or headform acceleration. Performance criteria are typically peak force, peak g's, or some formulae combining headform acceleration with time (i.e., SI or HIC). Helmets that rely on a peak g criteria are generally stiffer than helmets that use peak force or a head injury criteria. Based on this study, it is recommended that the current ANSI Z89.1 Standard be used to establish minimum energy management for falling objects striking the top of the helmet, penetration resistance, and protection from electrical shock. To establish lateral impact protection performance, a drop test of a helmeted headform onto a test block is proposed. Since the ANSI Z89.1 energy level has resulted in comfortable helmets with an excellent record of protection from falling objects, the same energy level is proposed for lateral impacts.

A study of the effect of headform size on helmet performance was made. It was concluded that while the influence is small, it is possible that performance differs with headform size. Three ISO headforms are recommended to accommodate the range of potential users, 5th percentile females through 95th percentile males.

There are many helmet standards that include retention system tests. Invariably, these are tests of the strength of the retention system. A typical test would be a static or dynamic pull of the helmet against a headform. By their nature, these tests require a chin strap. While this type of test is design restrictive, it is reliable and repeatable. Our

efforts to develop a more performance-oriented test were unsuccessful. The effects of head geometry friction and local pressure intensity provide too many uncontrollable variables for a repeatable test. A dynamic pull test on a chin strap is proposed as a strength specification of the retention system.

A study of the effect of projections on energy management was performed. It was concluded that projections less than five millimeters had minimal effect on the transmitted force and pressure distribution provided they are covered with an adequate thickness of helmet liner. This conclusion is reflected in the proposed standard.

A study of conditioning temperature and water immersion requirements of other standards was made. The requirements proposed are comparable to current motorcycle, bicycle, and football helmet standards.

Based on these efforts, a prototype standard entitled, "ANSI Z89.2 Protective Headgear for Industrial Environments Requiring Protection from Lateral Impacts," was produced. The equipment and instrumentation requirements were outlined and a proven system described. The next phase of this activity will probably be a careful subcommittee review and revision of the proposed standard into a form appropriate for a full committee review.

B. Development of a Data Acquisition and Analysis System

A data acquisition and analysis system has been developed for the Lateral Impact Energy Attenuation Test. This system includes SI and HIC software for an IBM PC or PC-compatible microcomputer.

B.1. Hardware: The ISC-16 Data Acquisition System

The COMPUTERSCOPE-IND ISC-16 Data Acquisition package consists of a 16-channel A/D board, external Instrument Interface box, and Scope Driver software. In combination, the ISC-16 Data Acquisition package offers a 1MHz aggregate sampling rate capability with 12 bit resolution at input voltages within the range of -10 to +10 volts. Fully automated keystroke commands provide the user with fast and effective control over all features of the ISC-16, including channel selection, trigger control (external or any channel, +/- level or slope), sampling rate and memory buffer size (1K to 64K words). The Scope Driver software employs a ring buffer design, allowing the capture of data in pre-trigger intervals of virtually any length. A variety of waveform manipulation commands are incorporated, including time-base expansion and contraction, left and right scrolling, independent vertical gain adjustment, vertical cursors to measure latency and amplitude differences between any two data points, and waveform storage and retrieval. A strip-chart format is available allowing 64K words of storage to be output in a continuous record. Finally, hard copy output is available in 10 bit resolution on a dot matrix printer or HP plotter.

B.2. Software

B.2.1 The Severity Index and Head Injury Criterion

The Severity Index (SI) and the Head Injury Criterion (HIC) are analytical tools for assessing head injury potential. Although some controversy exists as to the validity of these criteria (Versace, Goldsmith), the SI and HIC are used extensively in the helmet and automotive industries. Application of these criteria requires knowledge of head acceleration-time histories.

The Severity Index is defined by the equation:

$$\mathrm{SI} = \int_{t_{begin}}^{t_{final}} \mathrm{a(t)}^{2.5}\mathrm{dt} \tag{1}$$

where $a(t)$ is the resultant acceleration profile of the head center of gravity, in g units and t_{begin} and t_{final} are beginning and ending pulse times, in seconds. The HIC is defined in the Federal Motor Vehicle Safety Standard 208 (FMVSS 208) by the expression:

$$\mathrm{HIC} = (\mathrm{t_2} - \mathrm{t_1}) \left[\frac{1}{\mathrm{t_2} - \mathrm{t_1}} \int_{t_1}^{t_2} \mathrm{a(t)}\ \mathrm{dt} \right]^{2.5} \tag{2}$$

where $a(t)$ is the resultant acceleration profile of the head center of gravity, in g units, and t_1 and t_2 are the two points in time which maximize the HIC, in seconds. An SI or HIC value over 1000 is deemed injurious.

B.2.2 Integration Algorithms

A significant amount of the computation time required to calculate the SI and HIC is devoted to numerical integration. For this reason, it is important to maximize the efficiency of the algorithms used. Several methods of numerical integration exist. The simplest is the Trapezoidal Rule, which approximates integrals by straight lines connecting neighboring data points:

$$\int_a^b f(x)dx \approx \Delta x\left(\frac{1}{2}\, y_0 + y_1 + y_2 + \cdots + y_{n-1} + \frac{1}{2}\, y_n\right) \tag{3}$$

A more accurate method of integration is Simpson's Rule, which approximates integrals as quadratics over intervals of three data points:

$$\int_a^b f(x)dx \approx \frac{\Delta x}{3}\left(y_0 + 4y_1 + 2y_2 + 4y_3 + \cdots + 4y_{n-1} + y_n\right) \tag{4}$$

The quadratic leads to a more precise answer than connecting points by lines. It is possible to extend each approximated interval, such as third-order equations approximating every interval of four points, fourth-order equations approximating every interval of five points, etc. There also exists an algorithm that uses the trapezoidal rule and Simpson's rule to arrive at a final value for the integral that is much more accurate than either method individually. Richardson extrapolation works by examining the result from the trapezoidal rule, calculating how much the result is 'improved' by Simpson's rule, and then extrapolating to an even more accurate value.

Trapezoidal integration has been implemented for two reasons: (1) its simplicity of logic, and (2) its "self-containedness." That is, with trapezoidal integration, only the two points which define the interval must be utilized for the calculation. With the other methods, the curve fitting requires that the neighboring points on each side of the interval be utilized.

B.2.3 Numerical Evaluation of the HIC

Numerical evaluation of the HIC requires analog-to-digital conversion of the acceleration-time history, using a sampling rate sufficient to characterize the pulse accurately. Sampling rates of $1-10$ KHz are typically used in automotive crash testing, where the acceleration pulse durations generally range from $50-200$ ms (Chou and Nyquist). The SAE Recommended Standard J885a states that, for a HIC calculation, time increments of 1 ms are adequate. In helmet testing, the acceleration pulse durations generally range from $5-30$ ms. Compliance with the SAE Recommmended Standard J211b Channel Class 1000 is required (i.e., an 8 KHz sampling rate).

Let $a(t)$ be an analog resultant acceleration profile. Suppose that this analog pulse is divided into m equal intervals of width dt such that $a^{**}(t)$ is the discretized representation of $a(t)$. Then, equation (2) can be rewritten in the form:

$$HIC = \max_{0 \le i < j \le m} \left\{ (t_j - t_i) \left[\frac{\int_{t_i}^{t_j} a^{**}(t)dt}{t_j - t_i} \right]^{2.5} \right\} \tag{5}$$

Implementation of the algorithm in equation (5) can be a tedious process, since the maximization process requires an double loop on (i, j), with each loop performing m iterations.

Several improvements were made to this algorithm. These reduced the number of calculations required for a particular size input data set, reduced the size of the stored data arrays, and decreased computation time. Three distinct methods were implemented to calculate HIC values: a brute force method, a simple partial sums technique, and the partial sums/sliding endpoints method.

The first improvement to the algorithm in equation (5) was to remove the actual integration from inside the loop. This modification, which was common to all three methods, was based on recognition of the fact that the same m "strips of area" were calculated over and over again. It is computationally inefficient to calculate every area every time it is needed.

Using Trapezoidal Integration, the area under the curve between point i and point $i+1$ is defined as:

$$\int_{t_i}^{t_{i+1}} a^{**}(t)dt = \frac{(a_i + a_{i+1})}{2} \cdot dt \tag{6}$$

So, if $\overline{a_i}$, the average height, is defined as:

$$\overline{a_i} = \frac{\int_{t_i}^{t_{i+1}} a^{**}(t)dt}{dt} \tag{7}$$

then

$$\overline{a_i} = \frac{(a_i + a_{i+1})}{2} \tag{8}$$

The area under the curve between point i and j is:

$$\sum_{k=i}^{j-1} \overline{a_k} = \frac{\int_{t_i}^{t_j} a^{**}(t)dt}{dt} \tag{9}$$

or

$$\int_{t_i}^{t_j} a^{**}(t)dt = dt \sum_{k=i}^{j-1} \overline{a_k} \tag{10}$$

Equation (5) may be rewritten in the form:

$$HIC = \max_{0 \le i < j \le m} \left\{ (t_j - t_i) \left[\frac{dt \sum_{k=i}^{j-1} \overline{a_k}}{(t_j - t_i)} \right]^{2.5} \right\} \tag{11}$$

The fact that the discretization of the analog pulse uses a constant time step dt implies that

$$t_j - t_i = dt \cdot (j - i) \tag{12}$$

So, equation (10) can be rewritten:

$$\int_{t_i}^{t_j} a^{**}(t)dt = \frac{(t_j - t_i) \sum_{k=i}^{j-1} \overline{a_k}}{(j - i)} \tag{13}$$

and Equation (11) becomes:

$$HIC = dt \cdot \max_{0 \le i < j \le m} \left\{ (j - i) \left[\frac{\sum_{k=i}^{j-1} \overline{a_k}}{(j - i)} \right]^{2.5} \right\} \tag{14}$$

For programming purposes, let $a^*(t)$ be defined by the following equation:

$$a^*(t) = \frac{\sum_{k=i}^{j-1} \overline{a_k}}{(j - i)} \tag{15}$$

and let h_{max} be defined as the quantity, in braces, that is maximized:

$$h_{max} = (j - i) \left[\frac{\sum_{k=i}^{j-1} \overline{a_k}}{(j - i)} \right]^{2.5} \tag{16}$$

Then, Equation (14) can be rewritten:

$$HIC = (dt) \cdot \max_{0 \le i < j \le m} \left[(j-i)(h_{max}) \right] \tag{17}$$

or

$$HIC = (t_2 - t_1)(h_{max}) \tag{18}$$

where t_1 and t_2 are the two points in time which maximize the HIC such that

$$t_2 = dt \cdot j \tag{19}$$

$$t_1 = dt \cdot i \tag{20}$$

The original storage scheme consisted of two matrices: a (1 x m) matrix, which held the resultant acceleration data points, and a (m x m) matrix, which held the average heights. Since this matrix was half-empty (i.e., upper-triangular), due to the fact that the average height from point 1 to point 6 was the same as from 6 to 1, only the average heights from a common origin were stored. That is, the matrix was reduced to a (1 x m) matrix, which held the average heights from point 0 to point m. Clearly, it is more computationally efficient to store only $\overline{a_i}$ or $\Sigma_{k=i}^{j-1} \overline{a_k}$ rather than the actual acceleration data, since the actual acceleration data was not used after the average height calculation. For this reason, in Method 1, this average height is stored in an array called H. In Methods 2 and 3, these average heights are summed and stored, in an array called H, such that $H(0)$ is 0, $H(1)$ is the average height between points 0 and 1, $H(2)$ is the average height between points 0 and 2, etc. The final memory necessary is a (1 x m) array and the calculation required to find the average height between any two points is one subtraction.

B.2.3.1 Brute Force Method

The brute force method was implemented first. For all i's and j's, where $0 \leq i < m$ and $i < j \leq m$, the integral in equation 1 was evaluated by first storing the average heights in H and, then, summing these values for every interval of every length. The maximization process required comparison of h_{max} for every interval of every length.

This method is obviously the simplest and the slowest. For 1000 data points, a half million intervals must be examined. This is calculated as follows: the shortest interval possible is 2 consecutive points. One thousand of these intervals exist. The longest interval possible is the entire range of points, only one of which exists. Therefore, 1000 different lengths are possible and the average interval (i.e., 500 data points) exists 500 times. This totals a half million intervals. This is expressed in computer jargon as an $\frac{n^2}{2}$ algorithm, where n is the total number of data points. If the actual acceleration data had been stored and the integration performed at each step, the algorithm would be even less efficient (i.e., an $\frac{n^3}{4}$ algorithm requiring a quarter billion sequences of calculations).

B.2.3.2 The Partial Sums Method

The modification that distinguishes the Partial Sums Method from the Brute Force Method is the removal of the loop on k from inside the loop on (i, j). This was accomplished in the program by summing the average heights and storing the sums in the H array. Because of this second modification, the sum

$$\sum_{k=i}^{j-1} \overline{a_k}$$

can be found by a single subtraction. Since the cumulative average heights have been stored in H, the average height between the ith and jth points is the average height from 0 to the jth point minus the average height from 0 to the ith point, or simply $H(j) - H(i)$. That is, let

$$H(p) = \sum_{k=0}^{p} \overline{a_k} \tag{21}$$

then

$$\sum_{k=i}^{j-1} \overline{a_k} = H(j) - H(i) \tag{22}$$

So, equation (15) can be rewritten as:

$$HIC = dt \cdot \max_{0 \le i < j \le m} \left\{ (j-i) \left[\frac{H(j) - H(i)}{(j-i)} \right]^{2.5} \right\} \tag{23}$$

This method is far more efficient since most of the intervals overlap other intervals, where the integral may have already been calculated. For example, the partial sum for the 500th data point is the sum of all the areas under the first 500 data points. The partial sum for the 501st data point is equal to the sum of the area under the curve from point 500 to point 501 plus the partial sum for the 500th point. The computational benefits are evident when, for example, the integral from the 500th point to the 1000th point is calculated by subtracting the partial sum of the 500th point from the partial sum of the 1000th point. For 1000 data points, 1000 partial sums are calculated. The remaining integrals are calculated for the remaining $\frac{n^2}{2}$ intervals from these 1000 partial sums by a simple subtraction. Therefore, $\frac{n^2}{2} + n$ (i.e. a half million calculation sequences) are required for this integration.

B.2.3.3 The Partial Sums/Sliding Endpoints Method

The third method, utilizing partial sums and sliding endpoints, reduces the number of calculation sequences to $2n$ (i.e., only 2000 calculation sequences for 1000 data points). This method is based upon the observation that the HIC for an interval is basically the integral over that interval divided by the length of the interval. Thus, the longer the interval, the larger the integral and the larger the number that is divided into the integral.

For the HIC calculation, the goal is to find two points t_1 and t_2 such that the HIC value is maximized. In the partial sums/sliding endpoints method, t_2 is started at t_{max} and slid left until a maximum is found. At that point, t_1 is started at t_0 and slid right until there is a maximum between t_1 and t_2. After these endpoints are found, the sliding endpoints must be repeatedly alternated, moved towards the center of the pulse, and the HIC reevaluated until the maximum HIC is found for both endpoints.

For example, let t_1 be the 1st slide of t_1, let t_{1_2} be the second slide of t_1, etc. First, slide t_2 left until a maximum HIC is found, corresponding to:

$$\int_{t_{1_0}}^{t_{2_1}} \tag{25}$$

Next, slide t_1 right until a maximum HIC is found, corresponding to:

$$\int_{t_{1_1}}^{t_{2_1}} \tag{26}$$

Then, slide t_2 left again until a maximum HIC is found, corresponding to:

$$\int_{t_{1_1}}^{t_{2_2}} \tag{27}$$

This procedure continues until the movement of an endpoint results in a decrease in the HIC, rather than an increase.

B.2.4 Computer Programs

(1) HICCNVRT.PAS (source)

HICCNVRT.EXE (executable)

Converts Computerscope files to ASCII files for HIC_BAS and HIC_C versions.

(2) HIC_BAS.BAS

Program to calculate SI and HIC, written in BASIC.

Note: Must run HICCNVRT first if using Computerscope data.

(3) HIC_C.C (source)

HIC_C.EXE (executable)

Program to calculate the SI and HIC, written in C.

Note: Must run HICCNVRT first if using Computerscope data.

(4) HIC_PAS.PAS (source)

HIC_PAS.EXE (executable)

Program to calculate the SI and HIC, written in PASCAL.

Note: Computerscope data conversion built in.

Variables in ISCCNVRT and HICCNVRT

Filenames:

Source	input .PRM file
DtaSource	input .DAT file
Dest	array of 8 output files .CH_ (I) output file .RES (H)
DestAsc	array of ASCII output files (H)

Arrays:

Buffer	input array of bytes from file
H	array for storage of partial sums (H)
Ave	array for ave calc for each channel
Cal	array for cal factor for each channel
Volts	array for point storage for each chan (H)
Max, Min	array of reals for storing max and min (I)
Channel_Status, Channel_Name, Threshold Trigger_Thrshld, Voltage_Offset Voltage_Scale, Channel_Enable, Display_Enable	Input parameters for each channel

Strings:

Filenm	Main file name for input and output
Chan	Extension for output file name
Offset	Answer of question of removing offset
Hold	Multipurpose dummy string
Disp_s, Ch_num, Vc_lf, Vc_rt, H_scl Title, Time, Time_mode, Size, Length Current_block, Current_point, Buffer_size Current_Block_MSB	Input strings for parameters

Reals:

Volts	Calculated output value in user units (I)
TimeLine	delta time
TimeScale	Scaling factor of ComputerScope data
Len	Number of Data Points Per Channel

Integers:

CurStep, ii	Counter of loop position
i	general purpose indexer
j	record position counter
k	frequency of output counter
off	offset enable
count	active channel counter
num_ch1	number of active channels
num_ch2	number of stored channels
num_ch_e	number of enabled channels
freq	frequency of output points
start, stop	file position of cursors
comp	compression factor exponent for shifting

Boolean:

ASCII	flag for ACSII output request (H)
Flag, Halt, Cease	general loop controllers (I)
Omit	flag for omission of excess points (I)

HICCNVRT.PAS

```
Program Hiccnvrt;       (* by Barry Fishburne *)

                (* This program reads in a marked ComputerScope file and
                   averages the precursor interval as the offset and
                   outputs an ASCII file for the HIC calculation programs
                   that is .RES it can also produce time,value ASCII files
                   for the active channels and the resultant *)
const
   Max_M = 8192;

type
   H_array = array[0..Max_M] of real;

var
   DtaSource                    : File;
   Source, Dest                 : text;
   DestAsc                      : array[1..9] of text;
   Buffer                       : array[1..128] of byte;

   H                            : H_array;

   off, i, ii, j, k, m, typ, comp,
   Count, num_ch1, num_ch2,
   freq, start, stop,num_ch_e   : integer;

   ascii                        : boolean;

   TimeLine, TimeScale,
   CurStep,len ,dt              : Real;

   chan,offset                  : string[8];
   filnam                       : string[30];
   num                          : array[1..4] of integer;
   ave,cal,volts                : array[1..8] of real;
   Title                        : string[80];
   Channel_Status               : array[1..16] of string[1];
   Channel_Name                 : array[1..16] of string[15];

   Time_Mode,
   Current_Block,
   Current_Block_MSB,
   hold                         : string[1];

   Time,
   Current_Point,
   Buffer_Size,
```

```
  disp_s, vc_lf, vc_rt, h_scl, ch_num    : string[2];

  Size              : string[4];
  Lenth             : string[4];

  Threshold,
  Trigger_Thrshld,
  Voltage_Offset,
  Voltage_Scale,
  Channel_Enable,
  Display_Enable     : array[1..16] of string[2];

Begin

        (* input information about run from user *)

   clrscr;

   write('Input file name without extension: ');
     readln(filnam);

   offset := 'YES';
   write('Subtract pre-start average as offset? <YES> ');
     readln(offset);

   off := 1;
   if (UpCase(offset[1]) = 'N') then
     off := 0;

   offset := 'NO';
   write('Include output in ASCII format for channels and resultant? <NO> ');
     readln(offset);
   writeln;

   ascii := false;
   if (UpCase(offset[1]) = 'Y') then
     ascii := true;

      (* set up parameter file as input *)

   Assign(Source, filnam + '.prm');
     Reset(Source);

      (* input and display file information *)

   read(Source,Title);
     write(Title);
     writeln;
   num_ch1 := 0;
   chan := '        ';

   For I := 1 to 16 do
     begin
       read(Source,Channel_Status[I]);
       read(Source,Channel_Name[I]);
       read(Source,Threshold[I]);
       if ord(channel_status[i]) = 3 then
```

```
      begin
        num_ch1:= num_ch1+1;
        str(i:1,hold);
        chan[num_ch1]:= copy(hold,1,1);
        writeln('Channel_Name ',I,': ',Channel_Name[I],'     ');
      end;
  end;

      (* continue reading and displaying file information *)

read(Source, Time);
writeln;
read(Source, Time_Mode);

      (* set multiplier *)

  case Ord(Time_Mode) of
    1 : TimeScale := 1e-6;
    2 : TimeScale := 1e-3;
    3 : TimeScale := 1;
  end;

      (* calculate dt *)

TimeLine := (Ord(Time[2])*256+Ord(Time[1]))*TimeScale;

read(Source, Size);

      (* calculate length of file (i.e. number of points) *)

read(Source, Lenth);
for i := 1 to 4 do
 num[i] := ord(lenth[i]);
len := (num[1] * 4096) + (num[2] * 256) + (num[3] * 16) + num[4];

      (* skip over unnecessary data *)

For i := 380 to 547 do
  read(Source, hold);

read(Source, disp_s);

      (* read number of active channels *)

read(source,ch_num);
  num_ch_e := ord(ch_num[2])*256+ord(ch_num[1]);

    (* set value for real number of channels and repeated channels
          (must be a power of 2)     *)

num_ch2 := num_ch_e;
if (num_ch_e mod 4 <> 0) and (num_ch_e > 2) then
  num_ch2 := 4 * ((num_ch_e div 4) + 1);

    (* read horizontal screen location *)

read(source, h_scl);
```

```
    (* skip over unnecessary data *)

for i:= 554 to 566 do
  read(source,hold);

    (* read vertical cursor left *)

read(source,vc_lf);
start := ord(vc_lf[2])*256+ord(vc_lf[1]);

    (* read vertical cursor right *)

read(source,vc_rt);
stop := ord(vc_rt[2])*256+ord(vc_rt[1]);

    (* find cursor posistions in files *)

comp := ord(h_scl[2])*256+ord(h_scl[1]);

case comp of
   -10..-1 : begin
                start := start*(1 shl abs(comp));
                stop := stop*(1 shl abs(comp));
             end;
     1..10 : begin
                start := start div (1 shl comp);
                stop := stop div (1 shl comp);
             end;
  end;

start := start + ord(disp_s[2])*256+ord(disp_s[1]);
writeln('Time to start output: ',start*timeline:10:6, ' sec.');

stop := stop + ord(disp_s[2])*256+ord(disp_s[1]);
writeln('Time to  stop output: ',stop*timeline:10:6, ' sec.');

writeln;
write('Total Points per Channel: ',len:6:0);
writeln('  --> Points between cursors: ', stop-start);

freq := 1;
write('Fraction denominator of output points: <',freq,'> ');
readln(freq);

     (* get calibration factors *)

for i := 1 to num_ch2 do
  cal[i] := 1;
for i := 1 to num_ch1 do
  begin
    write('Calibration Factor for Channel ', chan[i],
          ' (user units/volt)? ');
    readln(cal[i]);
  end;

      (* end of parameter file read section *)
```

```
Close(Source);

      (* set up data input file *)

Assign(DtaSource,filnam + '.dat');
  Reset(DtaSource);

      (* set up ascii output files *)

if ascii then
  begin
    for count := 1 to num_ch1 do
      begin
        assign(DestAsc[count],filnam + '.ch' + chan[count]);
          rewrite(DestAsc[count]);
      end;
    assign(DestAsc[num_ch1+1],filnam + '.plt');
      rewrite(DestAsc[num_ch1+1]);
  end;

CurStep := 0;
ii := 0;

for i := 1 to 8 do
  ave[i] := 0;

    (* loop through input and calculate resulatant.  Store only data
          between cursors *)

while curstep <= stop do
 begin
   BlockRead(DtaSource, Buffer, 1);
   j := 2;
   while j <= 128 do
      begin
        for count:= 1 to num_ch2 do
          begin
            Buffer[j] := buffer[j] and 15;          (* 00001111 Binary *)
            Volts[count] := 256*(Buffer[j])+Buffer[j-1];
            Volts[count] := Volts[count] / 204.8;
            Volts[count] := Volts[count] + (-10);
            j := j + 2;
          end;

      (* calculate average *)

        for count := 1 to num_ch1 do
          begin
            if curstep = start then
              begin
                writeln;
                ave[count]:= ave[count]/curstep;
                write('Channel ', chan[count], ' offset: ',
                                        ave[count]:10:6);
              end;
            if curstep < start then
```

```
            ave[count] := ave[count] + volts[count];
          end;

        (* calculate resultant and store in array *)

         if (curstep >= start) and (curstep <= stop) then
          begin
            if curstep = start then k := freq;
            if (k mod freq) = 0 then
              begin
                ii := ii + 1;
                H[ii] := 0;
                for count := 1 to num_ch1 do
                  begin
                    H[ii] := H[ii] +
                        sqr((volts[count]-ave[count] * off) * cal[count]);
                    if ascii then
                      writeln(DestAsc[count],(curstep-start)*timeline:15:6,
                        (volts[count]-ave[count] * off) * cal[count]:15:6);
                  end;
                H[ii] := sqrt(H[ii]);
                if ascii then
                  writeln(DestAsc[num_ch1+1],(curstep-start)*timeline:15:6,
                      H[ii]:15:6);
              end;
          end;
         k := k+1;
         CurStep := CurStep + 1;
       end;
end;

        (* close data input file *)

Close(DtaSource);

        (* close ASCII output files *)

if ascii then
  for count := 1 to Num_Ch1+1 do
    close(DestAsc[count]);

        (* open output file *)

assign(dest,filnam + '.res');
  rewrite(dest);

        (* output file description for future hic usage *)

m := ii - 1;                          (* m = number of subintervals *)
writeln(dest, m);

dt := TimeLine * freq;
writeln(dest, dt);

typ := 0;                             (* waveform type = nonstandard *)
writeln(dest, typ);
```

```
    for i := 1 to ii do
      begin
            (* output data points of time and user units *)
        writeln(dest, H[i]:15:6);
      end;
            (* close output file *)
    close(dest);
end.
```

HIC_BAS.BAS

```
10 REM *** This program calculates the Head Injury Criterion
20 REM *** (HIC) based on a discrectized acceleration-time
30 REM *** curve which must reside in the input file.
40 REM ***
50 REM *** define and initialize variables
60 REM ***
70 MAXM = 8192: NONSTANDARD = 0 : HALFSINE = 1 : TRIANGLE = 2 : SQUARE = 3
80 DIM H(MAXM)
90 AP = 0 : HMAX = 0 : H(0) = 0
100 REM ***
110 REM *** GET DATA
120 REM ***
130 REM *** read in data points
140 REM ***
150 CLS : INPUT "What is the input file name";FILE$
160 PRINT : INPUT"Use which method (1-3)";METHOD
170 CLS : PRINT "Reading Input Data - Please wait." : PRINT
180 OPEN FILE$ FOR INPUT AS 1
190 INPUT #1, M, DT, TYP
200 T = DT * M
210 FOR I = 0 TO M
220   INPUT #1, PT
230   IF PT > AP THEN AP = PT
240   IF I = 0 THEN H(I) = PT ELSE H(I) = H(I-1) + ((PREVPT + PT) / 2)
250   SI = SI + PT ^ 2.5
260   PREVPT = PT
270 NEXT
280 CLOSE #1
290 SI = SI * DT
300 PRINT "SI = ";SI : PRINT
310 REM *** CALCULATE H_MAX
320 ON METHOD GOSUB 510,680,810
330 GOSUB 350
340 END
350 REM ***
360 REM *** PROCEDURE PRINT_SOLUTION
370 REM ***
380 IF TYP=NONSTANDARD THEN S$="non-standard" ELSE IF TYP=HALFSINE THEN S$="half
-sine" : C=.4146 ELSE IF TYP=TRIANGLE THEN S$="triangle" : C=.2464 ELSE S$="squa
re" : C=1
390 IF TYP<>0 THEN EXACTHIC = C * AP^2.5 * T
400 PRINT" Waveform type = ";S$
410 PRINT"              m =     ";M
420 PRINT USING "             dt =        #.######";DT; : PRINT " sec."
430 PRINT USING "     peak acc. =     ####.######";AP; : PRINT " Gs" : PRINT
440 PRINT"             T1 = ";: PRINT USING "#######.#######";T1*DT;: PRINT" msec
."
450 PRINT"             T1 = ";: PRINT USING "#######.#######";T2*DT;: PRINT" msec
."
```

```
460 PRINT : PRINT"EXACT SOLUTION:  ";: IF TYP=0 THEN PRINT "cannot be found." EL
SE PRINT USING "#######.####";EXACTHIC
470 PRINT"CALCULATED HIC:  ";: PRINT USING "#######.####";HIC
480 IF TYP <> 0 THEN PRINT" PERCENT ERROR:  ";: PRINT USING "#######.####";(100*
(EXACTHIC-HIC)/EXACTHIC)
490 PRINT
500 RETURN
510 REM ***
520 REM *** METHOD I - BRUTE FORCE
530 REM ***
540 PRINT : PRINT "Calculating HIC using BRUTE FORCE METHOD..." : PRINT
550 FOR I = 0 TO M-1
560   FOR J = I+1 TO M
570     HC = 0
580     FOR K = I TO J-1
590       IF K = 0 THEN HC = H(0) ELSE HC = HC + H(K) - H(K-1)
600     NEXT K
610     ASTART = HC / (J-I)
620     HC = (J-I) * ASTART^2.5
630     IF HC > HMAX THEN HMAX = HC : T2 = J-1 : T1 = I-1
640   NEXT J
650 NEXT I
660 HIC = DT * HMAX
670 RETURN
680 REM ***
690 REM *** METHOD II - PARTIAL SUMS
700 REM ***
710 PRINT : PRINT "Calculating HIC using PARTIAL SUMS METHOD..." : PRINT
720 FOR I = 0 TO M-1
730   FOR J = I+1 TO M
740     ASTART = (H(J) - H(I)) / (J-I)
750     HC = (J-I) * ASTART^2.5
760     IF HC > HMAX THEN HMAX = HC : T2 = J : T1 = I
770   NEXT J
780 NEXT I
790 HIC = DT * HMAX
800 RETURN
810 REM ***
820 REM *** METHOD III - PARTIAL SUMS & SLIDING ENDPOINTS
830 REM ***
840 REM *** Find right endpoint
850 REM ***
860 PRINT : PRINT "Calculating HIC using PARTIAL SUMS/SLIDING ENDPOINTS METHOD..
." : PRINT
870 LEFT = 0 : RIGHT = -1 : FALSE = 0 : TRUE = -1
880 RDONE = FALSE : LDONE = FALSE : SLIDE = RIGHT
890 I = 0 : J = M
900 ASTART = (H(J) - H(I)) / (J-I)
910 HC = (J-I) * ASTART^2.5
920 IF HC >  HMAX THEN HMAX = HC ELSE 950
930 IF SLIDE = RIGHT THEN LDONE = FALSE : J=J-1 ELSE RDONE = FALSE : I=I+1
940 GOTO 980
```

```
950 REM *** HC <= HMAX
960 IF SLIDE = RIGHT THEN RDONE = TRUE : SLIDE = LEFT : J=J+1 : IF NOT(LDONE) TH
EN I=I+1 : GOTO 980 ELSE GOTO 980
970 LDONE = TRUE : SLIDE = RIGHT : I=I-1 : IF NOT(RDONE) THEN J=J-1
980 IF NOT(RDONE) OR NOT(LDONE) THEN 900
990 T1 = I : T2 = J
1000 HIC = DT * HMAX
1010 RETURN
```

HIC_C.C

```
/*************************************************************************
 *
 *      This program calculates the Head Injury Criterion
 *      (HIC) based on a discretized acceleration-time
 *      curve which must reside in the input file.
 *
 */

#include <stdio.h>
#include <math.h>

/* define and initialize variables                                      */

#define TWO_POINT_FIVE  2.5          /* the exponent in the integral */
#define          MAX_M 8192          /* maximum size of the problem  */

#define    NONSTANDARD    0          /* waveform shape:non-standard  */
#define       HALFSINE    1          /* waveform shape: half-sine    */
#define       TRIANGLE    2          /* waveform shape: triangle     */
#define         SQUARE    3          /* waveform shape: square       */

#define           LEFT    0
#define          RIGHT    1

#define          FALSE    0
#define           TRUE    1

          int m,                     /* number of subintervals       */
              typ,                   /* waveform shape, either
                                        HALFSINE, TRIANGLE or SQUARE */
              t1, t2;                /* two points in time which
                                        maximize the HIC             */


        float ap = 0.0,              /* peak acceleration            */
              dt,                    /* delta t                      */
              T,                     /* pulse width (time)           */
              HIC,                   /* final HIC number
                                        (i.e. h_max * dt)            */
              SI = 0.0,              /* Severity Index               */
              H[MAX_M];              /* data point array             */

         char *method,
              *filnam[30];

SetParameters(argc, argv)
int   argc;
char *argv[];
```

```
{
  char *s;

  system("cls");

  while (--argc > 0 && (*++argv)[0] == '/') {
    s = argv[0] + 1;
    switch(*s) {
      case 'm':
        method = argv[0] + 2;
        break;
      case 'f':
        strcpy(filnam, argv[0] + 2);
        break;
    }
  }

  printf("Parameters:\n");
  printf("\t\t    Method: %c\n", *method);
  printf("\t\tInput File: %s\n", filnam);

}

GetData()

{
        float   prevpt, pt, integ;
          int   i;
         FILE   *filvar;

        /* read in no. of subintervals, delta time, and waveform type   */

        printf("\nReading Input Data - Please Wait...");
        filvar = fopen(filnam, "r");

        fscanf(filvar, "%d %f %d", &m, &dt, &typ);

        T = dt * m;                     /* pulse duration               */

/* read in data points, find peak accelerations and integrate          */

        for (i=0; i<=m; i++) {
          fscanf(filvar, "%f", &pt);          /* read in data point            */

          if (pt > ap)              /*  find peak accelerations       */
            ap = pt;

        /*  Integrate piecewise, using trapezoidal integration, and
            storing the results in H[].                                */

           if (i > 0) {
             integ = (prevpt + pt) / 2;
             H[i] = H[i-1] + integ;
             SI += pow(integ, TWO_POINT_FIVE);
           }
           else
```

```
            H[i] = pt;

            prevpt = pt;

        }

        fclose(filvar);

}

method1()

{

          int i, j, k;

        float a_star_t,                 /* discretized representation
                                           of analog acc. pulse        */
              h_max = 0.0,              /* maximum HIC value           */
              HC = 0.0;                 /* temp HIC                    */

/* Calculate h_max                                                     */

        printf("\n\nCalculating HIC using BRUTE FORCE METHOD...");

        for (i=0; i<m; i++) {
           for (j=i+1; j<=m; j++) {

              HC = 0.0;

              for (k=i; k<j; k++)
                if (k == 0)
                  HC = H[0];
                else
                  HC += (H[k] - H[k-1]);

              a_star_t = HC / (j-i);
              HC = (j-i) * pow(a_star_t, TWO_POINT_FIVE );

              if (HC > h_max) {                      /*  Maximize H[i][j]    */
                 h_max = HC;
                 t1 = i-1;
                 t2 = j-1;
              } /* if H */
           } /* for j */
        } /* for i */

        /* Calculate the HIC, based on h_max, and print it out.        */

        HIC = dt * h_max;

}

method2()

{
```

```
          int i, j, k;

        float a_star_t,                 /* discretized representation
                                           of analog acc. pulse        */
              h_max = 0.0,              /* maximum HIC value           */
              HC = 0.0;                 /* temp HIC                    */

/* Calculate h_max                                                     */

        printf("\n\nCalculating HIC using PARTIAL SUMS METHOD...");

        for (i=0; i<m; i++) {
           for (j=i+1; j<=m; j++) {

              a_star_t = (H[j] - H[i]) / (j-i);
              HC = (j-i) * pow(a_star_t, TWO_POINT_FIVE );

              if (HC > h_max) {             /*  Maximize H[i][j]    */
                 h_max = HC;
                 t1 = i;
                 t2 = j;
              }
           }
        }

        /* Calculate the HIC, based on h_max, and print it out.        */

        HIC = dt * h_max;

}

method3()

{

          int L_DONE = FALSE,
              R_DONE = FALSE,
              SLIDE = RIGHT,
              i = 0,
              j = m,
              k;

        float a_star_t,                 /* discretized representation
                                           of analog acc. pulse        */
              h_max = 0.0,              /* maximum HIC value           */
              HC = 0.0;                 /* temp HIC                    */

/* Calculate h_max                                                     */

        printf("\n\nCalculating HIC using PARTIAL SUMS/SLIDING ENDPOINTS METHOD...")

        while (!R_DONE || !L_DONE) {

           a_star_t = (H[j] - H[i]) / (j-i);
           HC = (j-i) * pow(a_star_t, TWO_POINT_FIVE );

           if (HC > h_max) {                /*  Maximize H[0][j]    */
```

```
                h_max = HC;


                if (SLIDE == RIGHT) {
                  L_DONE = FALSE;
                  --j;
                }

                else {
                  R_DONE = FALSE;
                  ++i;
                }
              }
              else {
                if (SLIDE == RIGHT) {
                  R_DONE = TRUE;
                  SLIDE = LEFT;
                  ++j;
                  if (!L_DONE)
                    ++i;
                }
                else {
                  L_DONE = TRUE;
                  SLIDE = RIGHT;
                  --i;
                  if (!R_DONE)
                    --j;
                }
              }
            }

        t1 = i;
        t2 = j;

        HIC = dt * h_max;

}

/*****************************************************************************
 *
 *      PROCEDURE: PrintSolution()
 *
 *      This function prints the "exact solution" for a few
 *      particular waveform shapes (half-sine, triangle, and
 *      square waves.)  The percent error is calculated also.
 *
 *      Exact and calculated solutions and error are printed.
 *
 */

PrintSolution()

{
        float coeff, exact_hic;
        char *s;

        printf ("\n");
```

```
        switch (typ) {

           case NONSTANDARD:
              s = "non-standard";
              break;
           case HALFSINE:
              s = "half-sine";
              coeff = 0.4146;
              break;
           case TRIANGLE:
              s = "triangle";
              coeff = 0.2464;
              break;
           case SQUARE:
              s = "square";
              coeff = 1.;
              break;
           default:
              printf("\n %30s \n", "WARNING : Bad Data!");
        }

        if (typ != 0)
          exact_hic = (coeff * pow(ap, TWO_POINT_FIVE) * T );

        printf("\n%30s %-10s", "waveform = ", s);
        printf("\n\n%30s %-5d\n",   "no. of subintervals = ", m);
        printf("%30s %-5g\n", "delta time (msec) = ", dt);
        printf("%30s %-5g\n", "peak acceleration = ", ap);
        printf("\n%30s %-5g\n", "T1 = ", t1*dt);
        printf("%30s %-5g\n", "T2 = ", t2*dt);

        printf("\n%27s", "EXACT SOLUTION");
        if (typ != 0)
          printf(" = %-5g\n", exact_hic );
        else
          printf("%s\n", ": cannot be found.");

        printf("%30s %-5g\n", "CALCULATED HIC = ", HIC);

        if (typ != 0)
          printf("\n%30s %-5g\n\n", "percent error = ",
                (100*(exact_hic-HIC)/exact_hic) );

}

main (argc, argv)
int   argc;
char *argv[];

{

        SetParameters(argc, argv);

        GetData();

        SI *= dt;
        printf("\n\nSeverity Index = %-5g", SI);
```

```
        switch (*method) {
          case '1':
            method1();
            break;
          case '2':
            method2();
            break;
          case '3':
            method3();
            break;
        }

        PrintSolution();

}
```

HIC_PAS.PAS

```
(*************************************************************************
 *
 *      This program calculates the Head Injury Criterion
 *      (HIC) based on a descretized acceleration-time
 *      curve which must reside in the input file.
 *
 *)

program hic;

(* define and initialize variables                                      *)

const

           MAX_M = 8192;              (* maximum number of subintervals *)

      NONSTANDARD = 0;               (* typ=0 indicates nonstandard wave *)
         HALFSINE = 1;               (*     1 indicates halfsine wave    *)
         TRIANGLE = 2;               (*     2 indicates triangle wave    *)
           SQUARE = 3;               (*     3 indicates square wave      *)

type

   H_array = array[0..MAX_M] of real;   (* Array to hold "integrations" *)

var

  SI, ap, T, a_star_t,
  h_max, HIC, HC, dt      : real;
  H                       : H_array;
  m, typ, t1, t2, i, j, k : integer;
  filvar                  : TEXT;
  filnam                  : string[30]; (* Filename of input file        *)
  method                  : char;       (* method 1 = BRUTE FORCE
                                                   2 = PARTIAL SUMS
                                                   3 = PARTIAL SUMS
                                                       & SLIDING ENDPOINTS *)
  ComputerScope           : boolean;    (* Are we using a Computerscope
                                           input file? (True or False)    *)

(*************************************************************************
 *
 * PROCEDURE: GetCScopeData
 *
 * This procedure reads a marked computerscope output file, calculates
 * the resultant of the three active channels, and outputs the results
```

```
 * to <filnam>.res in the proper file format.  In addition, since the
 * data has already been read and the resultants calculated there is no
 * need to reread the data points so the resultants are "integrated"
 * and stored in the H array.
 *
 *)

procedure GetCScopeData;

var
   DtaSource                    : File;
   Source, Dest                 : text;
   DestAsc                      : array[1..9] of text;
   Buffer                       : array[1..128] of byte;

   off, i, ii, j, k, comp,
   Count, num_ch1, num_ch2,
   freq, start, stop, num_ch_e : integer;

   ascii                        : boolean;

   TimeLine, TimeScale,
   CurStep,len                  : Real;

   chan,offset                  : string[8];
   num                          : array[1..4] of integer;
   ave,cal,volts                : array[1..8] of real;
   Title                        : string[80];
   Channel_Status               : array[1..16] of string[1];
   Channel_Name                 : array[1..16] of string[15];

   Time_Mode,
   Current_Block,
   Current_Block_MSB,
   hold                         : string[1];

   Time,
   Current_Point,
   Buffer_Size,
   disp_s, vc_lf, vc_rt, h_scl,ch_num    : string[2];

   Size              : string[4];
   Lenth             : string[4];

   Threshold,
   Trigger_Thrshld,
   Voltage_Offset,
   Voltage_Scale,
   Channel_Enable,
   Display_Enable     : array[1..16] of string[2];

 Begin

       (* input information about run from user *)

    offset := 'YES';
    write('Subtract pre-start average as offset? <YES> ');
```

```
  readln(offset);

off := 1;
if (UpCase(offset[1]) = 'N') then
  off := 0;

offset := 'NO';
write('Include output in ASCII format for channels and resultant? <NO> ');
  readln(offset);
ascii := false;
if (UpCase(offset[1]) = 'Y') then
  ascii := true;

    (* set up parameter file as input *)

Assign(Source, filnam + '.prm');
  Reset(Source);

    (* input and display file information *)

read(Source,Title);
  GotoXY(1, 1);
  clreol;
  write(Title);
  clreol;
  writeln;
num_ch1 := 0;
chan := '         ';

For I := 1 to 16 do
  begin
    read(Source,Channel_Status[I]);
    read(Source,Channel_Name[I]);
    read(Source,Threshold[I]);
    if ord(channel_status[i]) = 3 then
      begin
        num_ch1:= num_ch1+1;
        str(i:1,hold);
        chan[num_ch1]:= copy(hold,1,1);
        writeln('Channel_Name ',I,': ',Channel_Name[I],'      ');
      end;
  end;

    (* continue reading and displaying file information *)

read(Source, Time);
writeln;
read(Source, Time_Mode);

    (* set multiplier *)

  case Ord(Time_Mode) of
    1 : TimeScale := 1e-6;
    2 : TimeScale := 1e-3;
    3 : TimeScale := 1;
  end;
```

```
    (* calculate dt *)

TimeLine := (Ord(Time[2])*256+Ord(Time[1]))*TimeScale;

read(Source, Size);

    (* calculate length of file (i.e. number of points) *)

read(Source, Lenth);
for i := 1 to 4 do
 num[i] := ord(lenth[i]);
len := (num[1] * 4096) + (num[2] * 256) + (num[3] * 16) + num[4];

    (* skip over unnecessary data *)

For i := 380 to 547 do
  read(Source, hold);

read(Source, disp_s);

    (* read number of enabled channels *)

read(source,ch_num);
num_ch_e := ord(ch_num[2])*256+ord(ch_num[1]);

    (* set value for real number of channels and repeated channels
       (must be a power of 2)                                      *)

num_ch2 := num_ch_e;
if (num_ch_e mod 4 <> 0) and (num_ch_e > 2) then
  num_ch2 := 4 * ((num_ch_e div 4) + 1);

    (* read horizontal screen location *)

read(source, h_scl);

    (* skip over unnecessary data *)

for i:= 554 to 566 do
  read(source,hold);

    (* read vertical cursor left *)

read(source,vc_lf);
start := ord(vc_lf[2])*256+ord(vc_lf[1]);

    (* read vertical cursor right *)

read(source,vc_rt);
stop := ord(vc_rt[2])*256+ord(vc_rt[1]);

    (* find cursor posistions in files *)

comp := ord(h_scl[2])*256+ord(h_scl[1]);

case comp of
   -10..-1 : begin
```

```
                    start := start*(1 shl abs(comp));
                    stop := stop*(1 shl abs(comp));
                end;
        1..10 : begin
                    start := start div (1 shl comp);
                    stop := stop div (1 shl comp);
                end;
    end;

  start := start + ord(disp_s[2])*256+ord(disp_s[1]);
  writeln('Time to start output: ',start*timeline:10:6, ' sec.');

  stop := stop + ord(disp_s[2])*256+ord(disp_s[1]);
  writeln('Time to  stop output: ',stop*timeline:10:6, ' sec.');

  writeln;
  write('Total Points per Channel: ',len:6:0);
  writeln('  -->  Points between cursors: ', stop-start);

  freq := 1;
  write('Fraction denominator of output points: <',freq,'> ');
  readln(freq);

      (* get calibration factors *)

  for i := 1 to num_ch2 do
    cal[i] := 1;
  for i := 1 to num_ch1 do
    begin
      write('Calibration Factor for Channel ', chan[i],
            ' (user units/volt)? ');
      readln(cal[i]);
    end;

      (* end of parameter file read section *)

  Close(Source);

      (* set up data input file *)

  Assign(DtaSource,filnam + '.dat');
    Reset(DtaSource);

(* set up ascii output files *)

  if ascii then
    begin
      for count := 1 to num_ch1 do
        begin
          assign(DestAsc[count],filnam + '.ch' + chan[count]);
            rewrite(DestAsc[count]);
        end;
      assign(DestAsc[num_ch1+1],filnam + '.plt');
        rewrite(DestAsc[num_ch1+1]);
    end;

  CurStep := 0;
```

```
ii := 0;
ap := 0;

for i := 1 to 8 do
  ave[i] := 0;

    (* loop through input and calculate resulatant.  Store only data
       between cursors *)

while curstep <= stop do
 begin
   BlockRead(DtaSource, Buffer, 1);
   j := 2;
   while j <= 128 do
      begin
        for count:= 1 to num_ch2 do
          begin
            Buffer[j] := buffer[j] and 15;          (* 00001111 Binary *)
            Volts[count] := 256*(Buffer[j])+Buffer[j-1];
            Volts[count] := Volts[count] / 204.8;
            Volts[count] := Volts[count] + (-10);
            j := j + 2;
          end;

    (* calculate average *)

        for count := 1 to num_ch1 do
          begin
            if curstep = start then
              begin
                writeln;
                ave[count]:= ave[count]/curstep;
                write('Channel ', chan[count], ' offset: ',
                      ave[count]:10:6);
              end;
            if curstep < start then
              ave[count] := ave[count] + volts[count];
          end;

    (* calculate resultant and store in array *)

        if (curstep >= start) and (curstep <= stop) then
          begin
            if curstep = start then k := freq;
            if (k mod freq) = 0 then
              begin
                ii := ii + 1;
                H[ii] := 0;
                for count := 1 to num_ch1 do
                  begin
                    H[ii] := H[ii] +
                        sqr((volts[count]-ave[count] * off) * cal[count]);
                    if ascii then
                      writeln(DestAsc[count],(curstep-start)*timeline:15:6,
                        (volts[count]-ave[count] * off) * cal[count]:15:6);
                  end;
                H[ii] := sqrt(H[ii]);
```

```
                    if H[ii] > ap then
                      ap := H[ii];
                    if ascii then
                      writeln(DestAsc[num_ch1+1],(curstep-start)*timeline:15:6,
                           H[ii]:15:6);
                  end;
              end;
            k := k+1;
            CurStep := CurStep + 1;
          end;
      end;

        (* close data input file *)

      Close(DtaSource);

  (* close ASCII output files *)

      if ascii then
        for count := 1 to Num_Ch1+1 do
          close(DestAsc[count]);

        (* open output file *)

     assign(dest,filnam + '.res');
       rewrite(dest);

        (* output file description for future hic usage *)

     m := ii - 1;                              (* m = number of subintervals *)
     writeln(dest, m);

     dt := TimeLine * freq;
     writeln(dest, dt);

     typ := 0;                                 (* waveform type = nonstandard *)
     writeln(dest, typ);

        (* initialize Severity Index *)

     SI := 0;

     for i := 1 to ii do
       begin

        (* output data points of time and user units *)

        writeln(dest, H[i]:15:6);

        (* calculate Severity Index *)

        SI := SI + (H[i] * H[i] * sqrt(H[i]));

        if i = 1 then
          H[0] := H[1]
        else
          begin
```

```
            if method = '1' then        (* store "heights" *)
              H[i-1] := H[i]
            else                        (* store summed "heights" *)
              H[i-1] := H[i] + H[i-2];
          end;
      end;

        (* close output file *)

    close(dest);

end;


(*************************************************************************
 *
 *      PROCEDURE: PrintSolution
 *
 *      This function prints the calculated solution and if known,
 *      the exact solution.  The percent error is also calculated.
 *
 *)

Procedure PrintSolution(ap, T, hic, dt:real; typ, t1, t2 : integer);

var
   c, exact_hic : real;
   s : string[12];

begin

   case typ of
     NONSTANDARD: s := 'non-standard';
       HALFSINE : begin
                    s := 'half-sine';
                    c := 0.4146;
                  end;
       TRIANGLE : begin
                    s := 'triangle';
                    c := 0.2464;
                  end;
         SQUARE : begin
                    s := 'square';
                    c := 1.0;
                  end;
   end; { case }

   if typ <> 0 then
     exact_hic := c * (ap * ap * sqrt(ap) * T);

   writeln('       Waveform type = ',s);
   writeln;
   writeln(' no. of subintervals = ', m:8);
   writeln('     delta time (dt) = ', dt:15:6, ' sec.');
   writeln('   peak acceleration = ',ap:15:6, ' Gs');
   writeln;
   writeln('                  T1 = ', t1*dt:15:6, ' sec.');
```

```
   writeln('                    T2 = ', t2*dt:15:6, ' sec.');
   writeln;

   write  ('      EXACT SOLUTION');
   if typ <> 0 then
     writeln(' = ', exact_hic:14:5)
   else
     writeln(': cannot be found.');

   writeln('      CALCULATED HIC = ', hic:14:5);
   writeln;

   if typ <> 0 then
      writeln('    Percent error = ',(100*(exact_hic-hic)/exact_hic):15:6);

end;

procedure GetData;

var
   prevpt, pt : real;
            i : integer;

begin

   ap := 0.0;
   h[0] := 0.0;
   SI := 0;

        (* read in data, find peak acceleration and integrate *)

   writeln('Reading Input Data - Please wait.');
   writeln;

   assign(filvar, filnam);
   reset(filvar);

   readln(filvar, m);
   readln(filvar, dt);
   readln(filvar, typ);
   prevpt := 0;
   for i := 0 to m do
     begin

        (* read data point *)

       read(filvar, pt);

        (* calculate Severity Index *)

       SI := SI + (pt * pt * sqrt(pt));

       if (pt > ap) then
         ap := pt;

       if (i > 0) then
         if (method = '1') then                (* store "height" *)
```

```
            h[i] := (prevpt + pt) / 2
          else                                    (* store summed "height" *)
            h[i] := h[i-1] + ((prevpt + pt) / 2)
        else
          h[i] := pt;

        prevpt := pt;
      end;

      close(filvar);

end;

procedure method1;

begin

   T := dt * m;
   h_max := 0.0;

   write('Calculating HIC using BRUTE FORCE METHOD...');

        (* loop through every possible combination of t1 and t2 *)

   for i := 0 to m - 1 do
     for j := i + 1 to m do
       begin
         HC := 0.0;

        (* sum the "heights" (i.e. integrate) *)

         for k := i to j-1 do
           HC := HC + H[k];

        (*    *              2.5
             a (t) = integral          *)

         a_star_t := HC / (j-i);
         HC := (j-i) * (a_star_t * a_star_t * sqrt(a_star_t));

         if (HC > h_max) then
            begin
               h_max := HC;
               t1 := i-1;
               t2 := j-1;
            end;
       end;

   hic := dt * h_max;

   writeln;
   writeln;

end;

procedure method2;
```

```
begin

   T := dt * m;
   h_max := 0.0;

   write('Calculating HIC using PARTIAL SUMS METHOD...');

   for i := 0 to m-1 do
     for j := i + 1 to m do
       begin
         a_star_t := (h[j] - h[i]) / (j-i);
         HC := (j-i) * (a_star_t * a_star_t * sqrt(a_star_t));

         if (HC > h_max) then
            begin
               h_max := HC;
               t1 := i;
               t2 := j;
            end
       end;

   hic := dt * h_max;

   writeln;
   writeln;

end;

procedure method3;

const
   LEFT  = 0;
   RIGHT = 1;

var
   L_DONE, R_DONE : boolean;
   SLIDE          : byte;

begin

   T := dt * m;
   h_max := 0.0;

   i := 0;
   j := m;

        (* R_DONE: indicates done sliding the right endpoint (T or F) *)

   R_DONE := FALSE;

        (* L_DONE: indicates done sliding the left endpoint (T or F) *)

   L_DONE := FALSE;

        (* slide: right indicates sliding the right endpoint *)

   SLIDE := RIGHT;
```

```
    write('Calculating HIC using PARTIAL SUMS/SLIDING ENDPOINTS METHOD...');

    while (not(R_DONE)) or (not(L_DONE)) do
       begin
          a_star_t := (h[j] - h[i]) / (j-i);
          HC := (j-i) * (a_star_t * a_star_t * sqrt(a_star_t));

          if (HC > h_max) then
            begin
              h_max := HC;

              if SLIDE = RIGHT then
                begin
                  L_DONE := FALSE; (* will want to check left endpoint again *)
                  j := j - 1;      (* continue to move j *)
                end
              else
                begin
                  R_DONE := FALSE; (* will want to check right endpoint again *)
                  i := i + 1;      (* continue to move i *)
                end;
            end
          else
            begin
              if SLIDE = RIGHT then
                begin
                  R_DONE := TRUE;  (* done sliding right endpoint (temporarily?) *)
                  SLIDE := LEFT;   (* start sliding left endpoint *)
                  j := j + 1;      (* put j back to last O.K. value *)

                  if not(L_DONE) then (* if going to move left endpoint again *)
                    i := i + 1;       (* move t1 toward center *)
                end
              else
                begin
                  L_DONE := TRUE;  (* done sliding left endpoint (temporarily?) *)
                  SLIDE := RIGHT;  (* start sliding right endpoint *)
                  i := i - 1;      (* put i back to last O.K. value *)
                  if not(R_DONE) then (* if going to move right endpoint again *)
                    j := j - 1;       (*  move t2 toward center *)
                end;
            end;
       end;

    t1 := i;
    t2 := j;

    hic := dt * h_max;

    writeln;
    writeln;

end;

procedure SetParameters;
```

```
var
   key : string[40];

begin

   method := '0';
   filnam := '';
   ComputerScope := FALSE;

   if ParamCount <> 0 then
     begin
       for i := 1 to ParamCount do
         begin
           key := ParamStr(i);
           case UpCase(key[2]) of
             'M': method := copy(ParamStr(i), 3, 1);
             'F': filnam  := copy(ParamStr(i), 3, length(ParamStr(i)) - 2);
             'C': ComputerScope := TRUE;
           end;
         end;
     end;
   if (ord(method) < 49) or (ord(method) > 51) then
     begin
       write('Method not specified.  Enter method (1-3): ');
       readln(method);
     end;
   if filnam = '' then
     begin
       write('No input file name specified.  Enter file name: ');
       readln(filnam);
     end;

   ClrScr;
   writeln('Parameters:');
   writeln('                          Method: ', method);
   writeln('                      Input File: ', filnam);
   write  ('              ComputerScope File: ');
   if ComputerScope then
     writeln('YES')
   else
     writeln('NO');
   writeln;

end;

begin

   ClrScr;
   SetParameters;

   Window(1, 6, 80, 40);

   if ComputerScope then
     GetCScopeData
   else
     GetData;
```

```
(* Calculate h_max                                                    *)

   ClrScr;

   SI := SI * dt;

   writeln('SI = ', SI:0:6);
   writeln;

   case method of
     '1': method1;
     '2': method2;
     '3': method3;
   end;

   PrintSolution(ap, T, hic, dt, typ, t1, t2);

   Window(1, 1, 80, 25);

end.
```

B.2.4.1 Program Execution

(1) Pascal Version: HICCNVRT [/f < *filename* >]

User will be prompted if < *filename* > is not specified.

(2) Basic version: BASICA HIC_BAS

User will be prompted for method and input file name.

(3) C Version: HIC_C [/m #][/f < *filename* >]

User will be prompted if # or < *filename* > is not specified.

is method number; < *filename* > is input file name.

(4) Pascal Version: HIC_PAS [/m #] [/f < *filename* >] [/c]

User will be prompted if # or < *filename* > is not specified.

is method number; < *filename* > is input file name.

/c parameter indicates that conversion of Computerscope data is necessary.

B.2.4.2 COMPUTERSCOPE File Format

The following section describes in detail how data is saved by the Scope Driver software. Included are descriptions of both the standard and compressed data file formats along with specifications for the parameter file. This information will be of use to those persons writing their own analysis software. Note that an understanding of I.B.M. DOS 2.1 is assumed throughout the following discussion.

PARAMETER FILE

Parameter files are identified by the extension 'prm' attached to the file name and consist of a single 1024 byte record. Each parameter file contains all of the scope settings in effect when the corresponding data file was saved. Note that many of the parameters, such as digital thresholds, are included for use by supplemental analysis programs provided by R.C. Electronics and may not be relevant to other applications.

Parameter file definition

1 to 80	TITLE -	Title field consisting of 80 ASCII characters
81 to 98	CH 1 -	CHANNEL DEFINITION consisting of 18 bytes for each channel. The first byte specifies the current status as follows; 0 - off,digital 1 - off,analog 2 - on,digital 3 - on,analog
99 to 116	CH 2	
117 to 134	CH 3	
135 to 152	CH 4	
153 to 170	CH 5	
171 to 188	CH 6	
189 to 206	CH 7	Bytes 2 through 16 are the name field consisting of 15 ASCII characters.
207 to 224	CH 8	
225 to 242	CH A	
243 to 260	CH B	Bytes 17,18 are the digital threshold in 12 bit offset binary notation (ie.. 000h = -10 volt, FFFh = +10 volt). Byte 17 contains the 8 LS bits while byte 18 has the 4 MS bits right justified.
261 to 278	CH C	
279 to 296	CH D	
297 to 314	CH E	
315 to 332	CH F	
333 to 350	CH G	
351 to 368	CH H	
369 to 370	TIME -	Sample period from 1 to 999 (LSB,MSB)
371	T.MODE -	Current time scale; 1 - usec. 2 - msec. 3 - sec
372 to 375	SIZE -	Size of the corresponding compressed archive file buffer in 1024 byte records.
376 to 379	LENGTH -	Total number of contiguous samples in the corresponding data file (divide by the number of enabled channels to obtain the number of samples per channel). This variable is used by the optional extended data acquisition program to specify the number of contiguous data points in the current buffer file.
380	C.BLK -	Current buffer block in use (64K increments).

Parameter file definition (continued)

381 to 382	C.PNT	- Last data point acquired within the current buffer block.
383 to 384	B.SIZE	- Buffer memory size in 64K byte increments.
385	CM.BLK	- Most sifnificant byte value of current block in use (LSB at 380)
386 to 387	EXT	- TRIGGER THRESHOLD LEVEL as a 12 bit (LS byte, MS nibble) offset binary value ranging from -10 volts (000) to +10 volts (FFFh). The current trigger channel selected determines which value will be loaded into the trigger DAC. Note that the DAC output, as measured at the J1 connector, will range from -2.5 volts (000) to +2.5 volts (FFFh).
388 to 389	CH 1	
390 to 391	CH 2	
392 to 393	CH 3	
394 to 395	CH 4	
396 to 397	CH 5	
398 to 399	CH 6	
400 to 401	CH 7	
402 to 403	CH 8	
404 to 405	CH A	
406 to 407	CH B	
408 to 409	CH C	
410 to 411	CH D	
412 to 413	CH E	
414 to 415	CH F	
416 to 417	CH G	
418 to 419	CH H	
420 to 421	CH 1	- DISPLAY VOLTAGE OFFSET as a 12 bit value used to shift the corresponding channel display vertically on the screen. A value of 0 corresponds to the top of the display with positive values shifting the zero voltage down the screen. Negative values (2's compliment) will shift the trace vertically up the display. A value of 95 will center a trace zero voltage in the middle of the display.
422 to 423	CH 2	
424 to 425	CH 3	
426 to 427	CH 4	
428 to 429	CH 5	
430 to 431	CH 6	
432 to 433	CH 7	
434 to 435	CH 8	
436 to 537	CH A	
438 to 439	CH B	
440 to 441	CH C	
442 to 443	CH D	
444 to 445	CH E	
446 to 447	CH F	
448 to 449	CH G	
450 to 451	CH H	

452 to 453	CH 1	- DISPLAY VOLTAGE SCALE controls the amplitude scale
454 to 455	CH 2	of displayed traces by varying the number of converter
456 to 457	CH 3	quantization levels per display pixel according to the
458 to 459	CH 4	following table;
460 to 461	CH 5	
462 to 463	CH 6	0 - 200 mV/div. (9.7 mV/pixel)
464 to 465	CH 7	1 - 500 mV/div. (19.5 mV/pixel)
466 to 467	CH 8	2 - 1 Volt/div. (39 mV/pixel)
468 to 469	CH A	3 - 2 Volt/div. (78 mV/pixel)
470 to 471	CH B	4 - 5 Volt/div. (156 mV/pixel)
472 to 473	CH C	5 - 10 Volt/div. (312 mV/pixel)
474 to 475	CH D	6 - 20 Volt/div. (625 mV/pixel)
476 to 477	CH E	
478 to 479	CH F	
480 to 481	CH G	
482 to 483	CH H	

484 to 485	CH 1	- CHANNEL ENABLE FLAGS are two byte variables which turn
486 to 487	CH 2	the corresponding channels on and off.
488 to 489	CH 3	
490 to 491	CH 4	1 = Channel ON
492 to 493	CH 5	-1 = Channel OFF
494 to 495	CH 6	
496 to 497	CH 7	
498 to 499	CH 8	
500 to 501	CH A	
502 to 503	CH B	
504 to 505	CH C	
506 to 507	CH D	
508 to 509	CH E	
510 to 511	CH F	
512 to 513	CH G	
514 to 515	CH H	

516 to 517	CH 1	DISPLAY ENABLE FLAGS are two byte flags which enable or
518 to 519	CH 2	disable the corresponding channel's graphics display.
520 to 521	CH 3	Note that a channel can still be active (acquires data)
522 to 523	CH 4	even though the graphics display has been disabled.
524 to 525	CH 5	
526 to 527	CH 6	1 = Channel display ON
528 to 529	CH 7	-1 = Channel display OFF
530 to 531	CH 8	
532 to 533	CH A	
534 to 535	CH B	
536 to 537	CH C	
538 to 539	CH D	
540 to 541	CH E	
542 to 543	CH F	
544 to 545	CH G	
546 to 547	CH H	

Parameter file definition (continued)

548 to 549 DISP.S - Display offset pointer. 16 bit value (LSB,MSB) is number of samples from the start of the data buffer to the first data point shown on the graphics display.

550 to 551 CH.NUM - Number of channels enabled.

552 to 553 H.SCL - Horizontal display expansion/contraction as a power of two (2's complement number). For Example: a waveform contraction of 4 would be represented by the 2's complement number -2. An expansion of 8 would be represented as the number +3.

554 to 555 DLY - Post trigger delay in number of data samples per channel. Note that the trigger point is the buffer length minus the post trigger delay.

556 to 557 T.SCL - Sample interval time scale.
(0- usec. 1 - msec. 2 - sec.)

558 to 559 T.VAL - Sample interval magnitude (001 to 999)

560 to 561 TRG.CH - Trigger channel (1 to 16 if internal, 0 if external)

562 to 563 MODE - Trigger mode; 0 - level 1 - slope
2 - threshold 3 - always

564 T.SIGN - Trigger polarity (0 - negative, 1 - positive)

565 DIS.M - Display mode (1 = dot mode, -1 = fill mode)

566 GRID.F - Background grid flag (1 - grid ON, 0 - grid OFF)

567 to 568 VC.LF - Left vertical cursor position on screen (0 to 499).

569 to 570 VC.RT - Right vertical cursor position on screen (0 to 499).

571 VC.CUR - Active cursor (0 - left, 1 - right).

572 to 573 LF.OFF - Left cursor offset within current sample (offset to the selected channel for multichannel data). 0 - first channel, 2 - second channel, 4 - third channel, etc..

574 to 575 RT.OFF - Right cursor offset within current sample.

576 V.REST - Reset vertical cursors to left edge of display.
(1 - RESET, 0 - no action)

577 V.INIT - Force vertical cursor initialization sequence.
(1 - initialize, 0 - no action)

578 to 579 R.SIZE - Active scope buffer size (on board RAM)
0 - 64K 1 - 32K 2 - 16K 3 - 8K 4 - 4K
5 - 2K 6 - 1K 7 - 1/2K 8 - 1/4K

580 EXT.M - Extended acquisition mode.
0 - Scope mode
1 - Triggered mode
2 - Continuous mode

581 to 582 OB.INX - Observe index. 16 bit value is the index into the data buffer in 1K increment to be observed.

583 to 584 VC.SCL - User defined vertical cursor scale with respect to 0.1V (1 - 999)
1 = 0.1V
999 = 99.9V

585 VC.NAM - User defined vertical cursor name represents the unit measurements.

586 EX.CLK - External clock flag is the flag for internal/external clock.
1 - External flag
2 - Internal flag

587 to 588 AVE.SWP - Total amount of sweeps for averager.

589 to 590 AVE.UPD - Update rate for averager in number of sweeps.

591 to 592 AVE.ACQ - Amount of sweeps acquired by averager.

593 to 594 AVE.REJ - Amount of sweeps rejected by averager.

595 AVE.MAG - Magnifier in powers of two for averager.
(0 - x1. 1 - x2. 2 - x4. 3 - x8, 4 - x16,
5 - x32, 6 - x64, 7 - x128).

599 ART.F - Artifact rejection flag enables or disables artifact detection for averager.
1 - Enable artifact detection
2 - Disable artifact detection

600 to 601 A.DIST - Starting position for the artifact detection in number of sampling points.

602 to 603 A.DISR - Ending position for the artifact detection in number of sampling points.

604 to 605 A.DISU - Upper rejection voltage for artifact detection (12 bit value)

606 to 607 A.DISL - Lower rejection voltage for artifact detection (12 bit value)

608 A.INIT - Force averager initialization sequence
(1 - Initialize, 0 - No action)

609	A.ZERO	- Autozeroing flag enable or disable autozeroing capabilities for eliminating DC offset. (1 - Autozero, 0 - No action)
610	V.RANG	- Voltage range flag for altering voltage input scale. (0 - =/-10V, 1 - +/-2.5V)
611 to 612	CH 1	- Reference enable flags are two byte flags which set the corresponding channel to be or not to be a reference channel. 1 = Reference channel 0 = Not a reference channel
613 to 614	CH 2	
615 to 616	CH 3	
617 to 618	CH 4	
619 to 620	CH 5	
621 to 622	CH 6	
623 to 624	CH 7	
625 to 626	CH 8	
627 to 628	CH A	
629 to 630	CH B	
631 to 632	CH C	
633 to 634	CH D	
635 to 636	CH E	
637 to 638	CH F	
639 to 640	CH G	
641 to 642	CH H	
643	MATH.F	- Mathpack flag. Enable or disable mathack function. 1 - enable mathpack 0 - disable mathpack
644 to 645	M.SUM	- Summation channel for mathpack.
646 to 647	M.OP1	- First operand channel for summation.
648 to 649	M.OP2	- Second operand channel for summation.
650	M.SGN1	- Sign for the first operand of mathpack. (1 - Negative, 0 - Positive)
651	M.SGN2	- Sign for second operand of mathpack. (1 - Negative, 0 - Positive)
652 to 1024		***Reserved for future expansion***

DATA FILE

Data files are identified by the extension 'dat' attached to the file name and contain the converter results as 12 bit offset binary numbers (000h to FFFh). A value of 000 corresponds to a minus full scale signal (-10 volt) while a value of 4096 (FFFh) corresponds to a plus full scale input (+10 volt). Two bytes (LSB,MSB) are used for each data point with data for multichannel acquisitions interleaved in memory.

The lowest numbered active channel will be the first two bytes of data, followed by the next active channel, etc... For a four channel system with channels 2, 7, 8 and A enabled the data format would be as follows;

LS.2, MS.2, LS.7, MS.7, LS.8, MS.8, LS.A, MS.A, LS.2, MS.2, etc...

Although any number of active channels may be selected, data acquistion is restricted to binary powers (sample 1, 2, 4, 8, 16 channels). Selecting an odd number of channels will cause multiple samples of the first channel to fill the data buffer up to the next higher binary power. For a five channel system with channels 1, 4, 5, 6, and 7 enabled the data format would be as follows;

LS.1, MS.1, LS.4, MS.4, LS.5, MS.5, LS.6, MS.6, LS.7, MS.7
LS.1, MS.1, LS.1, MS.1, LS.1, MS.1,

LS.1, MS.1, LS.4, MS.4, LS.5, MS.5, LS.6, MS.6, LS.7, MS.7
LS.1, MS.1, LS.1, MS.1, LS.1, MS.1, etc...

Note that 8 data samples are acquired during each cycle since this is the next higher power of 2 greater than the number of selected channels (5).

ARCHIVE FILES

Archive files are identified by the extension 'cmp' and contain the same information as 'dat' files, but in a compressed format. A compression of 33% is acheived by packing two 12 bit data points into three 8 bit bytes. In addition, when the number of active channels does not equal a power of two, the repetitive samples used to fill the sample cycle are also eliminated (see 5 channel example above).

B.2.4.3 *.RES ASCII File Format

m

dt

typ

data pt 1 (a_1)

data pt 2 (a_2)

⋮

where: m = number of intervals

dt = Δt (in ms)

typ = type of waveform

0 = nonstandard

1 = halfsine

2 = triangle

3 = square

B.2.4.4 *.CH1, *.CH2, *.CH3, *.PLT ASCII FILE FORMAT

data pt 1 (t_1, a_1)

data pt 2 (t_2, a_2)

⋮

B.3 Using The Computerscope With The HIC Calculation Software

Three channels of Computerscope data, corresponding to a_x, a_y, and a_z, should be collected. These channels should be one of the first eight on the instrument interface. After data acquisition and prior to executing the HIC software, the Computerscope program must be executed and the starting and ending points of the acceleration pulses marked. The HIC software expects, as input, the three channels of cursored Computerscope data. The software eliminates any dc offset in the three acceleration channels, multiplies each channel by its calibration constant (g's/volt), and calculates a_r, the resultant acceleration profile:

$$a_r = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

HICCNVRT performs the initial data reduction and Computerscope to ASCII conversion for subsequent HIC calculations in HIC_BAS and HIC_C. HIC_PAS performs the initial data reduction, the Computerscope to ASCII file conversion, and the HIC calculations. This data is saved as

$$< filename > .res$$

Both HIC_PAS and HICCNVRT also create ACSII files of the original data and resultant acceleration profiles in a different format for subsequent plotting, with software like STATGRAPHICS, etc. This data is saved as

$$< filename > .ch1$$

$$< filename > .ch2$$

$$< filename > .ch3$$

$$< filename > .plt$$

B.4 Data Resolution And Program Limitations

In order to satisfy the specifications of SAE Recommended Standard J211b, Channel Class 1000, an 8 KHz sampling rate is suggested (i.e., 8 points per ms). In a typical helmet drop test, the acceleration pulse duration is less than 50 ms. Therefore, a maximum of 400 data points are required. Since the Computerscope can capture up to 8192 data points and the HIC programs can analyze up to 8192 data points, the resolution of the proposed data acquisition and analysis system is more than adequate. It is important to note that if 200 data points will suffice, a 1K buffer is recommended, since the calculation time increases drastically as the number of points increases.

In the HIC_PAS and HIC_C programs, the maximum value of m, the number of intervals, is 10240 (10K). This limitation is due to the fact that the Pascal and C languages permit a data segment of 64K and require 6 bytes to store each real number. Thus, the maximum number of real numbers that can be stored is $\approx$11K. Since the Computerscope produces buffer dumps of size 2^nK, where $0 \leq n \leq 6$, the programs were written to handle 8192 (8K) data points. If the Computerscope is not utilized, m can be increased to $\approx$10K. In the HIC_BAS program, the maximum value of m is 8K; the Basic language permits 64K total for both the source code and the data segment.

B.5 Program Validations

Validations are provided for the HIC programs. These utilize acceleration-time histories, which were generated by a Krohn-Hite Function Generator and collected and digitized by the Computerscope data acquisition system. First, using three identical triangular waveforms as input, results for t_1, t_2, the SI, and HIC were compared from each of the three methods in each of the three languages. Next, calculated HIC values were compared to theoretical results. Since closed-form solutions for the HIC are only available for simple waveforms, the input acceleration pulses were triangle, square, and half-sine waveforms. The exact HIC values, which are functions of the pulse height and duration, were abstracted from a study by Chou and Nyquist (1974). Also, the relationships between the HIC and SI were compared to theoretical values for the three input waveforms. In addition, the accelerations at t_1 and t_2, which theoretically are equal, were compared. Finally, the ASCII file outputs were validated using the triangular waveform as input.

```
Parameters:
                        Method: 3 (HIC_PAS)
                    Input File: cscop.dat\trihic4.res
              ComputerScope File: NO

SI = 4064.131142

Calculating HIC using PARTIAL SUMS/SLIDING ENDPOINTS METHOD...

       Waveform type = triangle

 no. of subintervals =       690
     delta time (dt) =         0.000200 sec.
   peak acceleration =       101.267466 Gs

                  T1 =         0.027400 sec.
                  T2 =         0.105800 sec.

      EXACT SOLUTION =      3509.09113
      CALCULATED HIC =      3504.74227

    Percent error =         0.123931
```

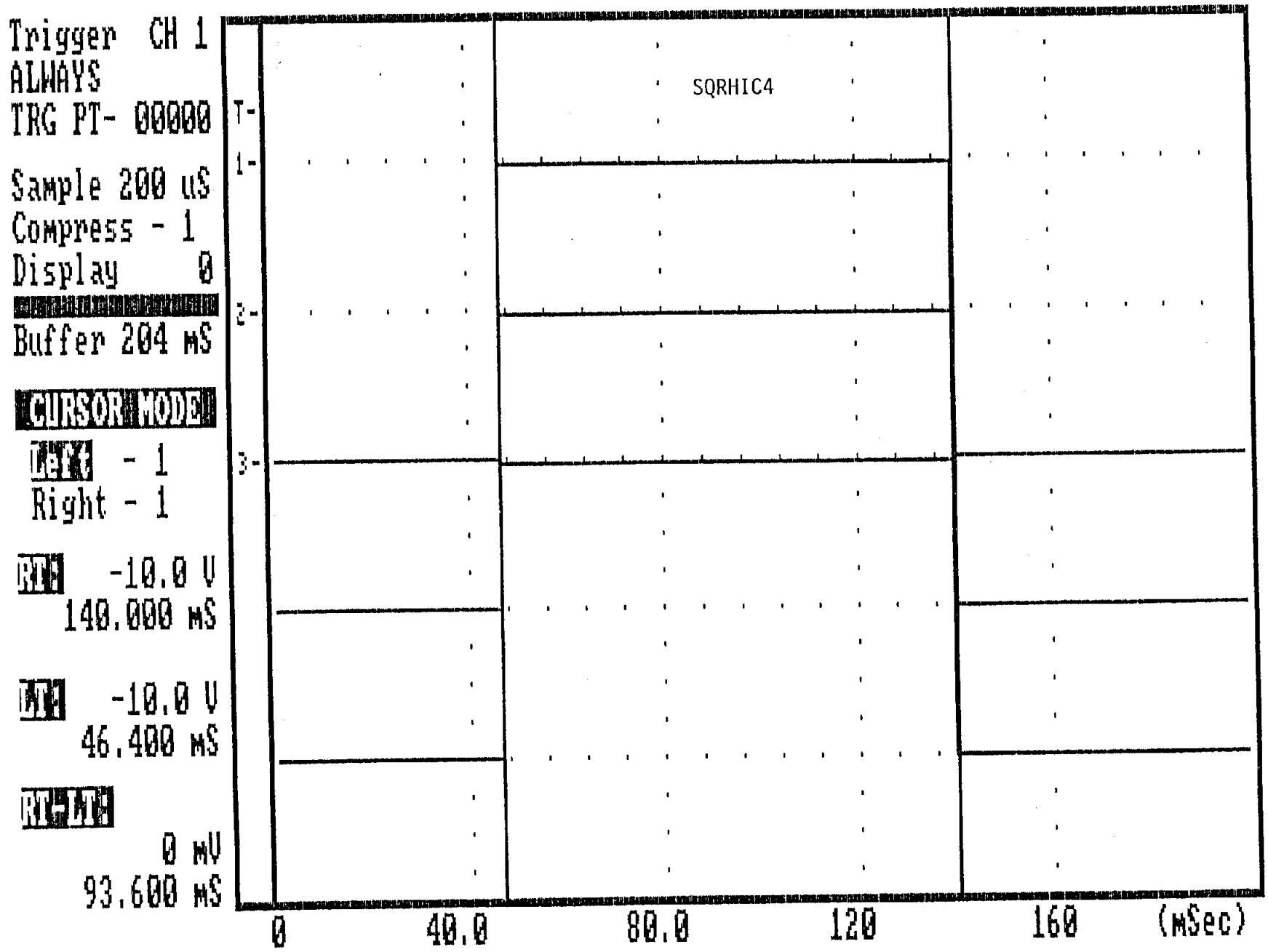
```
                    Method: 3 (HIC_C)
                Input File: cscop.dat\trihic4.res

Reading Input Data - Please Wait...

Severity Index = 4064.091097

Calculating HIC using PARTIAL SUMS/SLIDING ENDPOINTS METHOD...

                  waveform =  triangle

        no. of subintervals =  690
          delta time (msec) =  0.0002
          peak acceleration =  101.267464

                         T1 =  0.0274
                         T2 =  0.1058

            EXACT SOLUTION = 3509.09082
            CALCULATED HIC =  3504.741211

             percent error =  0.123953
```

```
Reading Input Data - Please wait.

SI =  4064.129

Calculating HIC using PARTIAL SUMS/SLIDING ENDPOINTS METHOD... (HIC_BAS)

 Waveform type = triangle
             m =      690
            dt =        0.000200 sec.
     peak acc. =      101.267500 Gs

            T1 =        0.0272000 msec.
            T1 =        0.1060000 msec.

EXACT SOLUTION:      3509.0910
CALCULATED HIC:      3504.7370
 PERCENT ERROR:         0.1241




Parameters:
                        Method: 2 (HIC_PAS)
                    Input File: trihic4.res
            ComputerScope File: NO

SI = 4064.131142

Calculating HIC using PARTIAL SUMS METHOD...

       Waveform type = triangle

 no. of subintervals =       690
    delta time (dt) =         0.000200 sec.
  peak acceleration =       101.267466 Gs

                 T1 =         0.027400 sec.
                 T2 =         0.105800 sec.

      EXACT SOLUTION =      3509.09113
      CALCULATED HIC =      3504.74227

   Percent error =         0.123931
```

```
                    Method: 2 (HIC_C)
                Input File: trihic4.res

Reading Input Data - Please Wait...

Severity Index = 4064.091097

Calculating HIC using PARTIAL SUMS METHOD...

                      waveform =  triangle

          no. of subintervals =  690
            delta time (msec) =  0.0002
            peak acceleration =  101.267464

                           T1 =  0.0274
                           T2 =  0.1058

              EXACT SOLUTION = 3509.09082
              CALCULATED HIC =  3504.741211

               percent error =  0.123953




Reading Input Data - Please wait.

SI =  4064.129


Calculating HIC using PARTIAL SUMS METHOD... (HIC_BAS)

 Waveform type = triangle
             m =      690
            dt =        0.000200 sec.
     peak acc. =      101.267500 Gs

            T1 =        0.0274000 msec.
            T1 =        0.1058000 msec.

EXACT SOLUTION:      3509.0910
CALCULATED HIC:      3504.7380
 PERCENT ERROR:         0.1241
```

```
Parameters:
                          Method: 1 (HIC_PAS)
                      Input File: trihic4.res
              ComputerScope File: NO

SI = 4064.131142

Calculating HIC using BRUTE FORCE METHOD...

       Waveform type = triangle

 no. of subintervals =       690
     delta time (dt) =         0.000200 sec.
   peak acceleration =       101.267466 Gs

                  T1 =         0.027400 sec.
                  T2 =         0.105800 sec.

      EXACT SOLUTION =      3509.09113
      CALCULATED HIC =      3504.74227

    Percent error =         0.123931
```

```
19:04:40.04>hic_c /ml /ftrihic4.res
Parameters:
                    Method: 1
                Input File: trihic4.res

Reading Input Data - Please Wait...

Severity Index = 4064.091097

Calculating HIC using BRUTE FORCE METHOD...

                   waveform =  non-standard

        no. of subintervals =  690
          delta time (msec) =  0.0002
          peak acceleration =  101.267464

                         T1 =  0.0274
                         T2 =  0.1058

            EXACT SOLUTION: cannot be found.
            CALCULATED HIC =  3504.741211
```


Trigger CH 1
ALWAYS
TRG PT- 00000
Sample 200 uS
Compress - 1
Display 0
Buffer 204 mS
CURSOR MODE
LeFt - 1
Right - 1
RT -10.0 V
140.000 mS
LT -10.0 V
46.400 mS
RT-LT
0 mV
93.600 mS
SQRHIC4
T-
1-
2-
3-
0
40.0
80.0
120
160
(mSec)

```
Input file name without extension: SQRHIC4
Subtract pre-start average as offset? <YES>
Include output in ASCII format for channels and resultant? <NO> Y

Test for HIC speed with Square

Channel_Name 1: Square 1
Channel_Name 2: Square 2
Channel_Name 3: Square 3

Time to start output:    0.046400 sec.
Time to  stop output:    0.140000 sec.

Total Points per Channel:   1024  -->  Points between cursors: 468
Fraction denominator of output points: <1>
Calibration Factor for Channel 1 (user units/volt)? 6
Calibration Factor for Channel 2 (user units/volt)? 6
Calibration Factor for Channel 3 (user units/volt)? 6

Channel 1 offset: -10.000000
Channel 2 offset: -10.000000
Channel 3 offset: -10.000000
```

Parameters:

Method: 3
Input File: CSCOP.DAT\SQRHIC4.RES
ComputerScope File: NO

SI = 10009.210867

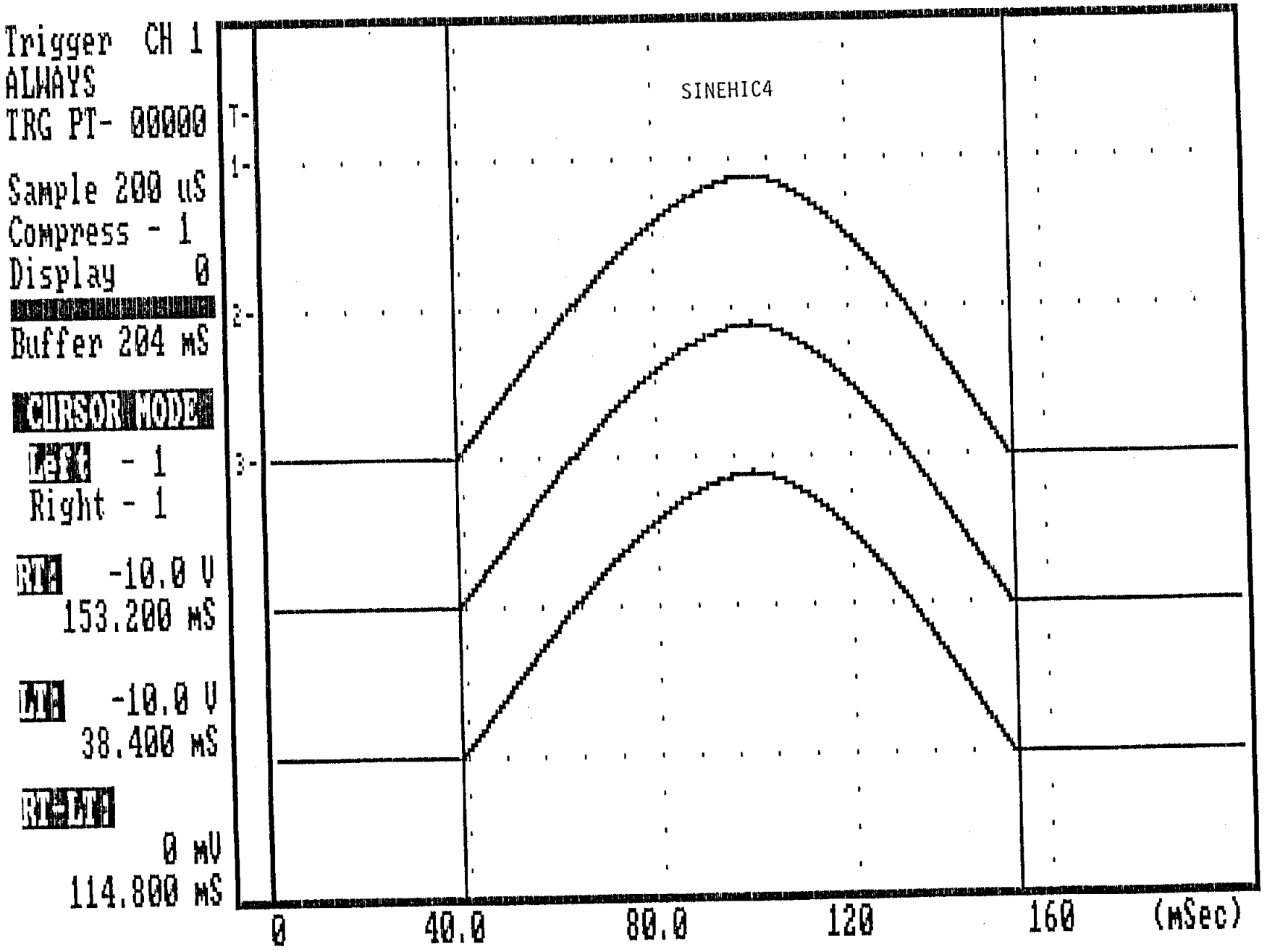
Calculating HIC using PARTIAL SUMS/SLIDING ENDPOINTS METHOD...

Waveform type = square

no. of subintervals =	468
delta time (dt) =	0.000200 sec.
peak acceleration =	103.161902 Gs
T1 =	0.000400 sec.
T2 =	0.093400 sec.
EXACT SOLUTION =	10117.52300
CALCULATED HIC =	9987.70128
Percent error =	1.283137

From Chou: HIC=SI
Percent Error=0.22

From Chou: $a(t_1)=a(t_2)$
$a(t_1)$=102.739 Gs
$a(t_2)$=103.162 Gs
Percent Error=0.41


Trigger CH 1
ALWAYS
TRG PT- 00000
Sample 200 uS
Compress - 1
Display 0
Buffer 204 mS
CURSOR MODE
Left - 1
Right - 1
RT -10.0 V
153.200 mS
LT -10.0 V
38.400 mS
RT-LT
0 mV
114.800 mS
SINEHIC4
T-
1-
2-
3-
0
40.0
80.0
120
160
(mSec)

```
Input file name without extension: SINEHIC4
Subtract pre-start average as offset? <YES>
Include output in ASCII format for channels and resultant? <NO> Y

Test for HIC speed with Sine

Channel_Name 1: Sine 1
Channel_Name 2: Sine 2
Channel_Name 3: Sine 3

Time to start output:    0.038400 sec.
Time to  stop output:    0.153200 sec.

Total Points per Channel:   1024  -->  Points between cursors: 574
Fraction denominator of output points: <1>
Calibration Factor for Channel 1 (user units/volt)? 6
Calibration Factor for Channel 2 (user units/volt)? 6
Calibration Factor for Channel 3 (user units/volt)? 6

Channel 1 offset: -10.000000
Channel 2 offset: -10.000000
Channel 3 offset: -10.000000
```

Parameters:

Method: 3
Input File: CSCOP.DAT\SINEHIC4.RES
ComputerScope File: NO

SI = 4666.408607

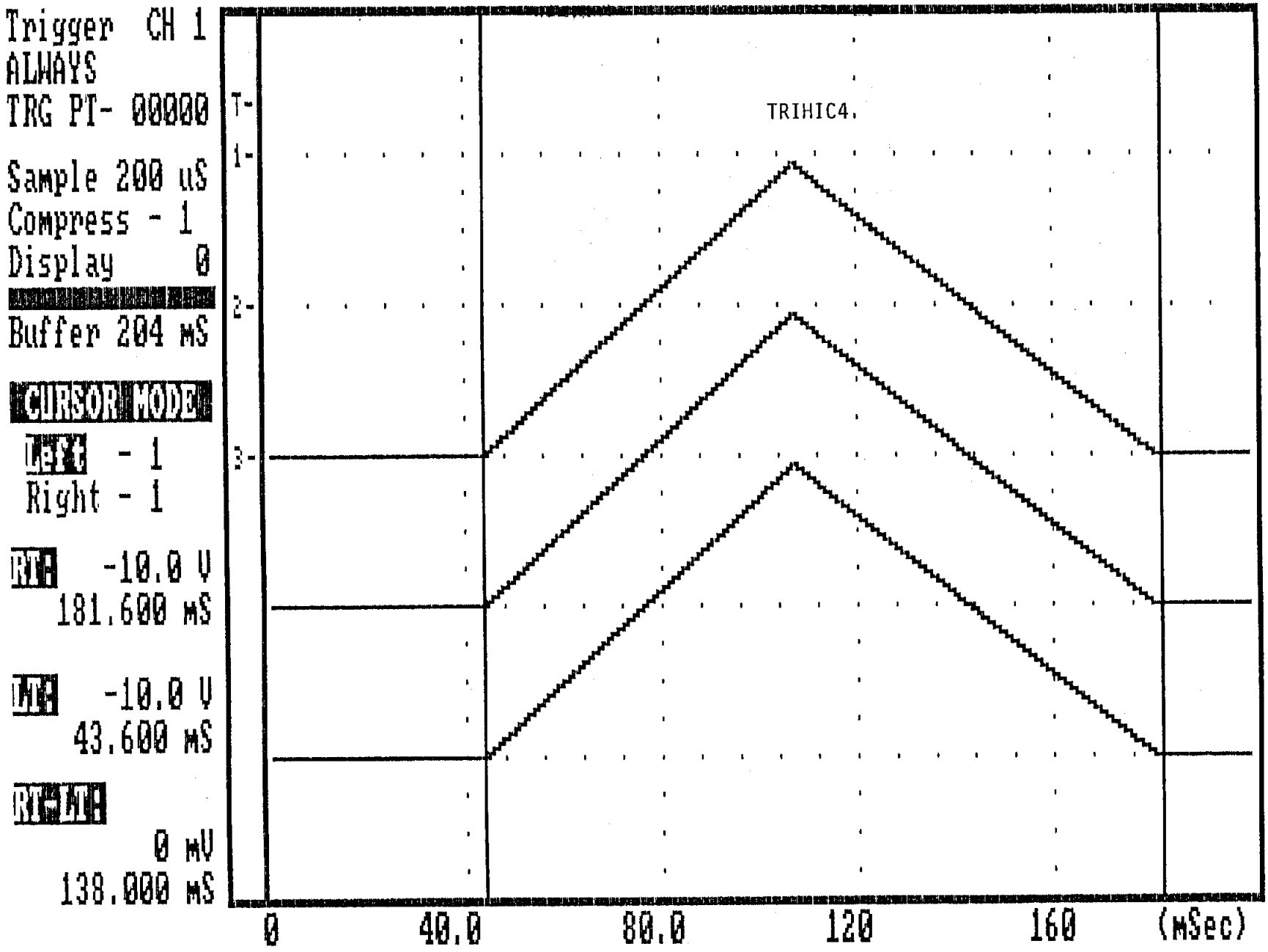
Calculating HIC using PARTIAL SUMS/SLIDING ENDPOINTS METHOD...

Waveform type = half-sine

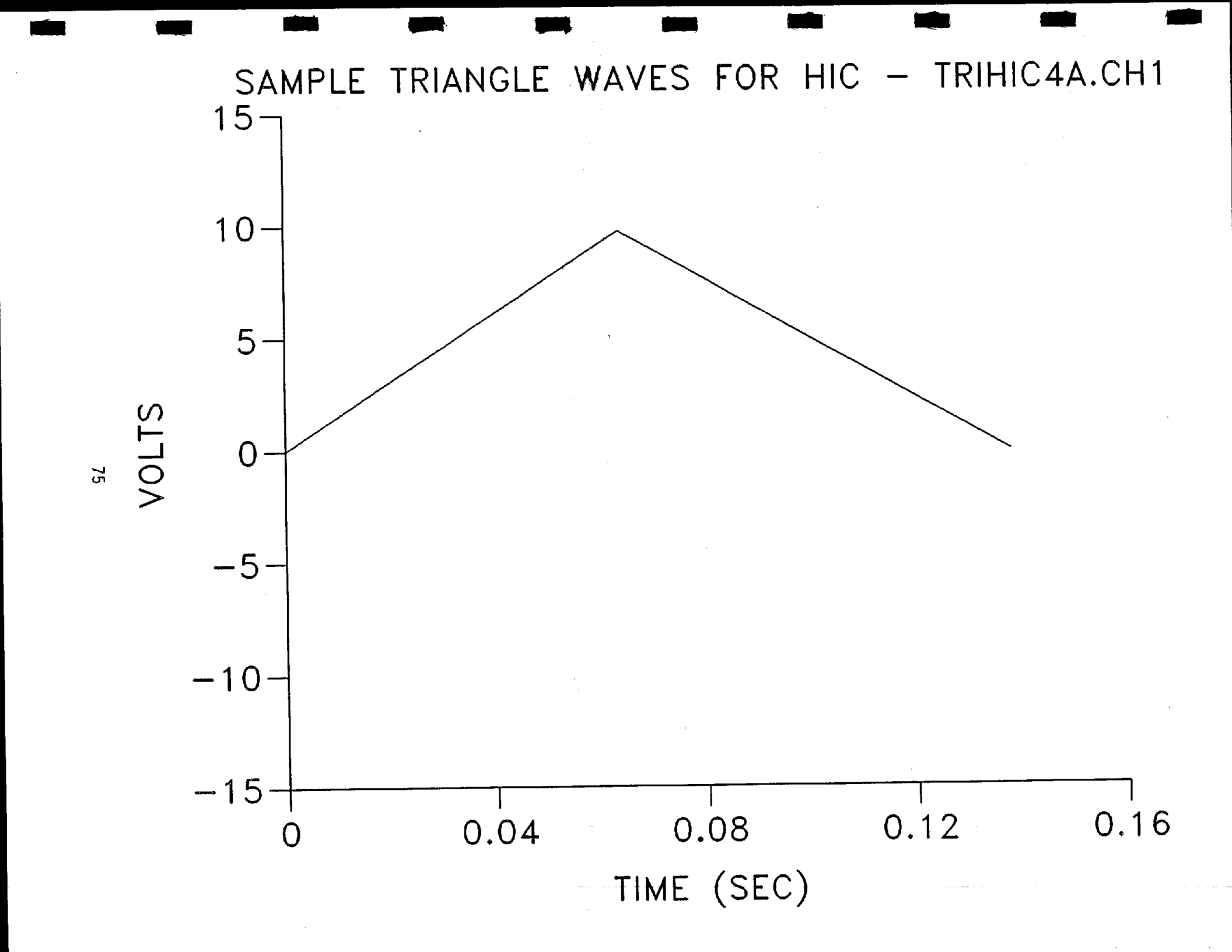
no. of subintervals	=	574
delta time (dt)	=	0.000200 sec.
peak acceleration	=	99.051664 Gs
T1	=	0.022000 sec.
T2	=	0.095800 sec.
EXACT SOLUTION	=	4647.56664
CALCULATED HIC	=	4191.59729
Percent error	=	9.810927

From Chou: HIC=0.9058(SI)=4226.91
Percent Error=0.84

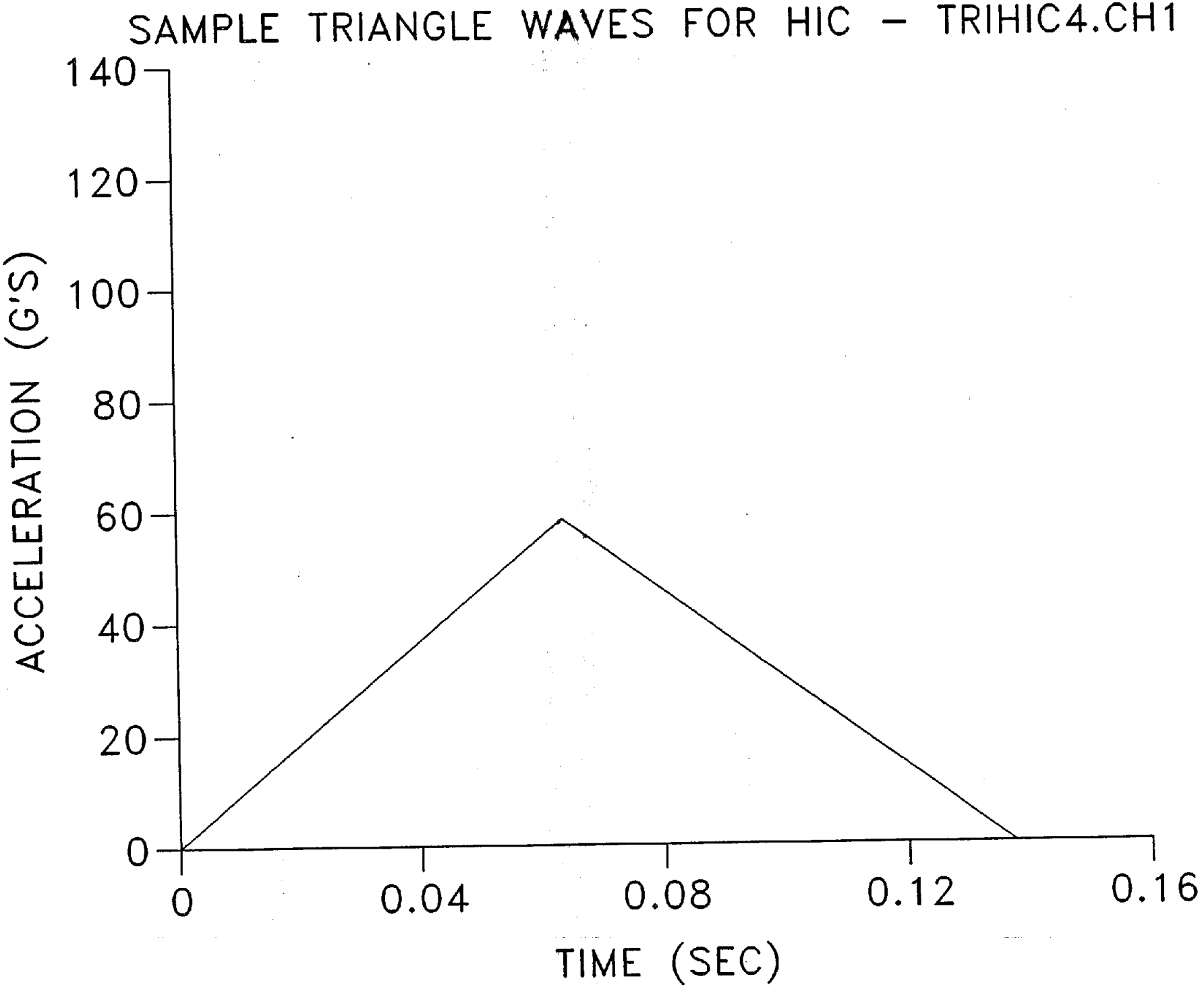
From Chou: $a(t_1)=a(t_2)$
$a(t_1)$=47.733 Gs
$a(t_2)$=47.784 Gs
Percent Error=0.11

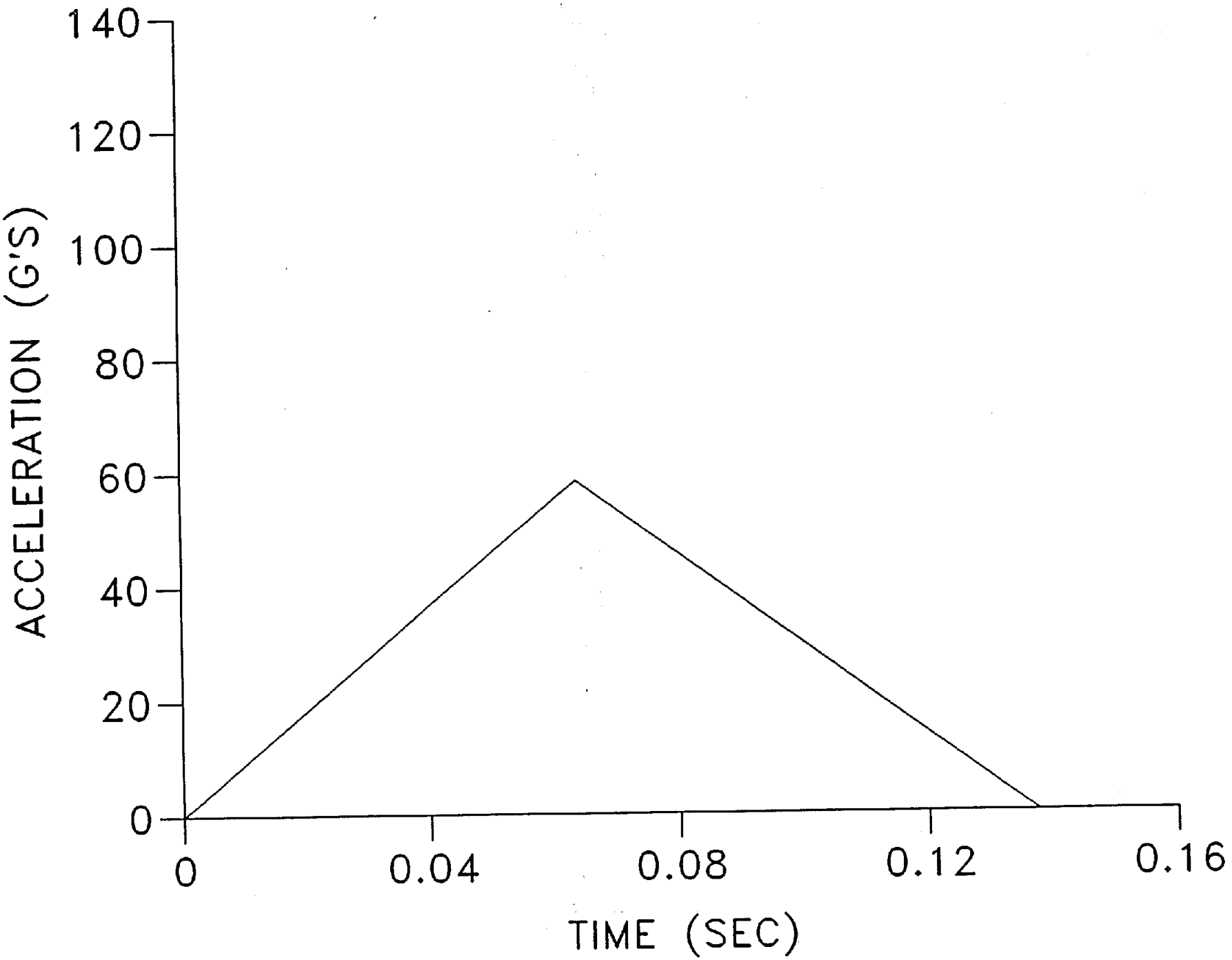

Trigger CH 1
ALWAYS
TRG PT- 00000
Sample 200 uS
Compress - 1
Display 0
Buffer 204 mS
CURSOR MODE
LFT - 1
Right - 1
RT -10.0 V
181.600 mS
LT -10.0 V
43.600 mS
RT-LT
0 mV
138.000 mS
TRIHIC4.
T-
1-
2-
3-
0
40.0
80.0
120
160
(mSec)

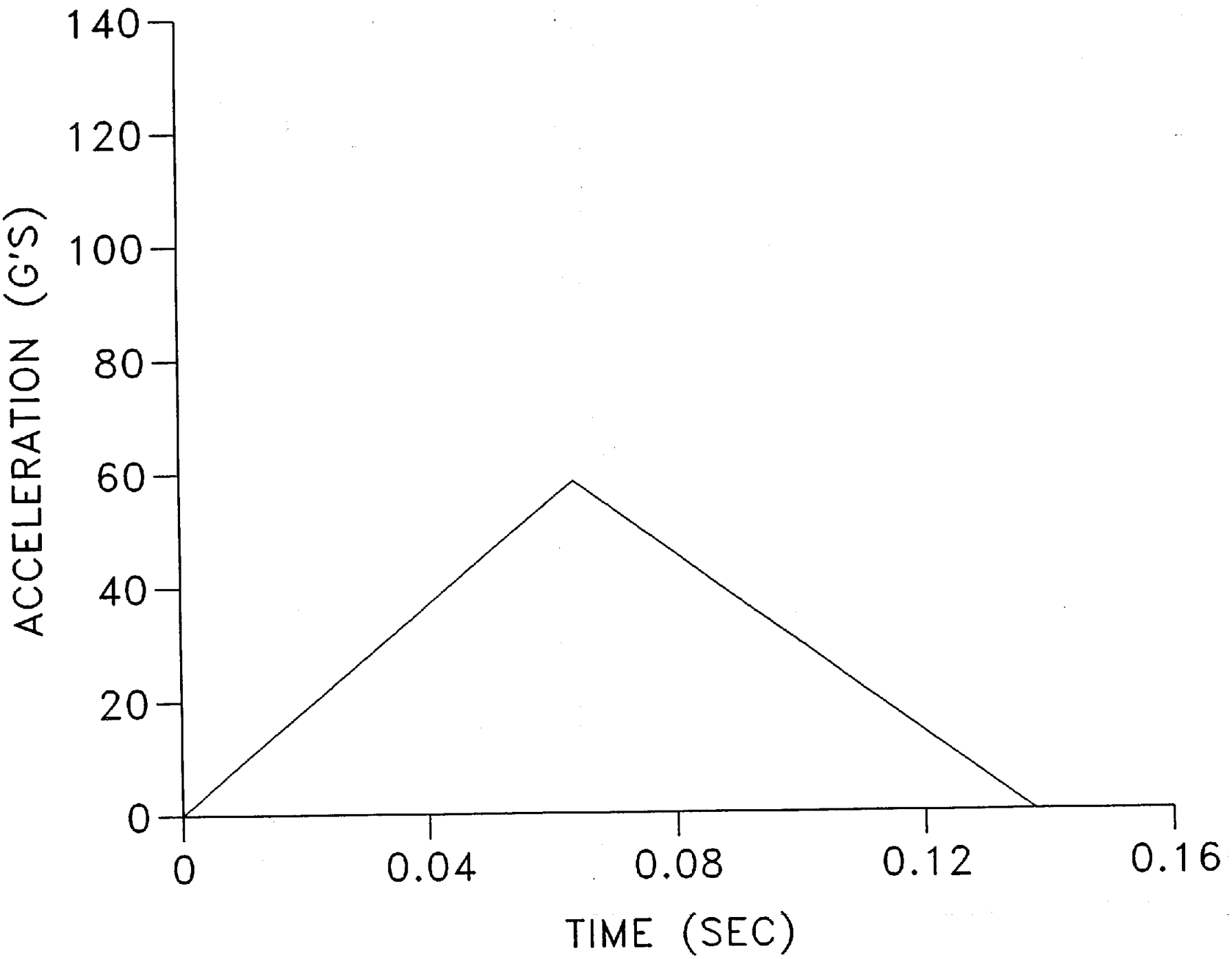
```
Input file name without extension: TRIHIC4A
Subtract pre-start average as offset? <YES>
Include output in ASCII format for channels and resultant? <NO> Y

Test for HIC speed with Triangle

Channel_Name 1: Triangle 1
Channel_Name 2: Triangle 2
Channel_Name 3: Triangle 3

Time to start output:    0.043600 sec.
Time to  stop output:    0.181600 sec.

Total Points per Channel:   1024  -->  Points between cursors: 690
Fraction denominator of output points: <1>
Calibration Factor for Channel 1 (user units/volt)? 1
Calibration Factor for Channel 2 (user units/volt)? 1
Calibration Factor for Channel 3 (user units/volt)? 1

Channel 1 offset: -10.000000
Channel 2 offset: -10.000000
Channel 3 offset: -10.000000
```

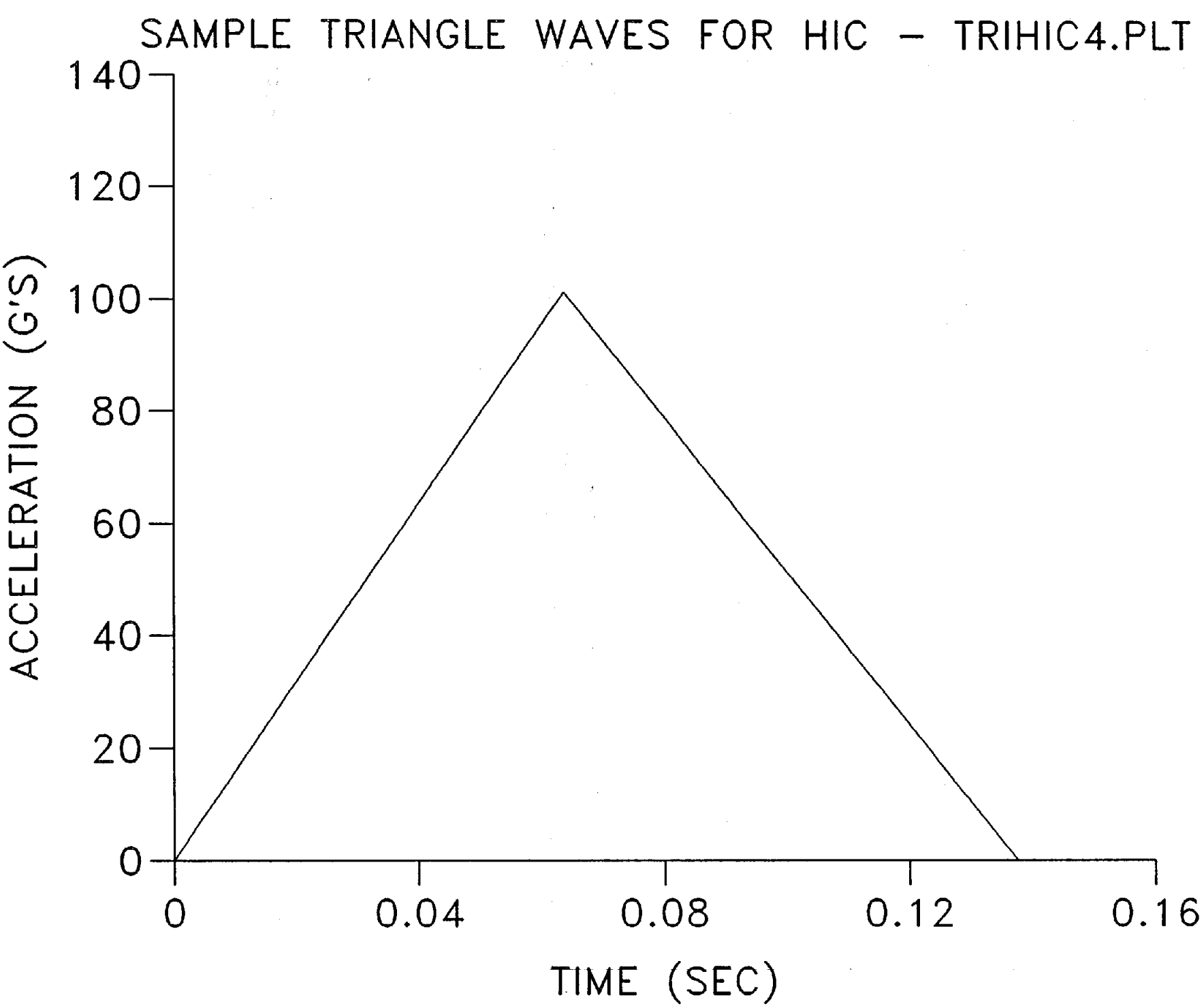

SAMPLE TRIANGLE WAVES FOR HIC – TRIHIC4A.CH1
15
10
5
0
−5
−10
−15
VOLTS
0
0.04
0.08
0.12
0.16
TIME (SEC)

```
Input file name without extension: TRIHIC4
Subtract pre-start average as offset? <YES>
Include output in ASCII format for channels and resultant? <NO> Y

Test for HIC speed with Triangle

Channel_Name 1: Triangle 1
Channel_Name 2: Triangle 2
Channel_Name 3: Triangle 3

Time to start output:    0.043600 sec.
Time to  stop output:    0.181600 sec.

Total Points per Channel:   1024  -->  Points between cursors: 690
Fraction denominator of output points: <1>
Calibration Factor for Channel 1 (user units/volt)? 6
Calibration Factor for Channel 2 (user units/volt)? 6
Calibration Factor for Channel 3 (user units/volt)? 6

Channel 1 offset: -10.000000
Channel 2 offset: -10.000000
Channel 3 offset: -10.000000
```


SAMPLE TRIANGLE WAVES FOR HIC - TRIHIC4.CH1
ACCELERATION (G'S)
140
120
100
80
60
40
20
0
0
0.04
0.08
0.12
0.16
TIME (SEC)


SAMPLE TRIANGLE WAVES FOR HIC – TRIHIC4.CH2
ACCELERATION (G'S)
140
120
100
80
60
40
20
0
0
0.04
0.08
0.12
0.16
TIME (SEC)


SAMPLE TRIANGLE WAVES FOR HIC - TRIHIC4.CH3
ACCELERATION (G'S)
140
120
100
80
60
40
20
0
0
0.04
0.08
0.12
0.16
TIME (SEC)


SAMPLE TRIANGLE WAVES FOR HIC - TRIHIC4.PLT
ACCELERATION (G'S)
140
120
100
80
60
40
20
0
0
0.04
0.08
0.12
0.16
TIME (SEC)

```
Parameters:
                        Method: 3
                    Input File: CSCOP.DAT\TRIHIC4.RES
              ComputerScope File: NO

SI = 4064.131142

Calculating HIC using PARTIAL SUMS/SLIDING ENDPOINTS METHOD...

       Waveform type = triangle

 no. of subintervals =       690
     delta time (dt) =         0.000200 sec.
   peak acceleration =       101.267466 Gs

                  T1 =         0.027400 sec.
                  T2 =         0.105800 sec.

      EXACT SOLUTION =      3509.09113
      CALCULATED HIC =      3504.74227

    Percent error =         0.123931
```

From Chou: HIC=0.862(SI)=3503.28
Percent Error=0.042

From Chou: $a(t_1)=a(t_2)$
$a(t_1)=43.589$ Gs
$a(t_2)=43.521$ Gs
Percent Error=0.16

B.6 Program Execution Times

The HIC programs were executed on an IBM PCAT and clone (6-8 MHz clock speed) to measure runtimes. For a typical helmet test duration of 30 msec, an 8 KHz sampling rate requires 240 data points. To simulate a typical scenario, a triangular input waveform was utilized with 230 data points. The Basic version is slowest; the Pascal and C version execution times are similar. For method 3, Pascal and C version runtimes were less than 30 seconds.

C. The Necessity of Multiple Size Headforms/Helmets

C.1 Headform Test Apparatus and Protocol

Drop tests were conducted to assess the necessity of different headform sizes and types in the proposed ANSI Z89.2 standard. Vertical impact energy attenuation tests were performed using the Duke helmet drop test system. For these tests, a continuous THK* linear bearing track, rigidly supported by a 4 inch steel I beam, acted as a free-fall guide for the headform drop assembly.

A helmeted headform was dropped in guided fall onto flat and hemispherical anvils. Five headforms were utilized: an MTS magnesium alloy K1A headform and four ISO headforms (sizes A, E, J, and M). The headforms were ballasted to weigh 11.0 lbs. For comparison, these headform sizes were compared to head sizes of 5th and 50th percentile females and 50th and 95th percentile males (see Table C.1), using GEBOD, an interactive computer program that generates anthropometric data about adults and children. Three helmets were tested: a padded football helmet, an industrial helmet, and a padded baseball helmet. The drop height used for the football and industrial helmet tests was selected to yield an impact energy of 40 ft-lbs. Due to bottoming, the drop height used for the baseball helmet tests was considerably less. A velocity sensing system measured impact velocity; basically, it operates as an interval timer activated by a noncontacting optical sensor for a given displacement of the drop carriage. Acceleration measurements were obtained from a triaxial accelerometer located at the headform center of gravity. Load measurements were obtained from a load cell mounted between the anvil and the base plate.

C.2 Headform Test Results

Figures C.1-C.38 are plots of load-time and SI acceleration-time results for all tests. Table C.2 summarizes the results. The table and individual plots should be used as references for identifying results in the multiple plots.

The effect of headform size and type on drop test performance was investigated first. The results are plotted in Figures C.39-C.48 and tabulated in Table C.3.

Two ISO headforms (sizes A and J) were tested with the industrial and football helmets using the hemispherical impactor. Little differences in pulse shape and duration were observed. Load and acceleration peaks were similar for the industrial helmet. For the football helmet, peak accelerations were similar but higher peak loads were observed with the smaller size headform.

All headforms and helmets were tested with the flat impactor. With the industrial helmet, the MTS headform showed higher loads and accelerations than the ISO headform. Comparing only the ISO headforms, similar pulse shapes and duration were observed. Acceleration peaks were similar. Load peaks increased with increasing head size. With the football helmets, similar pulse shapes and durations were observed for all headforms tested. Neglible differences were observed in the peak loads and accelerations. For the baseball helmets, similar pulse shapes and durations were observed for all headforms. Slight differences in drop heights and clipping make it difficult to compare peak values.

The effect of helmet type on drop test performance was investigated next. The results are plotted in Figures C.49-C.62 and tabulated in Table C.4. With the ISO headforms, football helmets showed higher loads and accelerations than the industrial helmets. With the MTS headform, football helmets performed the same, in terms of peak values, as the industrial helmets. The baseball helmets showed much higher loads and accelerations in all cases. With the hemispherical impactor, pulse durations for the football and industrial helmets were 20-25 msec and 25-30 msec, respectively. With the flat impactor, pulse duration were shorter (approximately 15 msec and 20 msec for the football and industrial

helmets, respectively). Pulse durations for the baseball helmets (approximately 10 msec) were less than pulse durations for both the football and industrial helmets.

The effect of impactor shape on drop test performance was also investigated. The results are plotted in Figures C.63-C.70 and tabulated in Table C.5. Two ISO headforms (sizes A and J) were tested with the industrial and football helmets using the hemispherical and flat impactors. In each case, the peak loads and accelerations were higher and the pulse durations shorter with the flat impactor compared to the hemispherical impactor.

TABLE C.1

Comparison of ISO Head Sizes

	Head Length(in)	Head Breadth (in)
5th percentile fem	7.069	5.603
50th percentile fem	7.247	5.714
50th percentile male	7.823	6.142
95th percentile male	7.969	6.250
A	6.875	5.125
E	7.375	5.75
J	7.875	6.0
M	8.125	6.5

The Necessity of Multiple Size Headforms/Helmets

Industrial Helmet Tests - Size Adjustable - Flat Impactor = ?IAF

Football Helmet Tests - Size Large - Flat Impactor = ?FLF

Baseball Helmet Tests - Size Large - Flat Impactor = ?BLF

Industrial Helmet Tests - Size Adjustable - Hemispherical Impactor = ?IAH

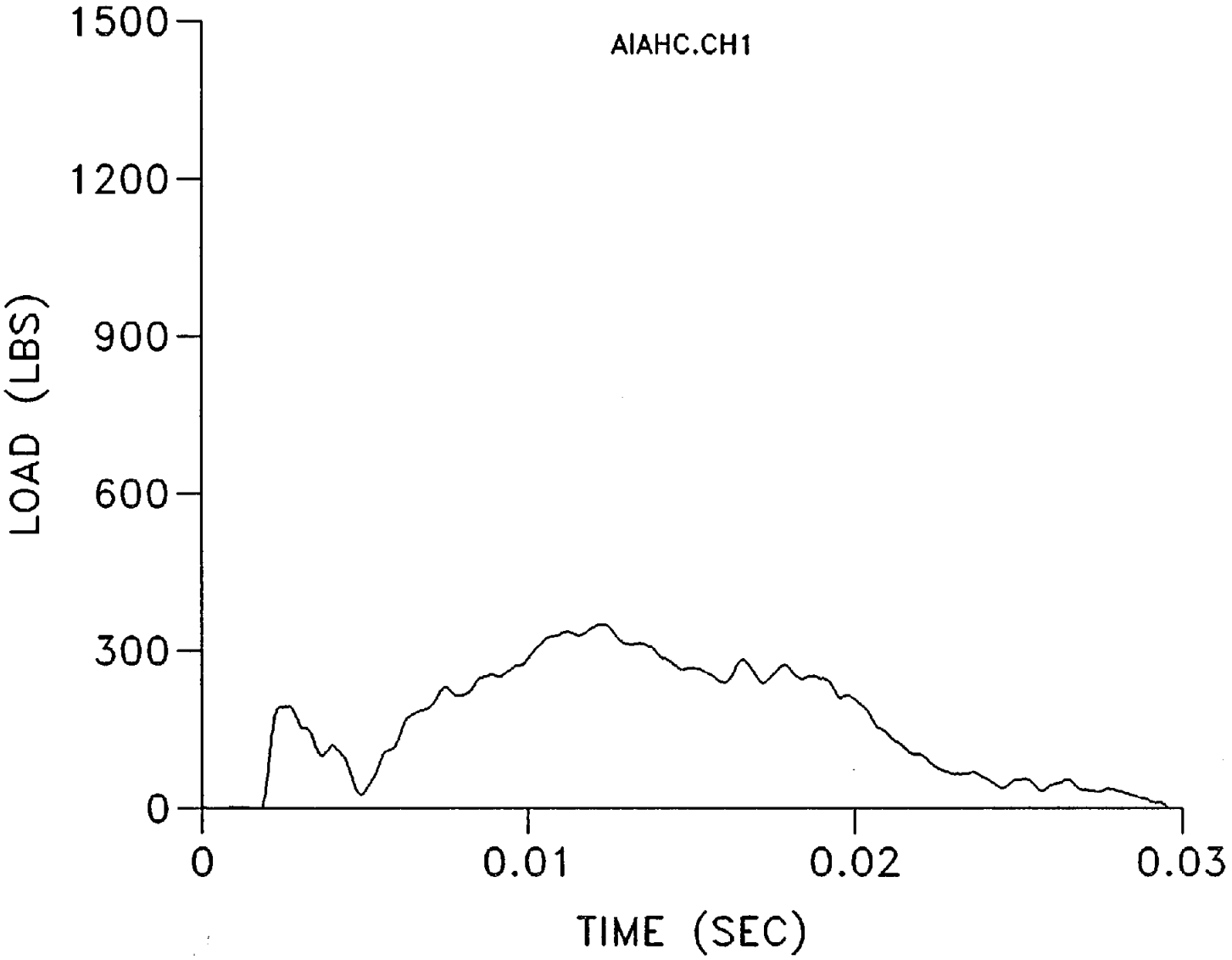
Football Helmet Tests - Size Large - Hemispherial Impactor = ?FLH

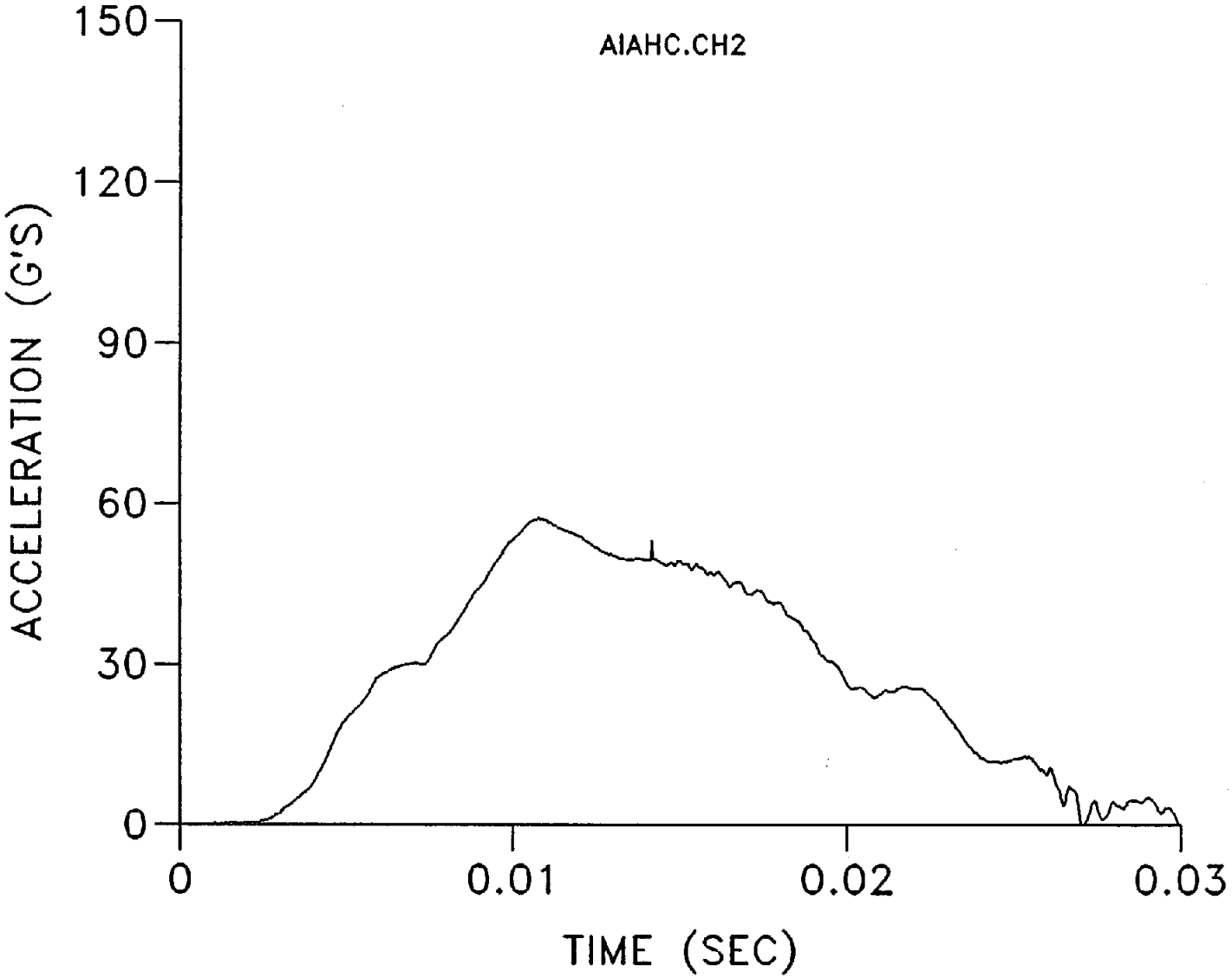
where: ? = Headform = ISO (Sizes A, E, J, M), MTS

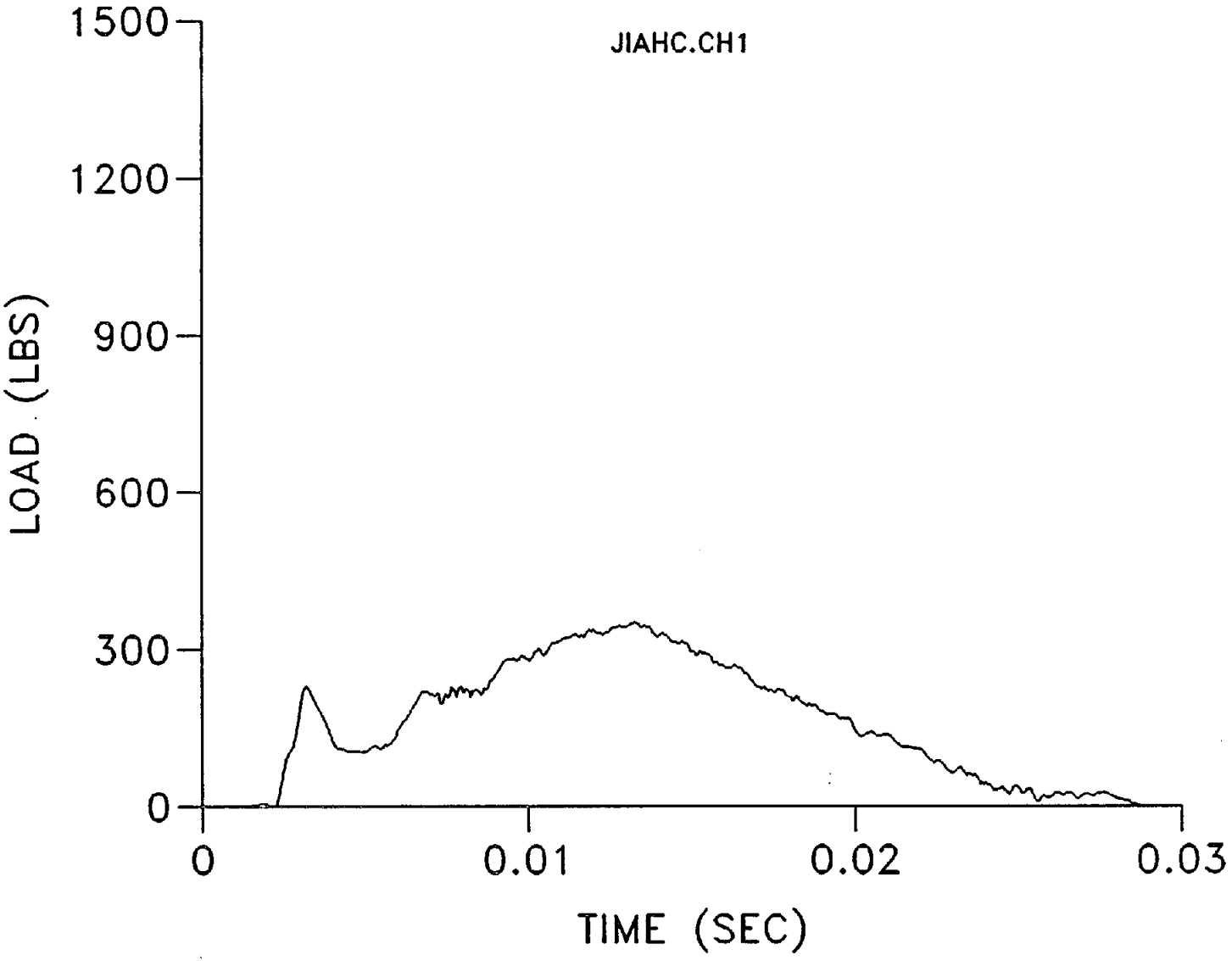
TABLE C.2

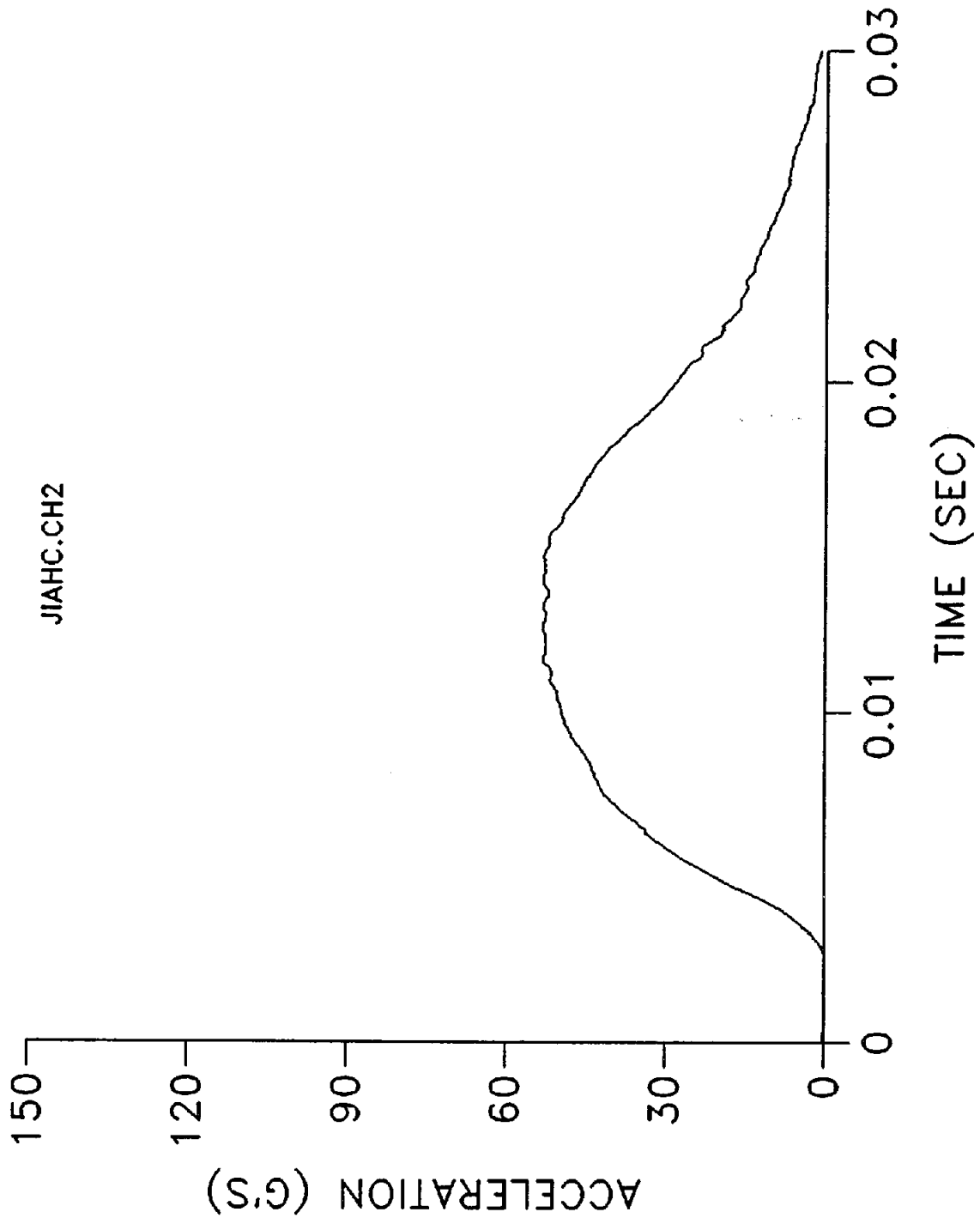
THE NECESSITY OF MULTIPLE SIZE HEADFORMS/HELMETS

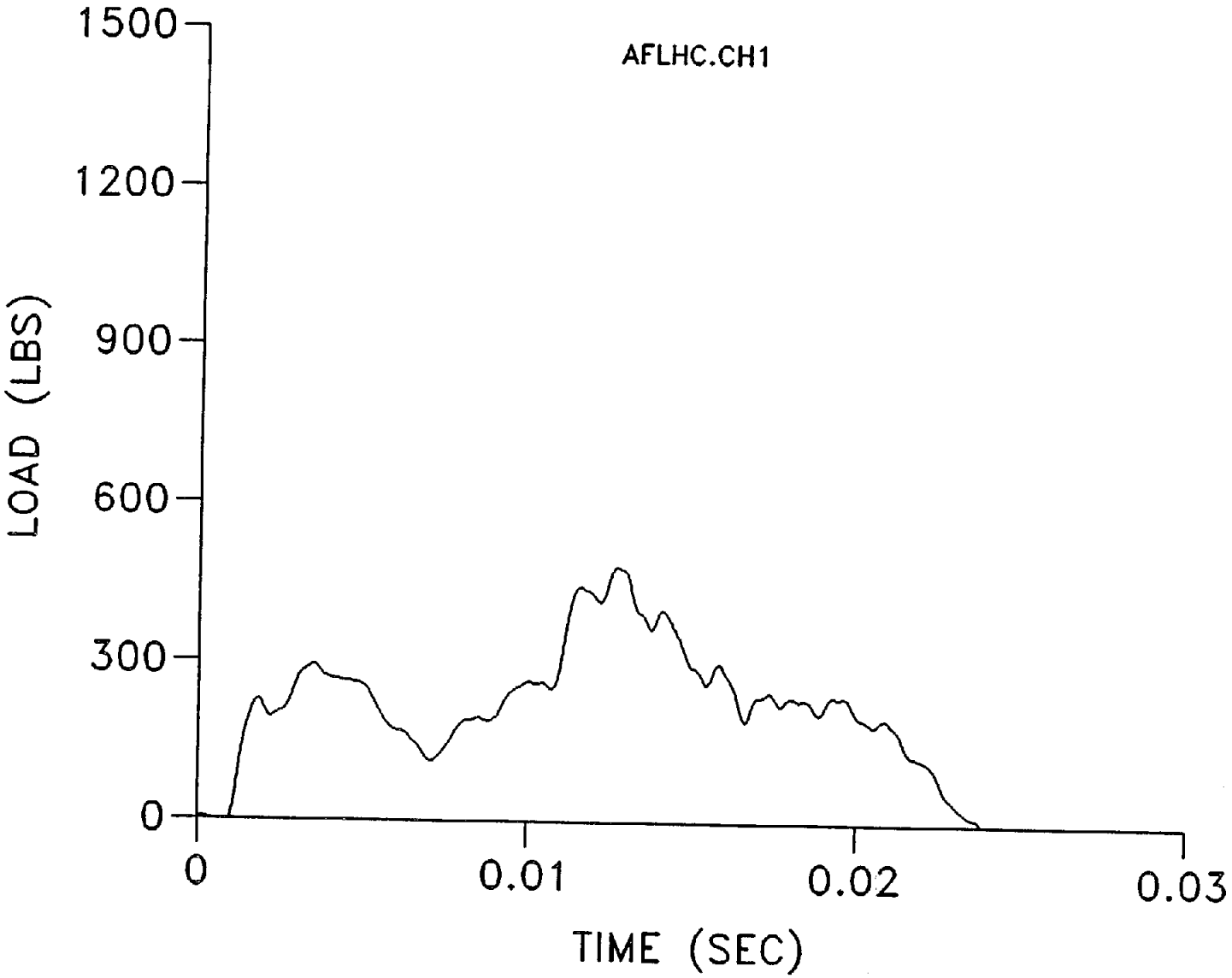
		CHANNEL 1					CHANNEL 2				
FILE	POINTS	NAME	CAL	OFFSET	MIN	MAX	NAME	CAL	OFFSET	MIN	MAX
AIAHC	16384	LOAD	258.000	0.020	-27.756	350.174	SI ACCEL	-50.000	0.034	-1.981	57.345
JIAHC	16384	LOAD	258.000	0.021	-11.661	351.151	SI ACCEL	-50.000	0.034	-0.760	53.195
AFLHC	16384	LOAD	258.000	0.015	-45.402	482.440	SI ACCEL	-50.000	0.037	-1.083	79.727
JFLHC	16384	LOAD	258.000	0.016	-79.603	376.432	SI ACCEL	-50.000	0.022	-1.828	79.471
AFLFC	16384	LOAD	258.000	0.020	-27.750	1147.611	SI ACCEL	-50.000	0.040	-4.095	104.792
KFLFC	16384	LOAD	258.000	0.018	-3.307	1164.496	SI ACCEL	-50.000	0.048	-236.606	502.408
JFLFC	16384	LOAD	258.000	0.020	-55.627	1173.905	SI ACCEL	-50.000	0.033	-3.008	103.926
MFLFC	16384	LOAD	258.000	-0.041	-39.871	1213.595	SI ACCEL	-50.000	0.033	-4.934	109.079
MTSFLFC	8192	LOAD	258.000	0.019	-38.855	1185.637	SI ACCEL	-50.000	0.030	-444.812	501.477
AIAFC	16384	LOAD	258.000	0.020	-40.484	759.468	SI ACCEL	-50.000	0.029	-3.681	72.735
KIAFC	8192	LOAD	258.000	0.018	-12.192	833.111	SI ACCEL	-50.000	0.025	-147.686	501.240
JIAFC	16384	LOAD	258.000	0.021	-30.671	862.503	SI ACCEL	-50.000	0.020	-3.663	76.171
MIAFC	16384	LOAD	258.000	0.021	-19.172	988.640	SI ACCEL	-50.000	0.034	-1.714	89.839
MTSIAFC	8192	LOAD	258.000	0.017	-18.211	1182.345	SI ACCEL	-50.000	0.028	-2.732	104.446
ABLFC	16384	LOAD	258.000	0.369	-146.901	2483.490	SI ACCEL	-50.000	0.027	-4.261	271.130
KBLFC	16384	LOAD	258.000	-0.030	-2.313	2586.506	SI ACCEL	-50.000	0.040	-6.814	333.274
JBLFC	16384	LOAD	258.000	0.018	-148.136	2574.218	SI ACCEL	-50.000	0.034	-1.209	261.975
MBLFC	16384	LOAD	258.000	0.021	-84.868	2573.238	SI ACCEL	-50.000	0.019	-6.869	406.705
MTSBLFC	8192	LOAD	258.000	0.021	-237.270	2573.267	SI ACCEL	-50.000	0.038	-335.507	501.895

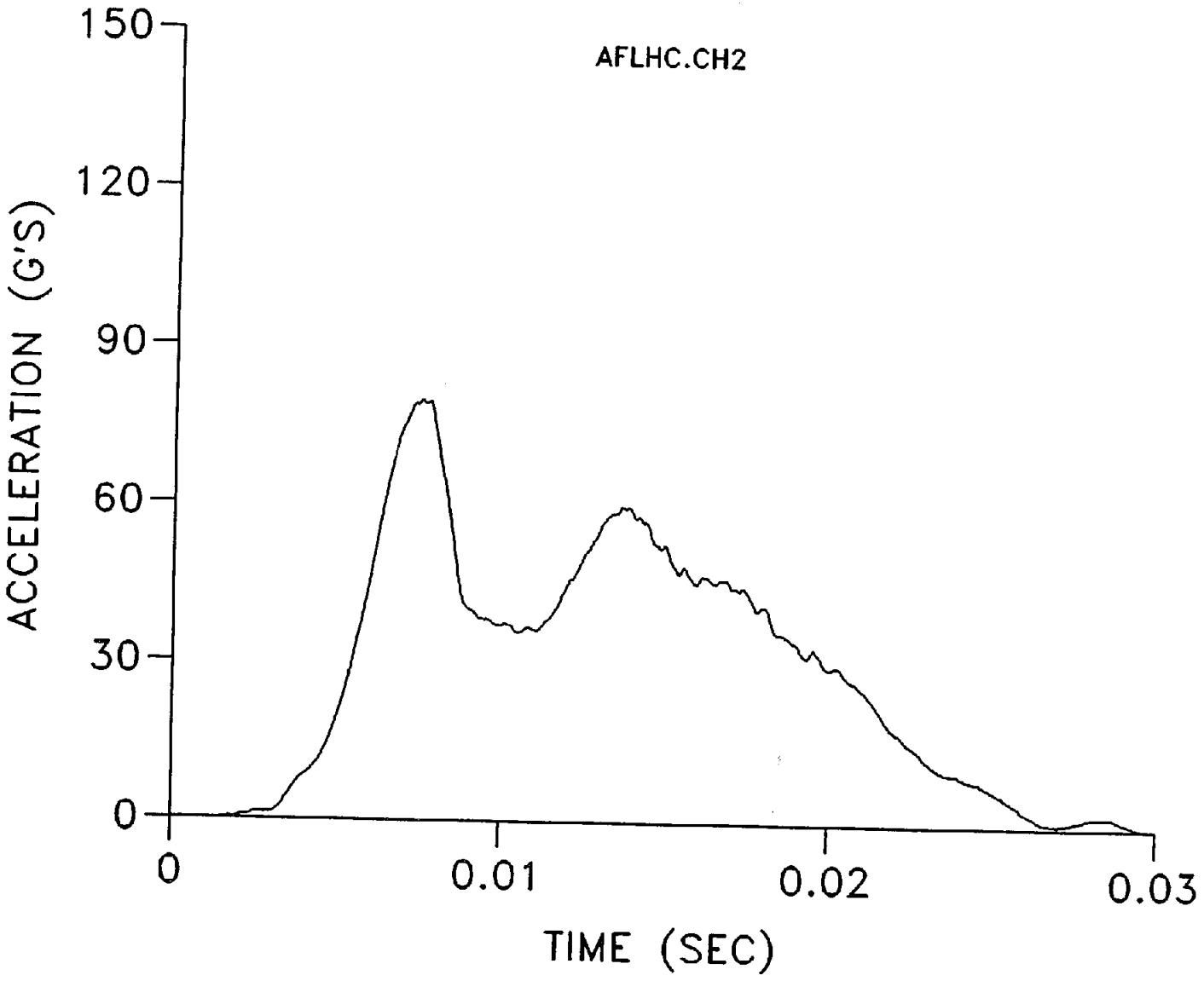

AIAHC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
0
0.01
0.02
0.03
TIME (SEC)

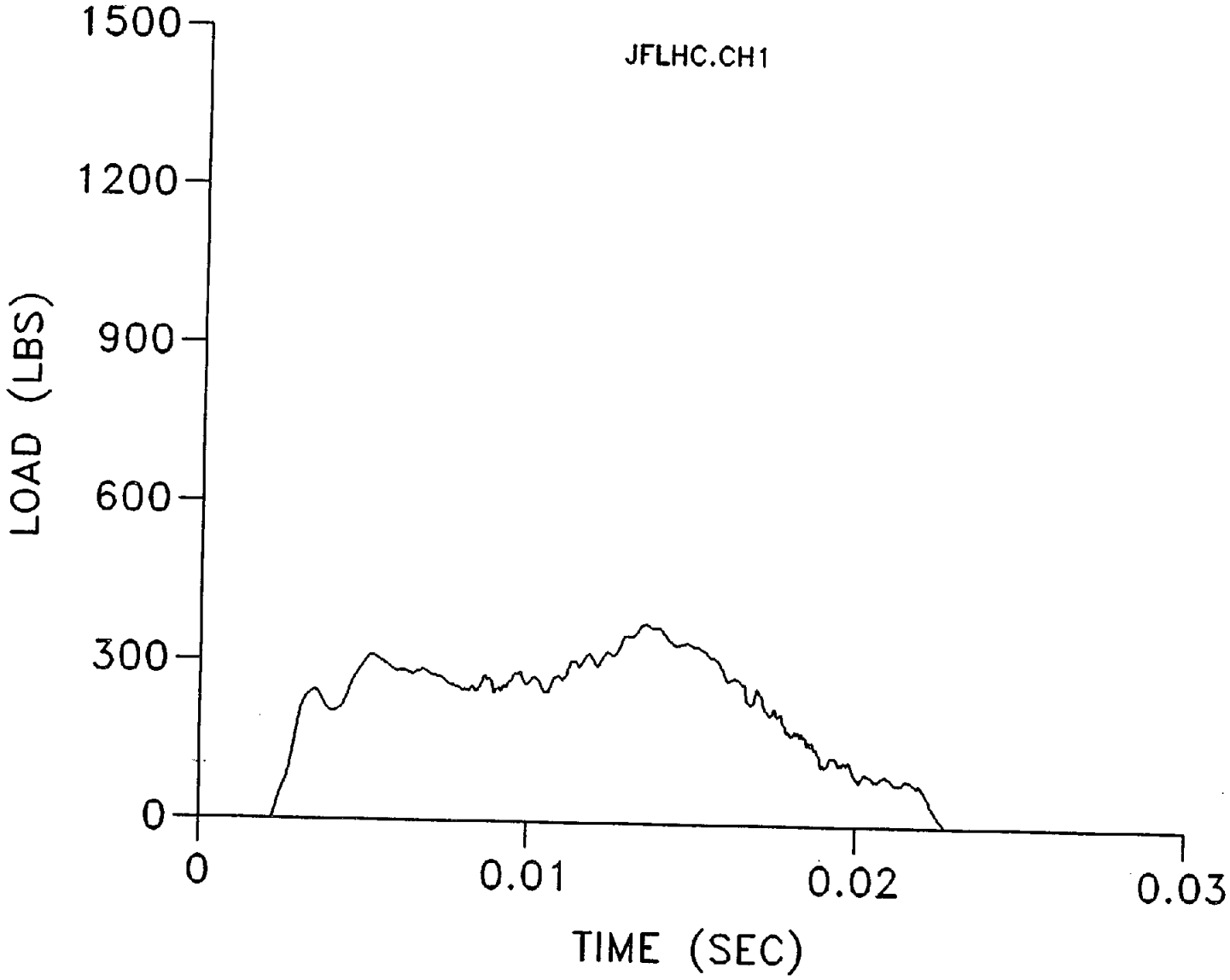

A1AHC.CH2
150
120
90
60
30
0
ACCELERATION (G'S)
0
0.01
0.02
0.03
TIME (SEC)

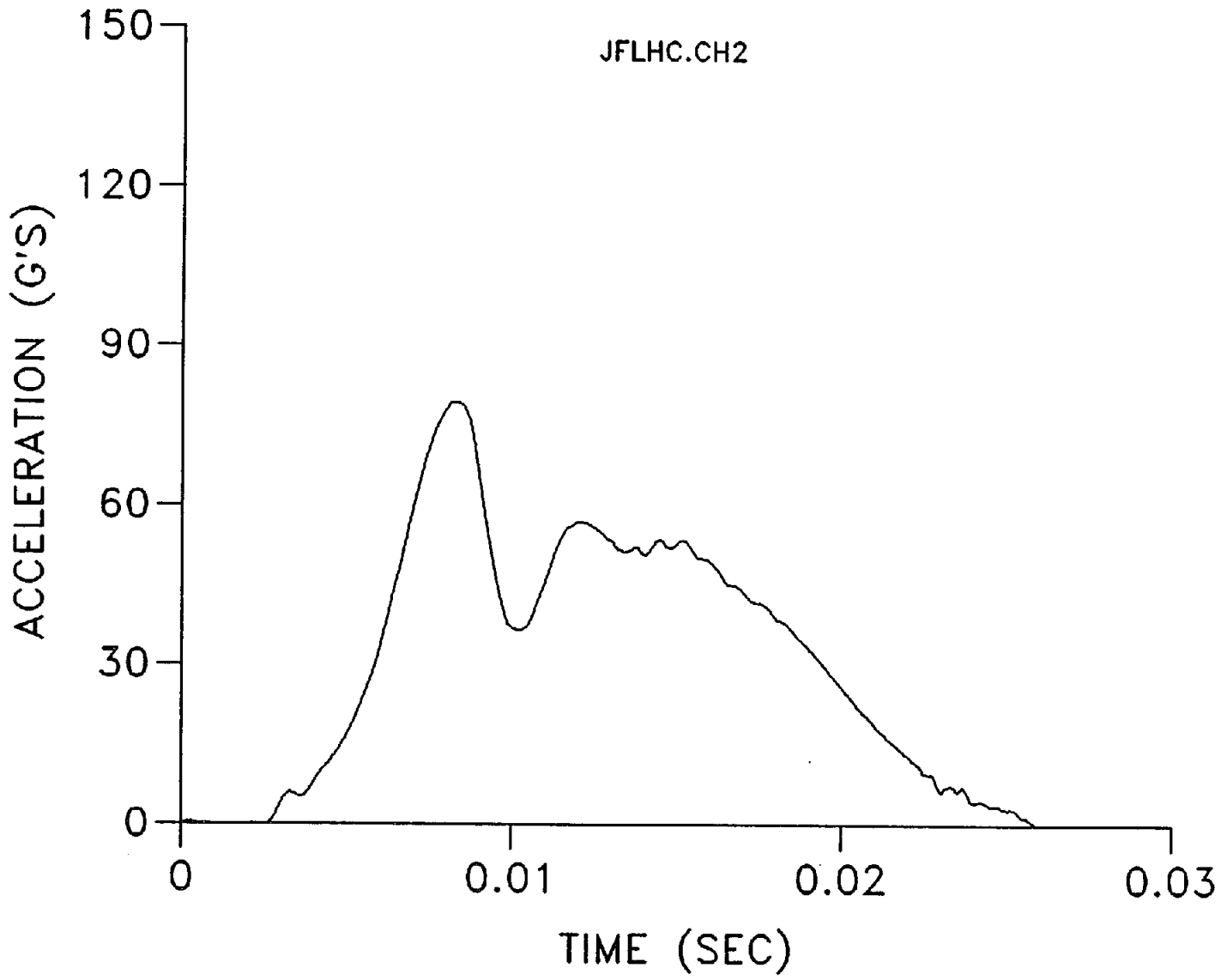

JIAHC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
0
0.01
0.02
0.03
TIME (SEC)

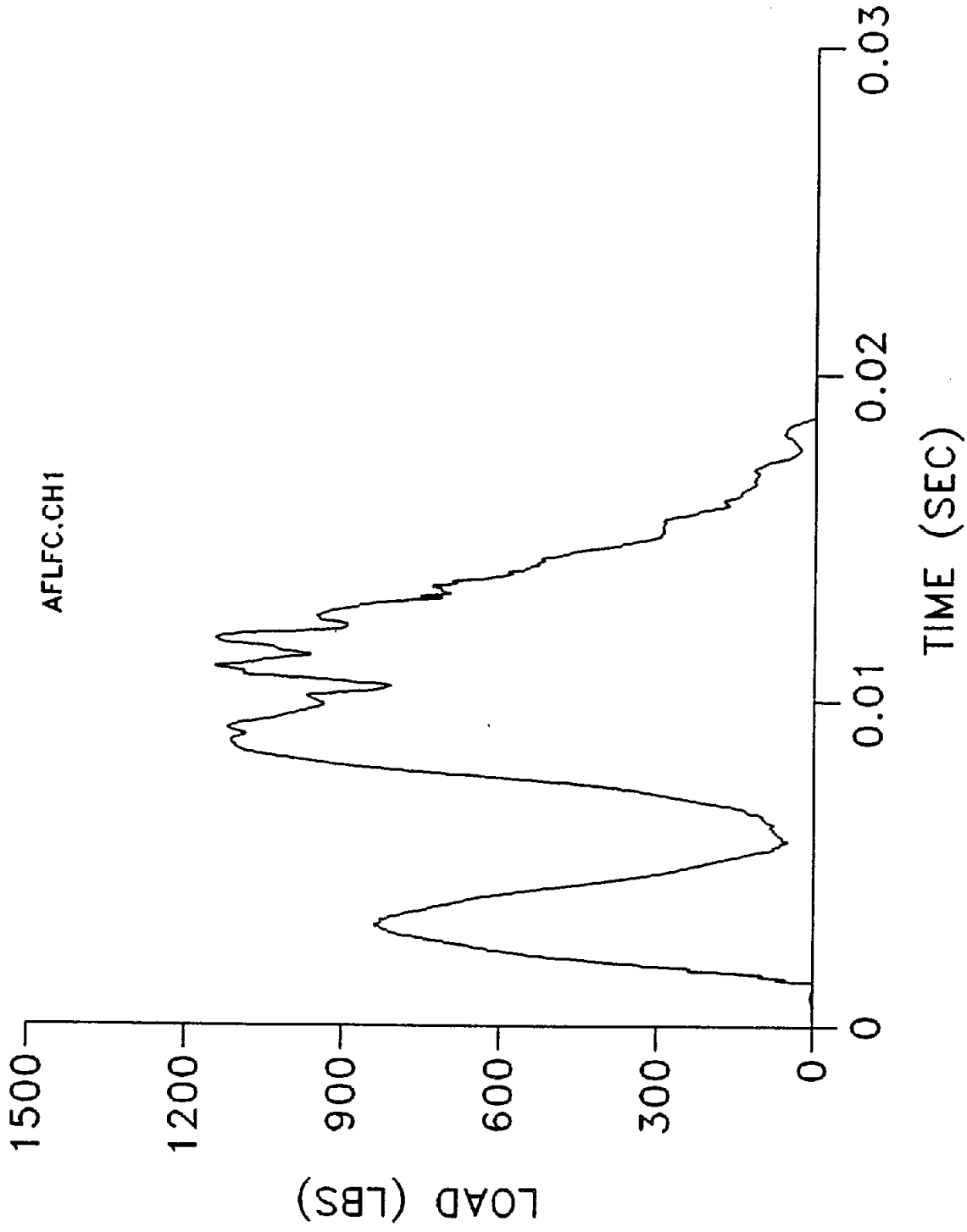

JIAHC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)

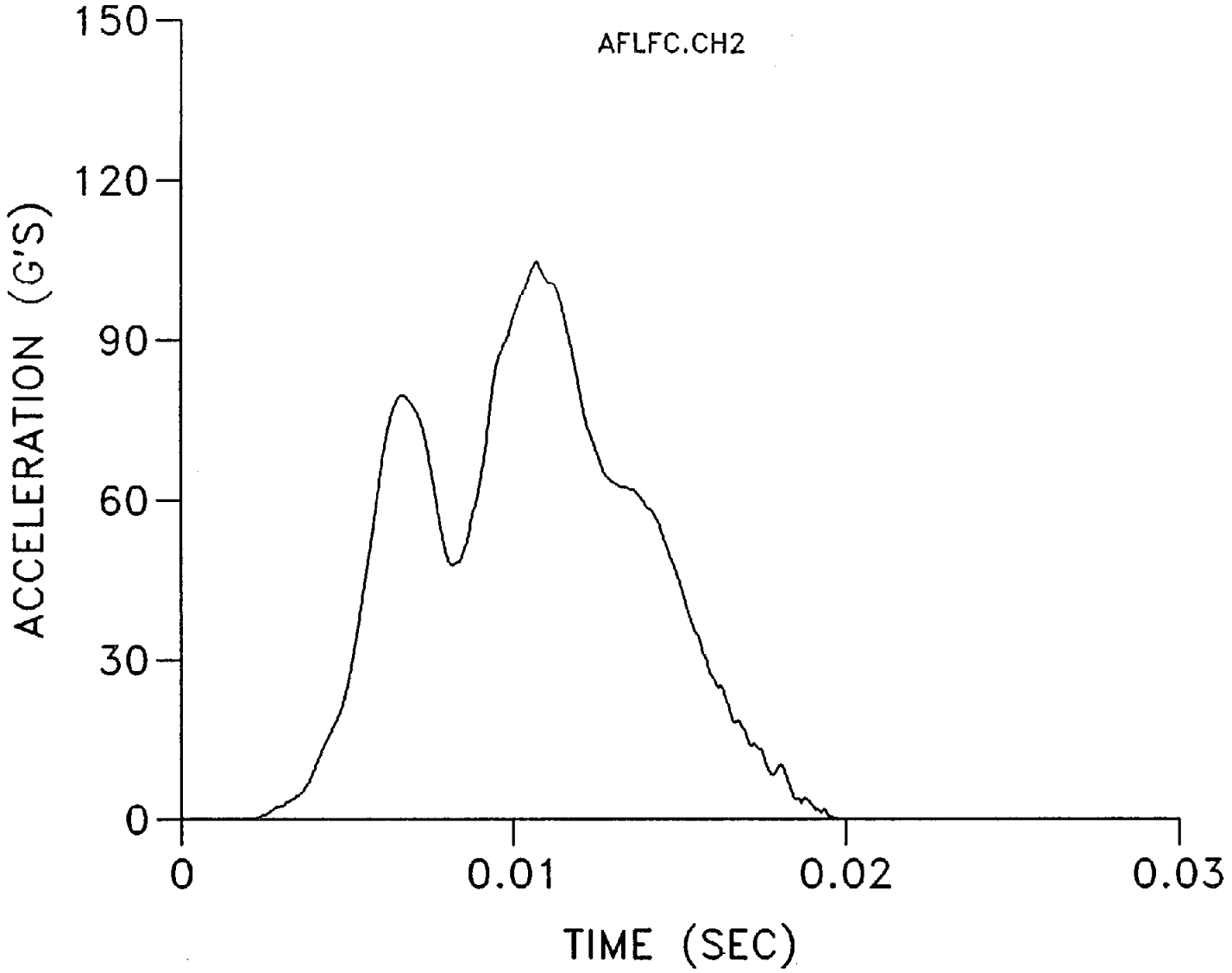

AFLHC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
0
0.01
0.02
0.03
TIME (SEC)

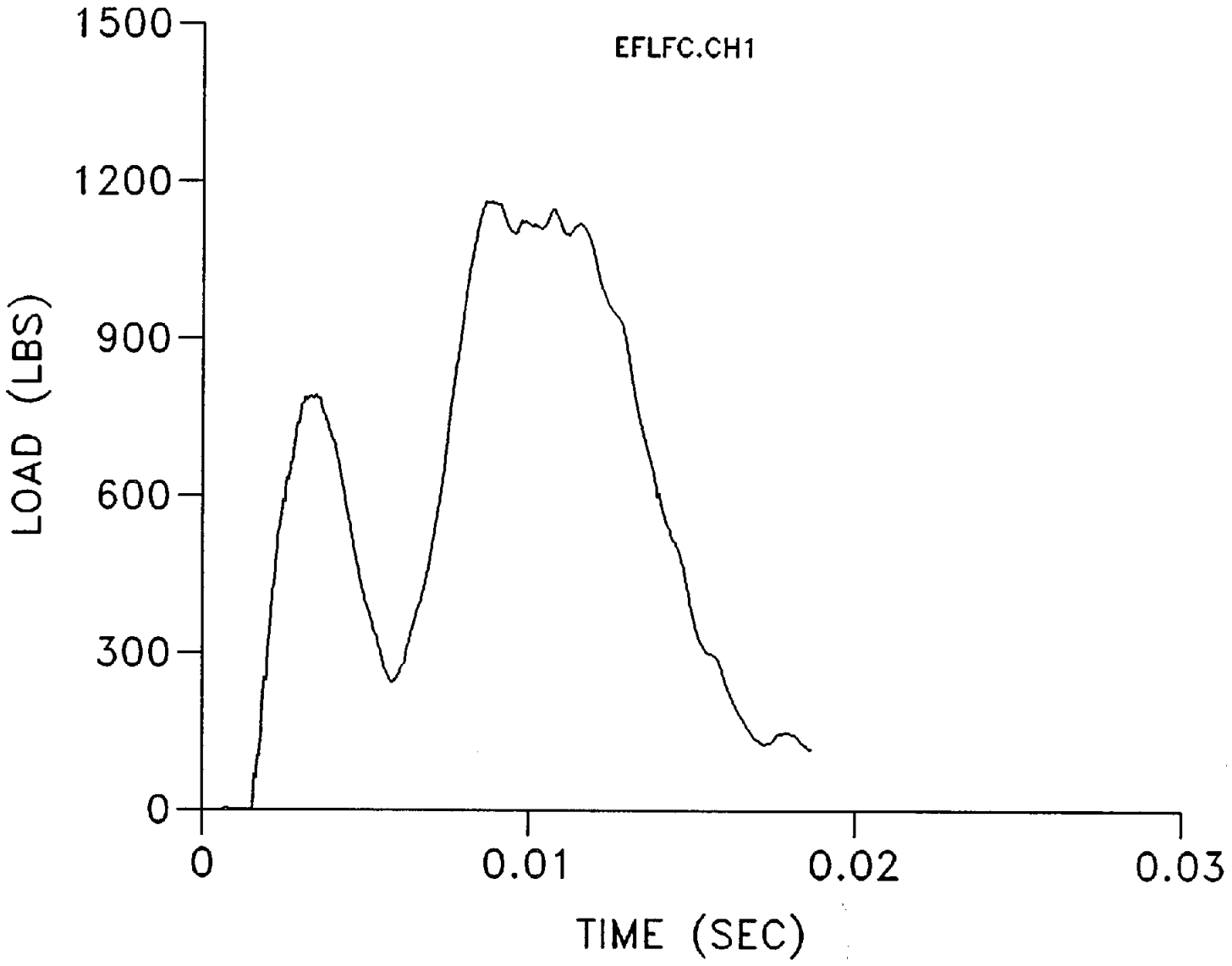

AFLHC.CH2
150
120
90
60
30
0
ACCELERATION (G'S)
0
0.01
0.02
0.03
TIME (SEC)

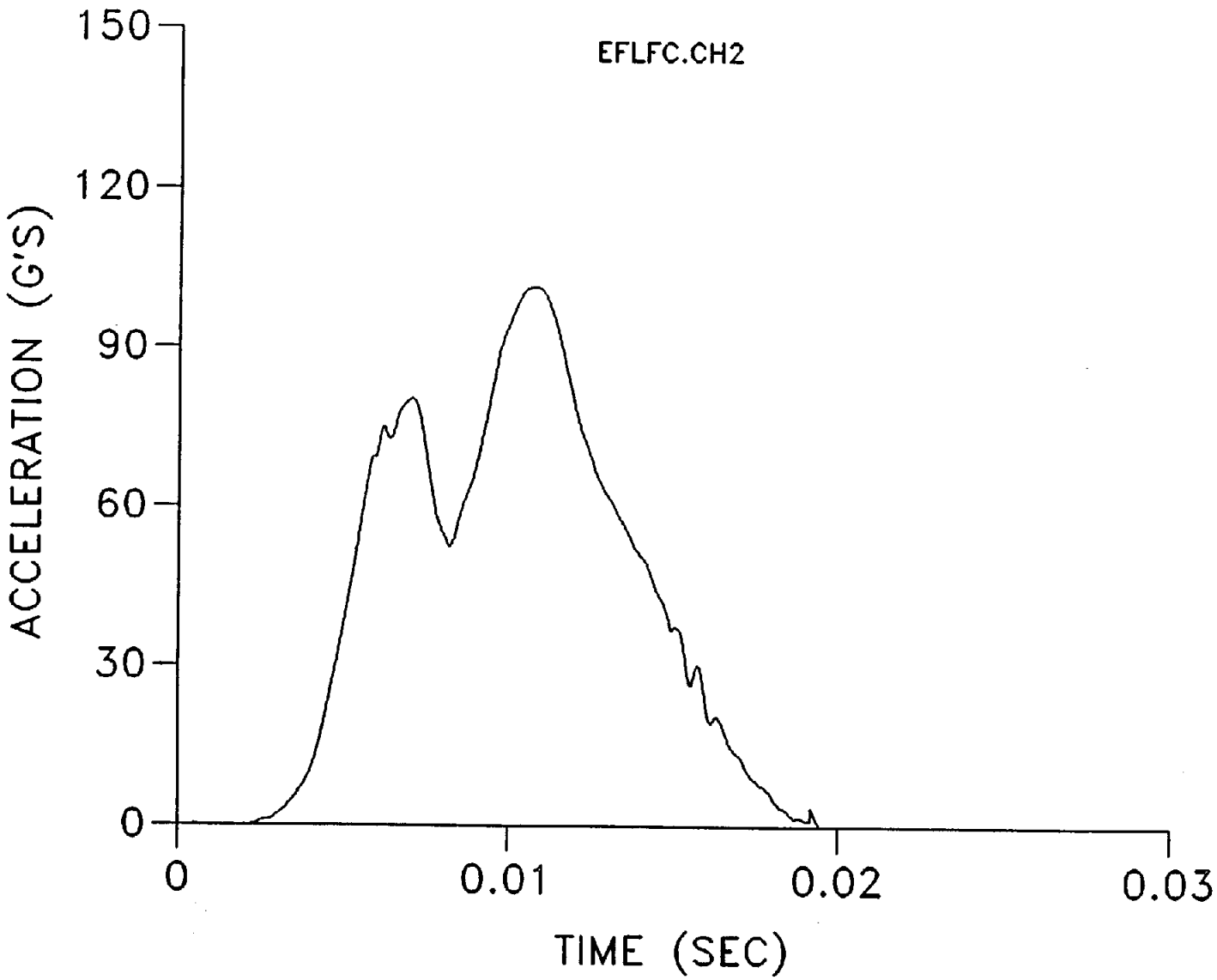

JFLHC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
0
0.01
0.02
0.03
TIME (SEC)

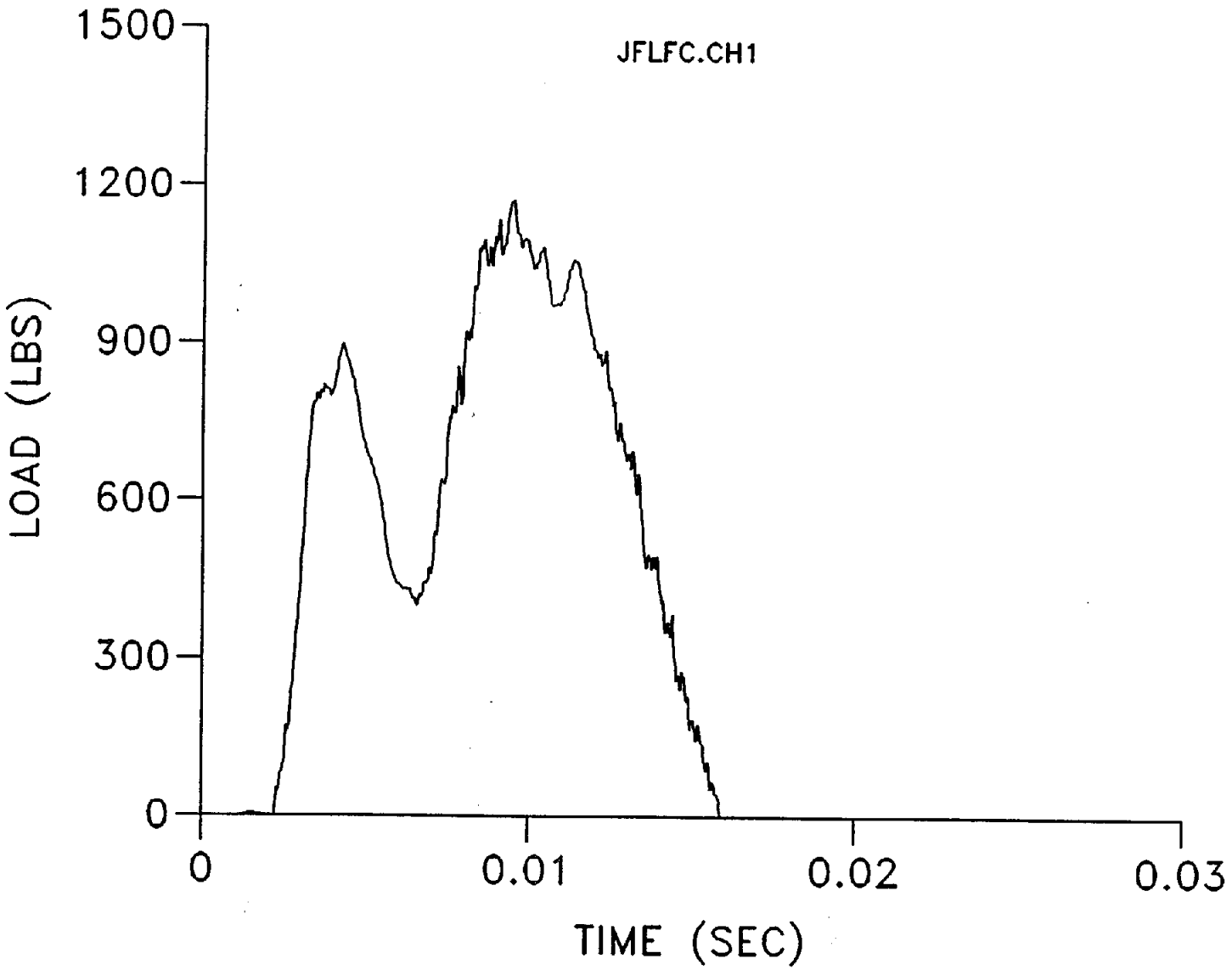

JFLHC.CH2
150
120
90
60
30
0
ACCELERATION (G'S)
0
0.01
0.02
0.03
TIME (SEC)

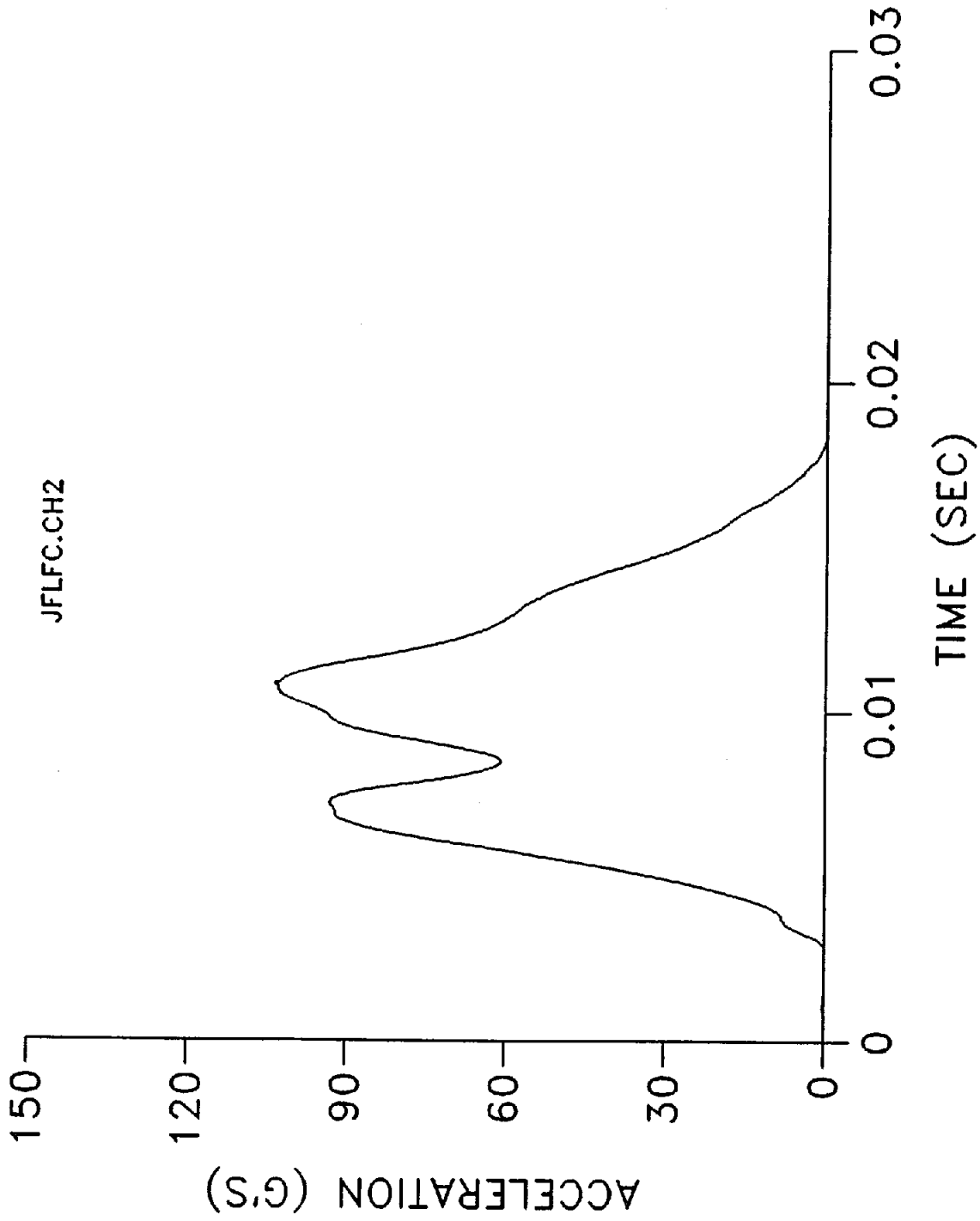

AFLFC.CH1
LOAD (LBS)
1500
1200
900
600
300
0
0
0.01
0.02
0.03
TIME (SEC)

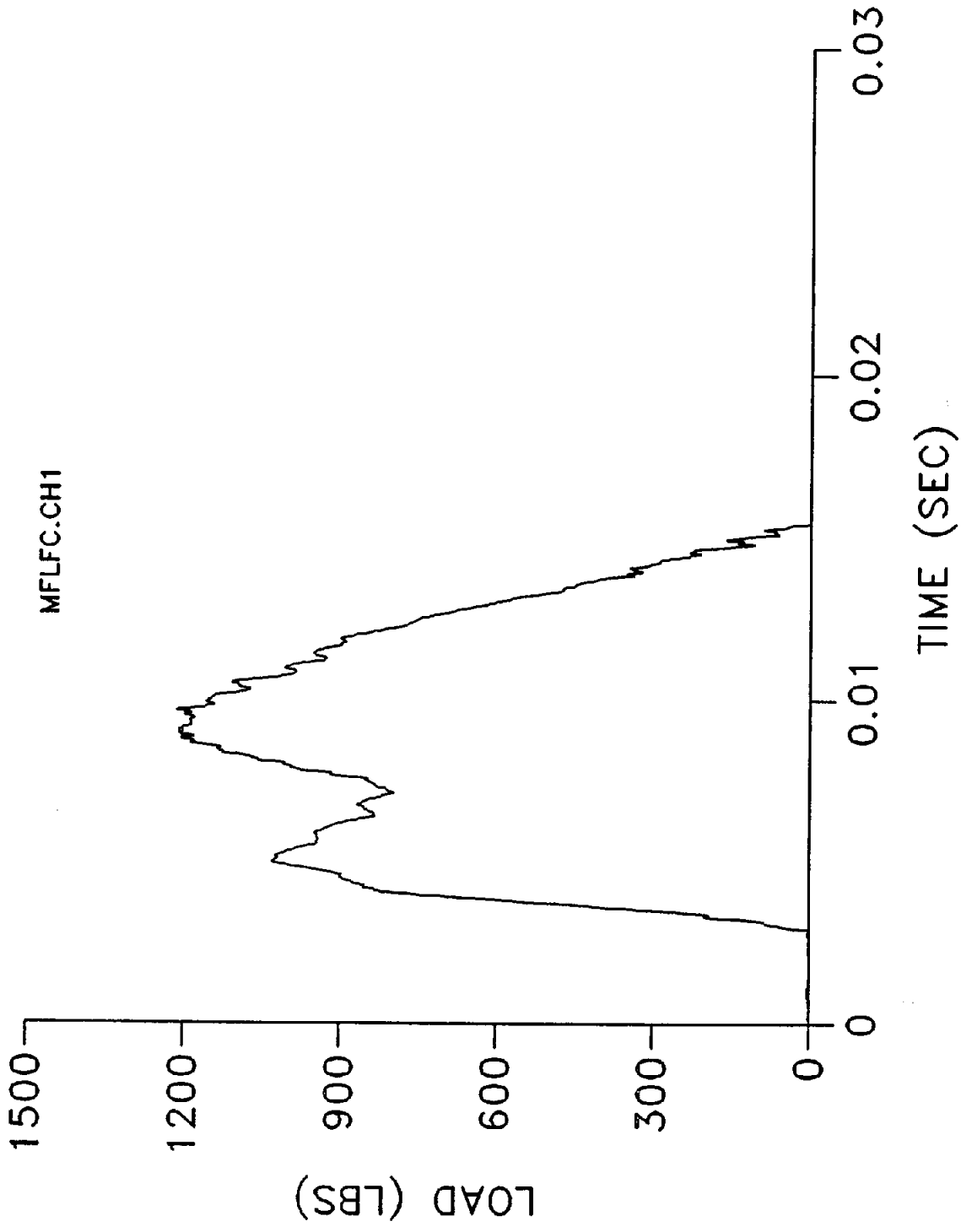

AFLFC.CH2
150
120
90
60
30
0
ACCELERATION (G'S)
0
0.01
0.02
0.03
TIME (SEC)

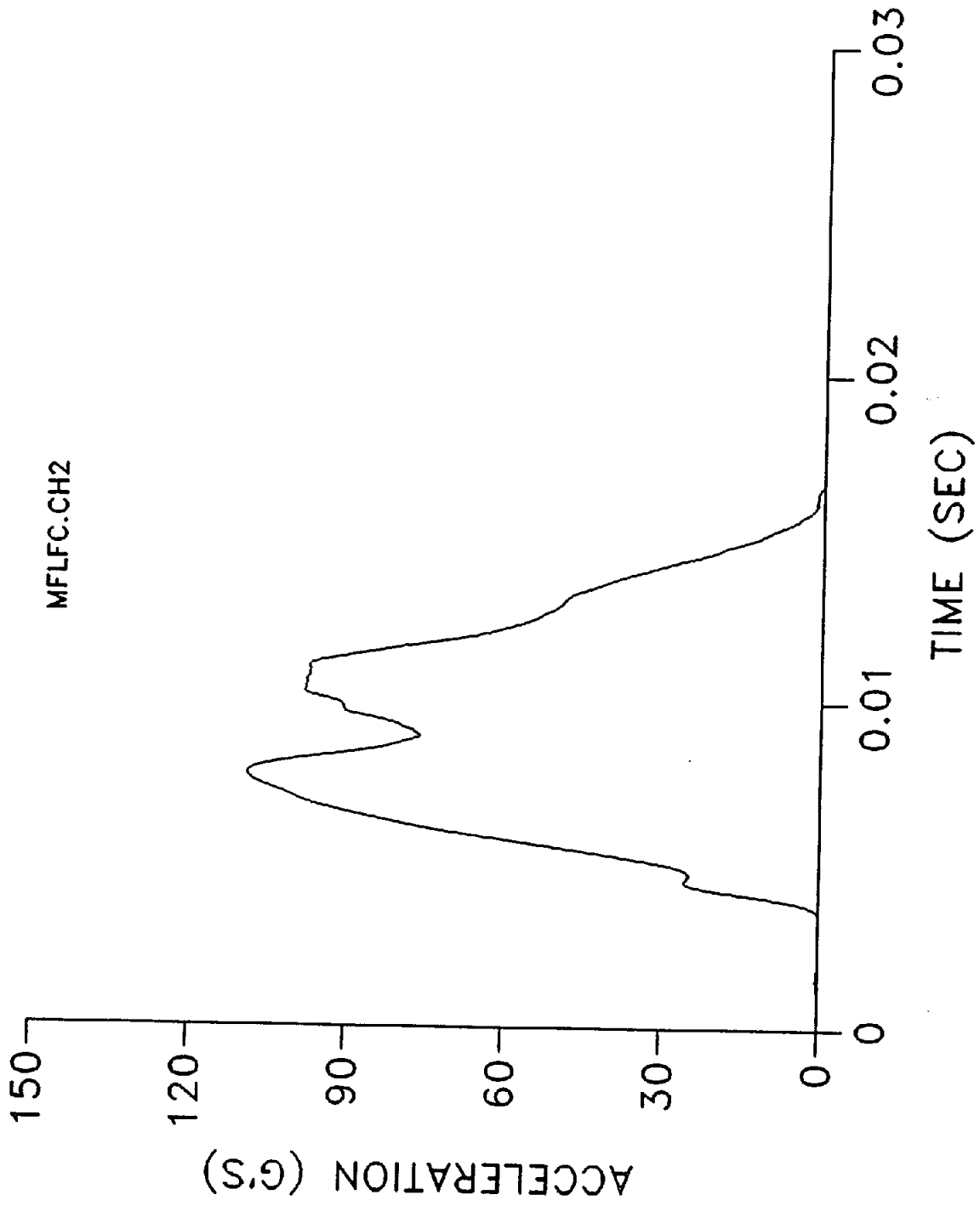

EFLFC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
0
0.01
0.02
0.03
TIME (SEC)

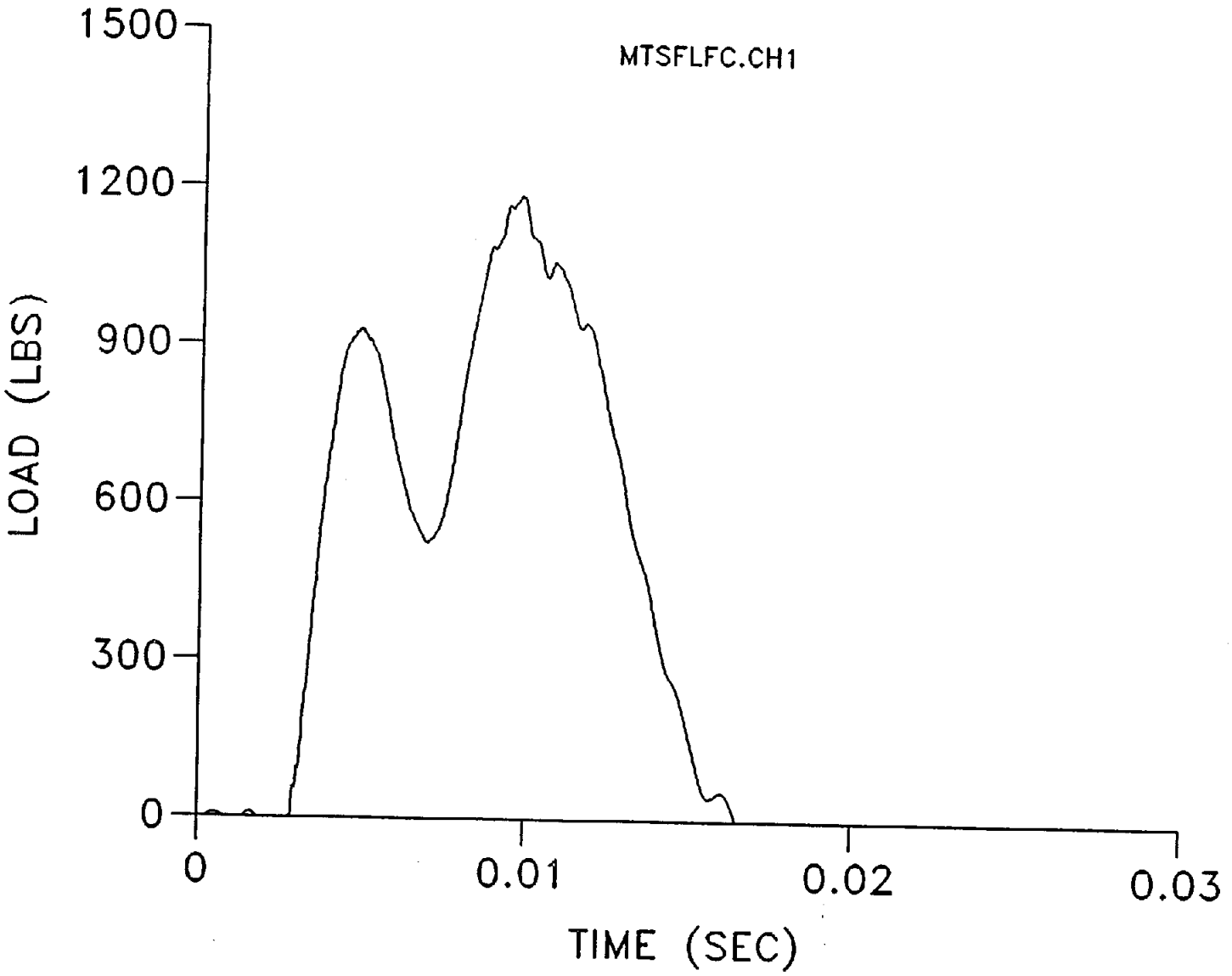

EFLFC.CH2
150
120
90
60
30
0
ACCELERATION (G'S)
0
0.01
0.02
0.03
TIME (SEC)

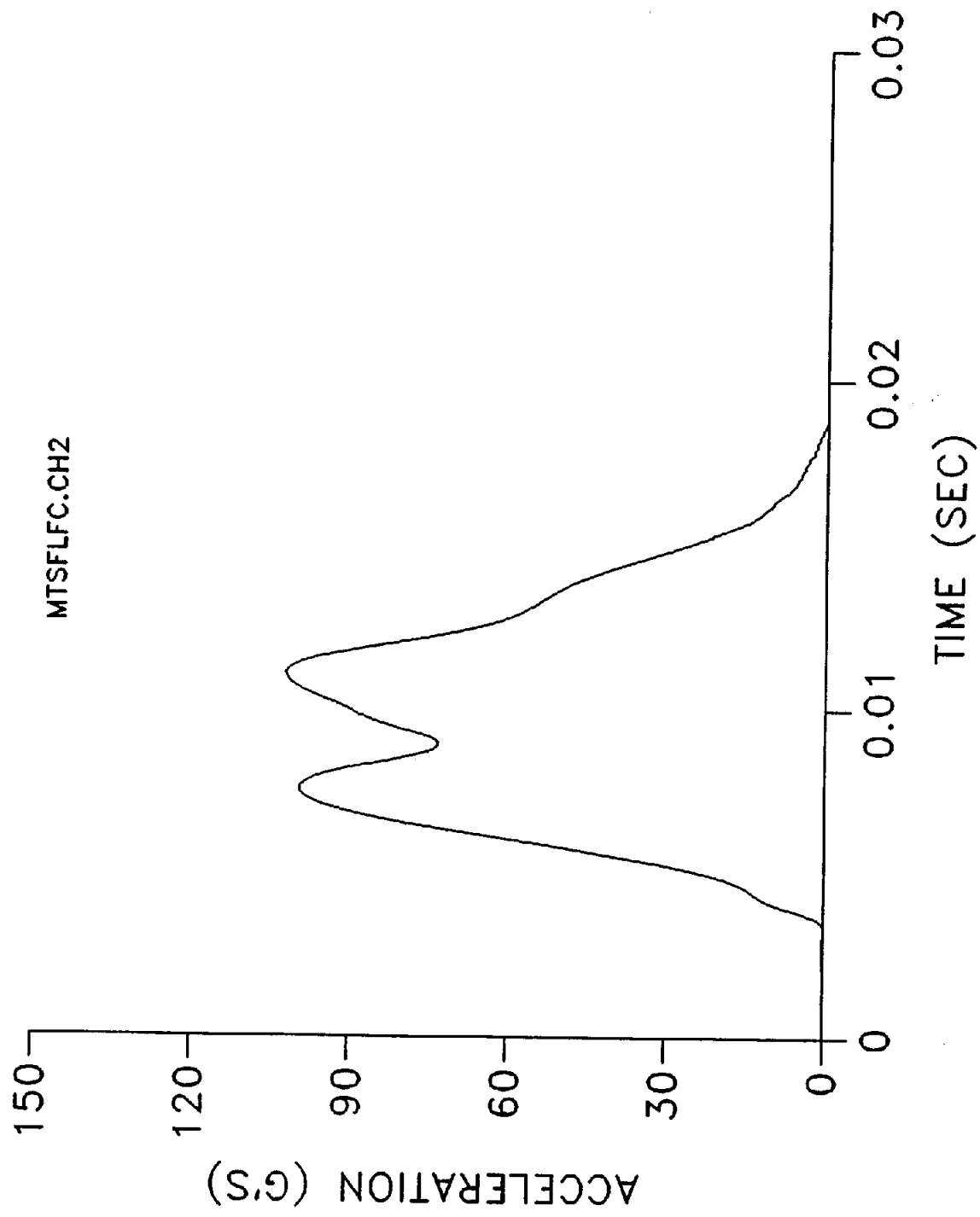

JFLFC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
0
0.01
0.02
0.03
TIME (SEC)

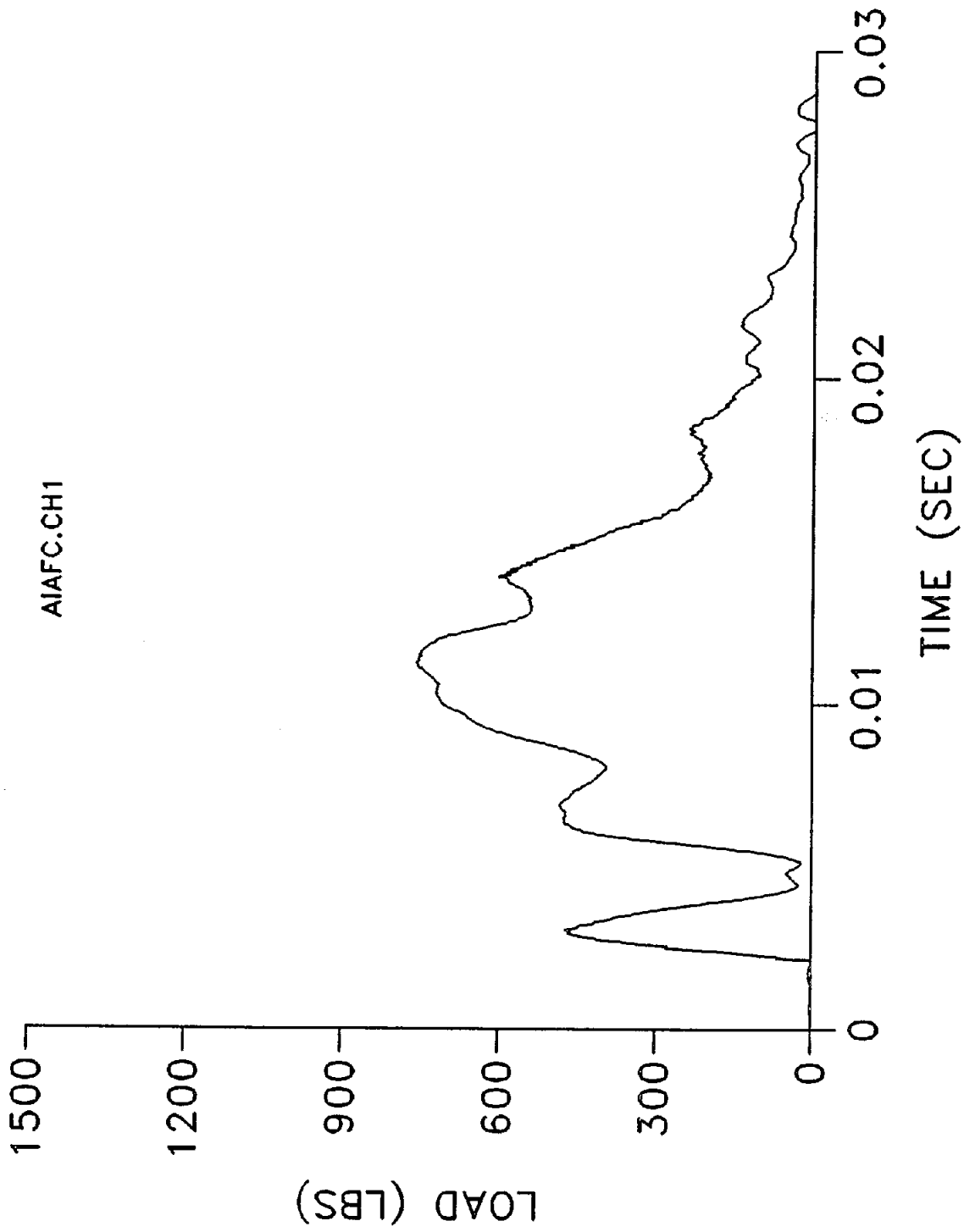

JFLFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)

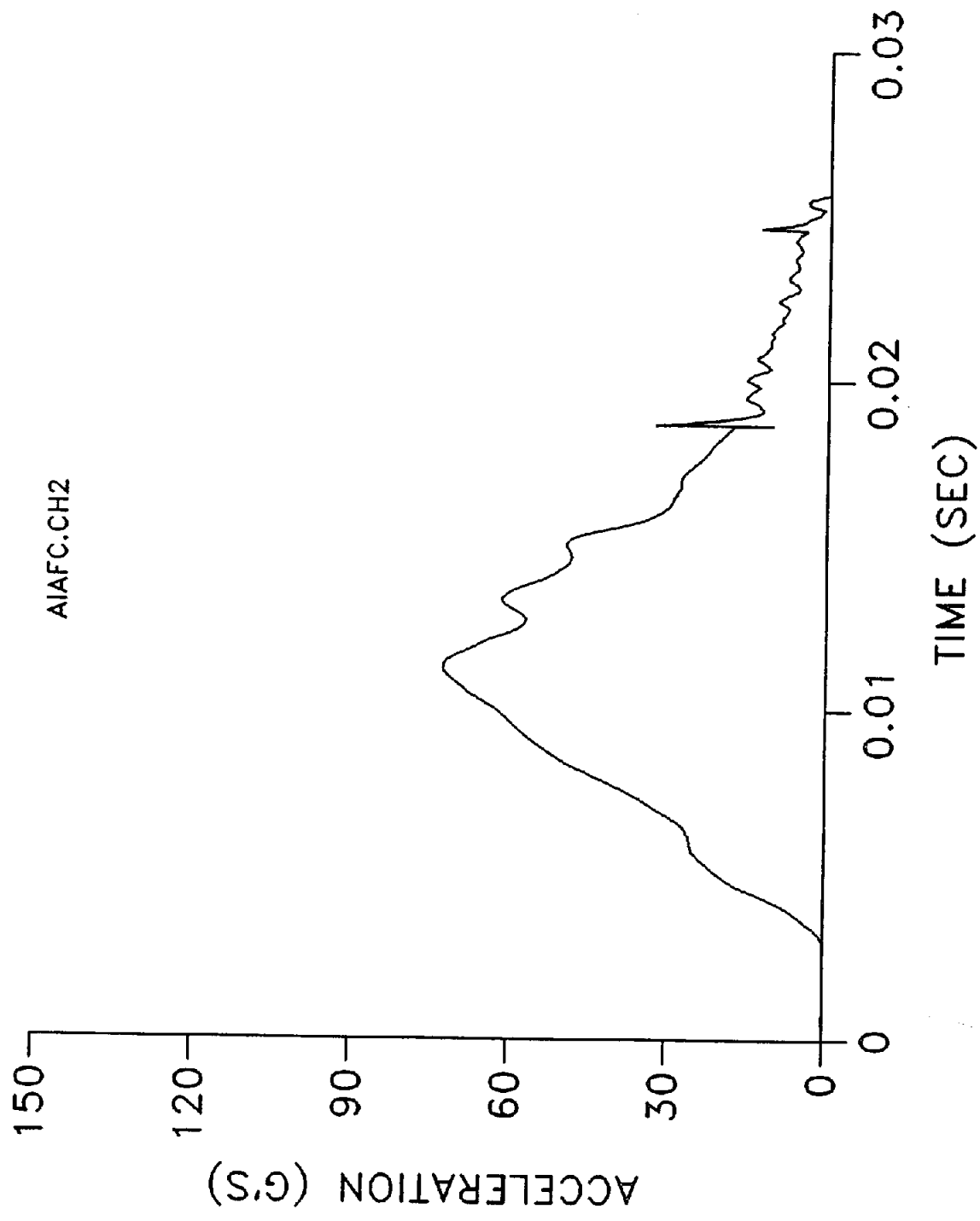

MFLFC.CH1
LOAD (LBS)
1500
1200
900
600
300
0
0
0.01
0.02
0.03
TIME (SEC)

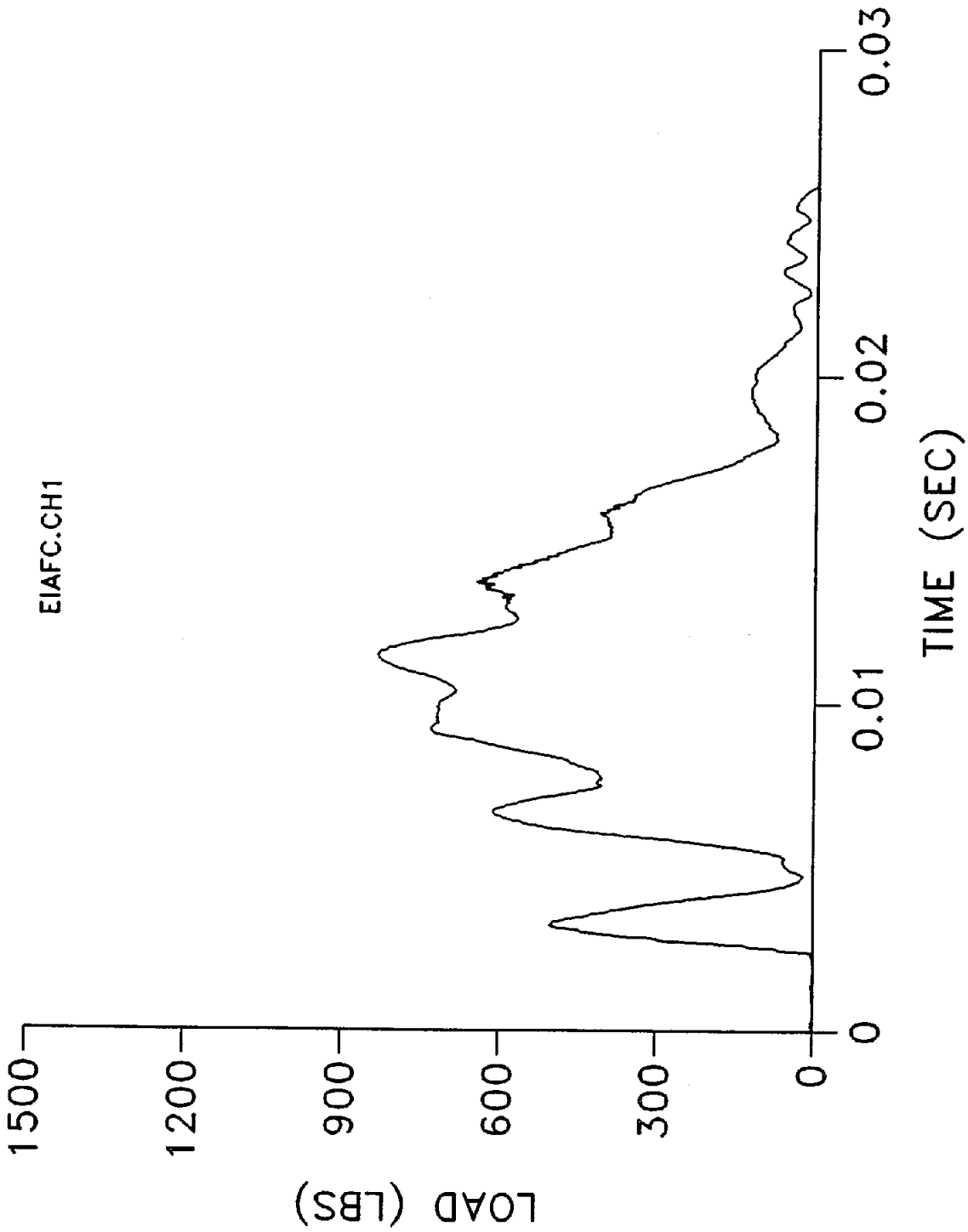

MFLFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)

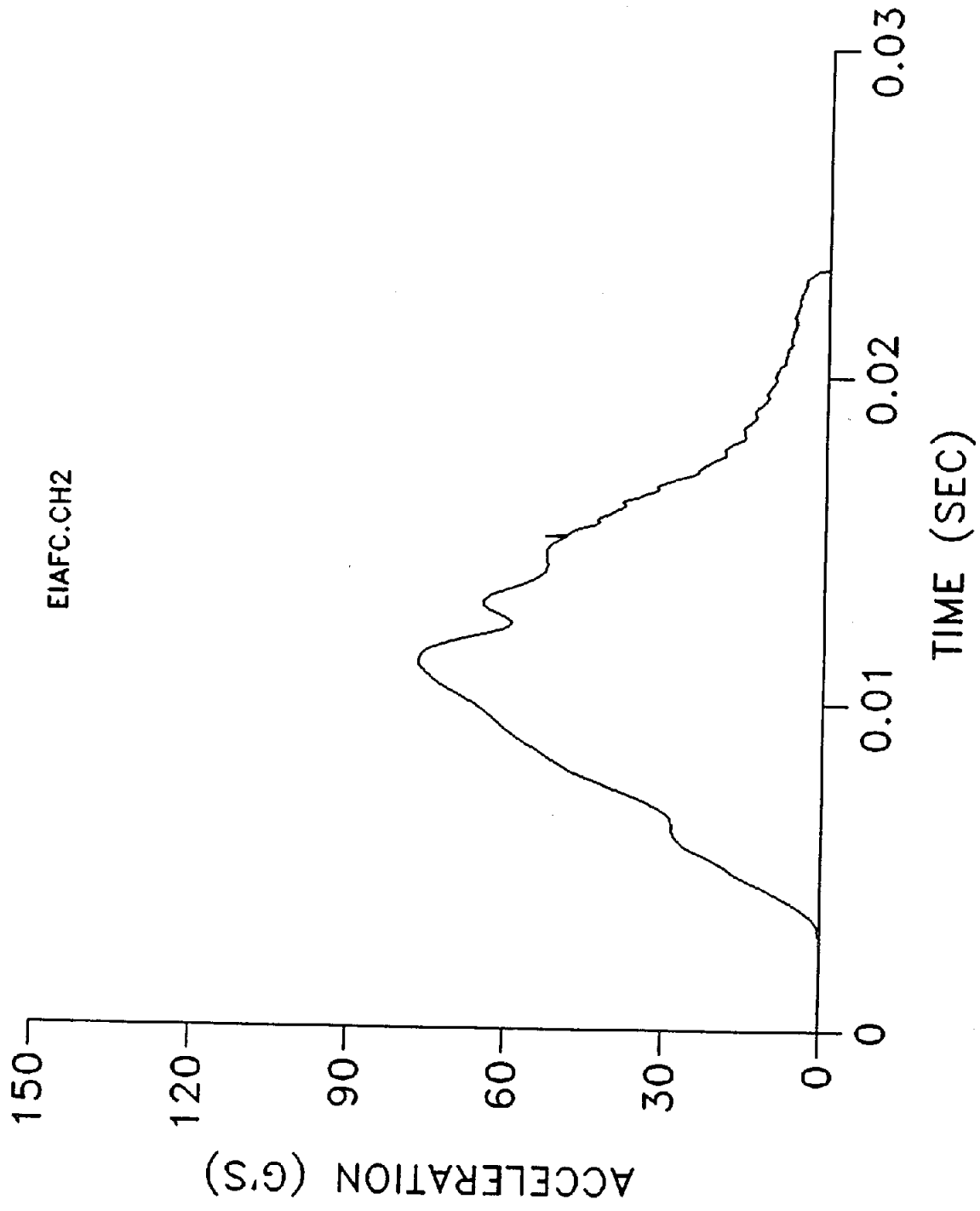

MTSFLFC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
0
0.01
0.02
0.03
TIME (SEC)

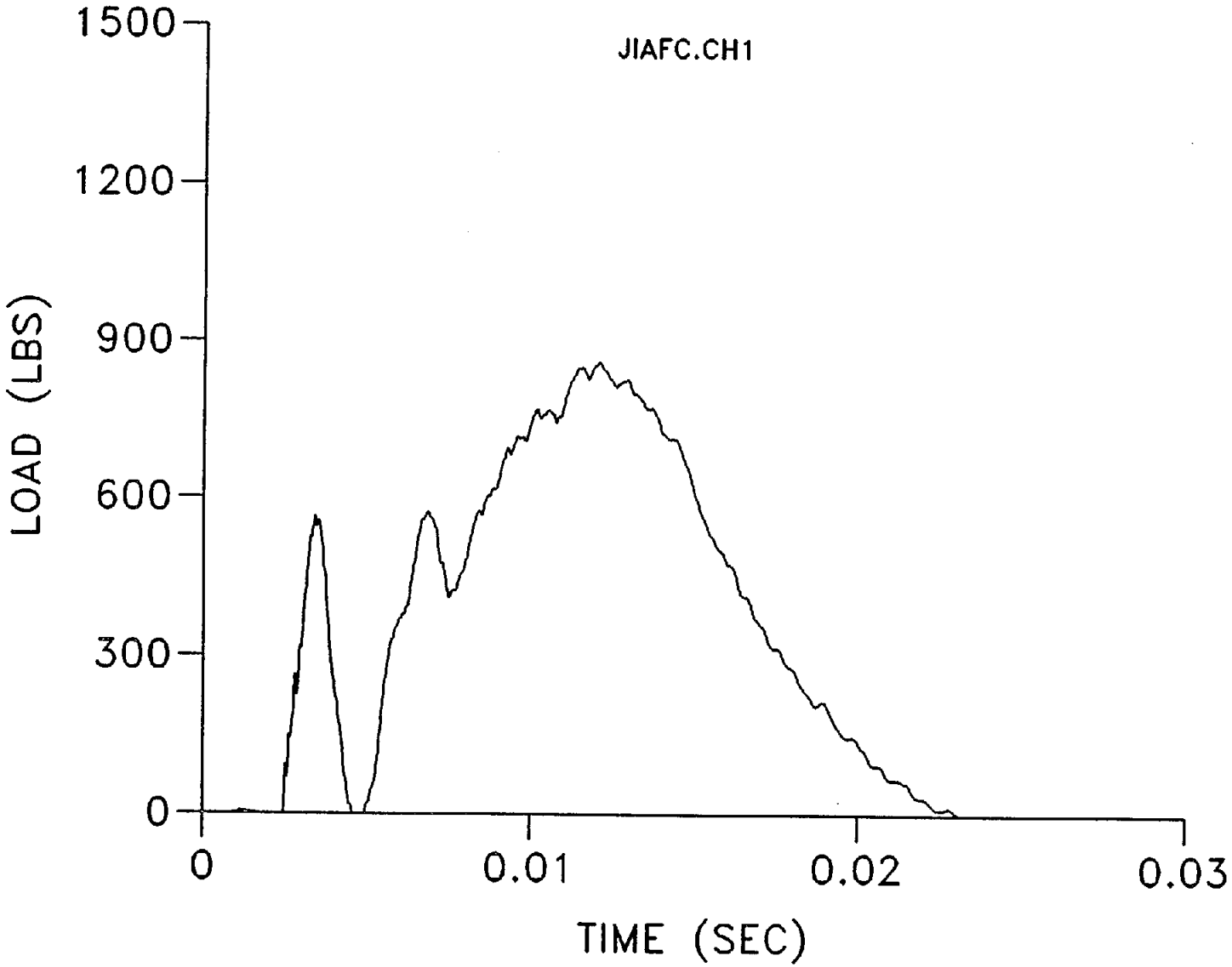

MTSFLFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)

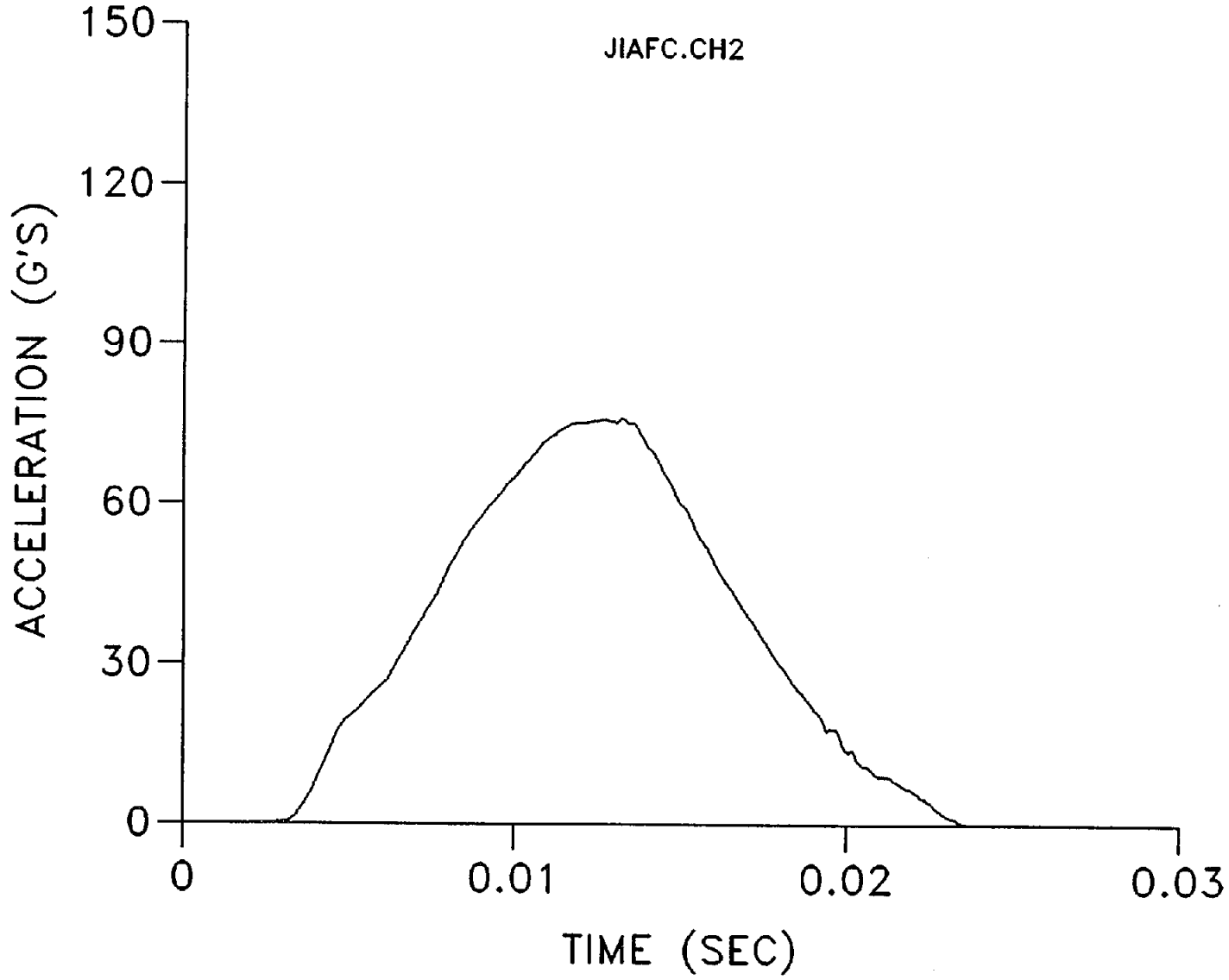

AIAFC.CH1
LOAD (LBS)
1500
1200
900
600
300
0
0
0.01
0.02
0.03
TIME (SEC)

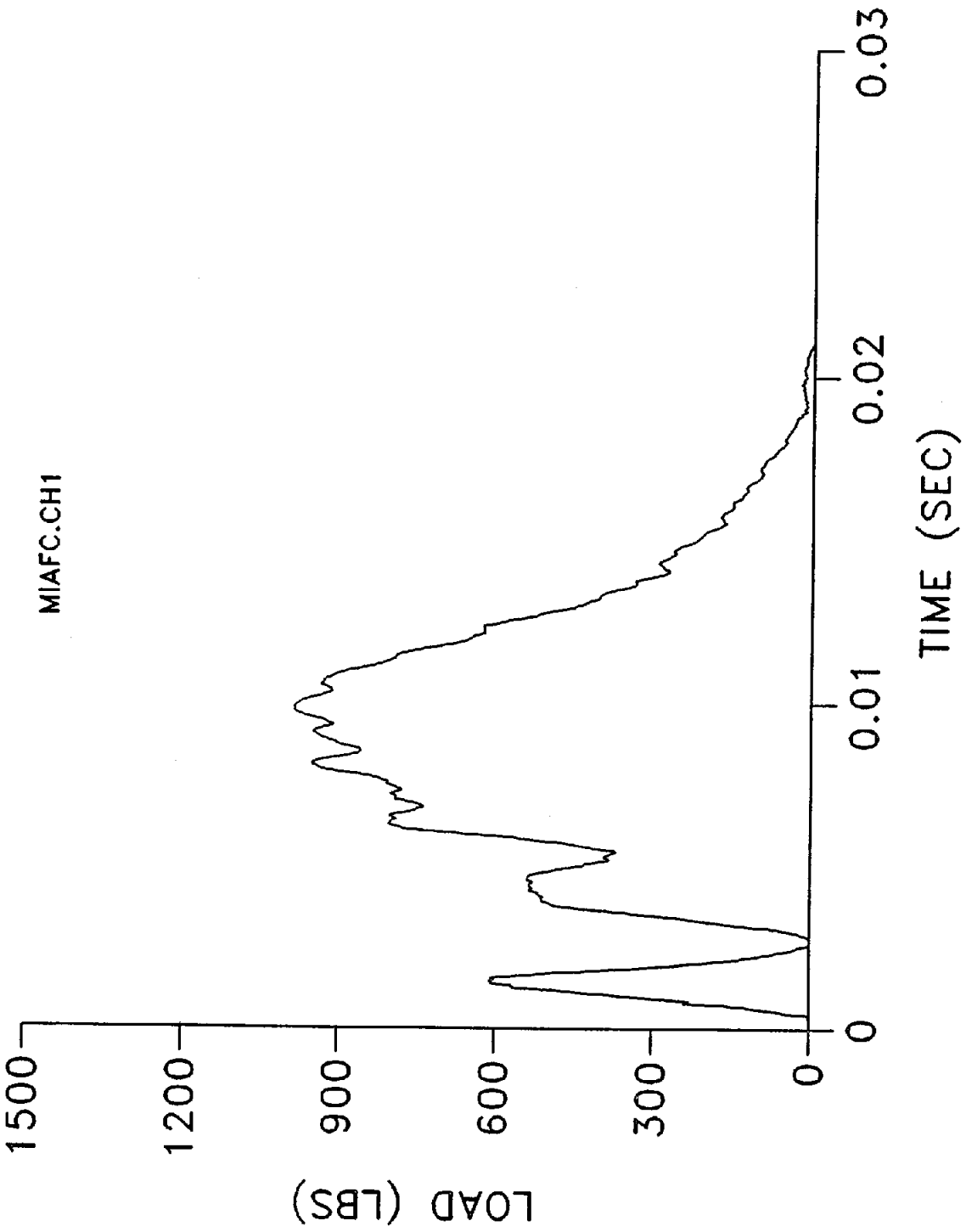

AIAFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)

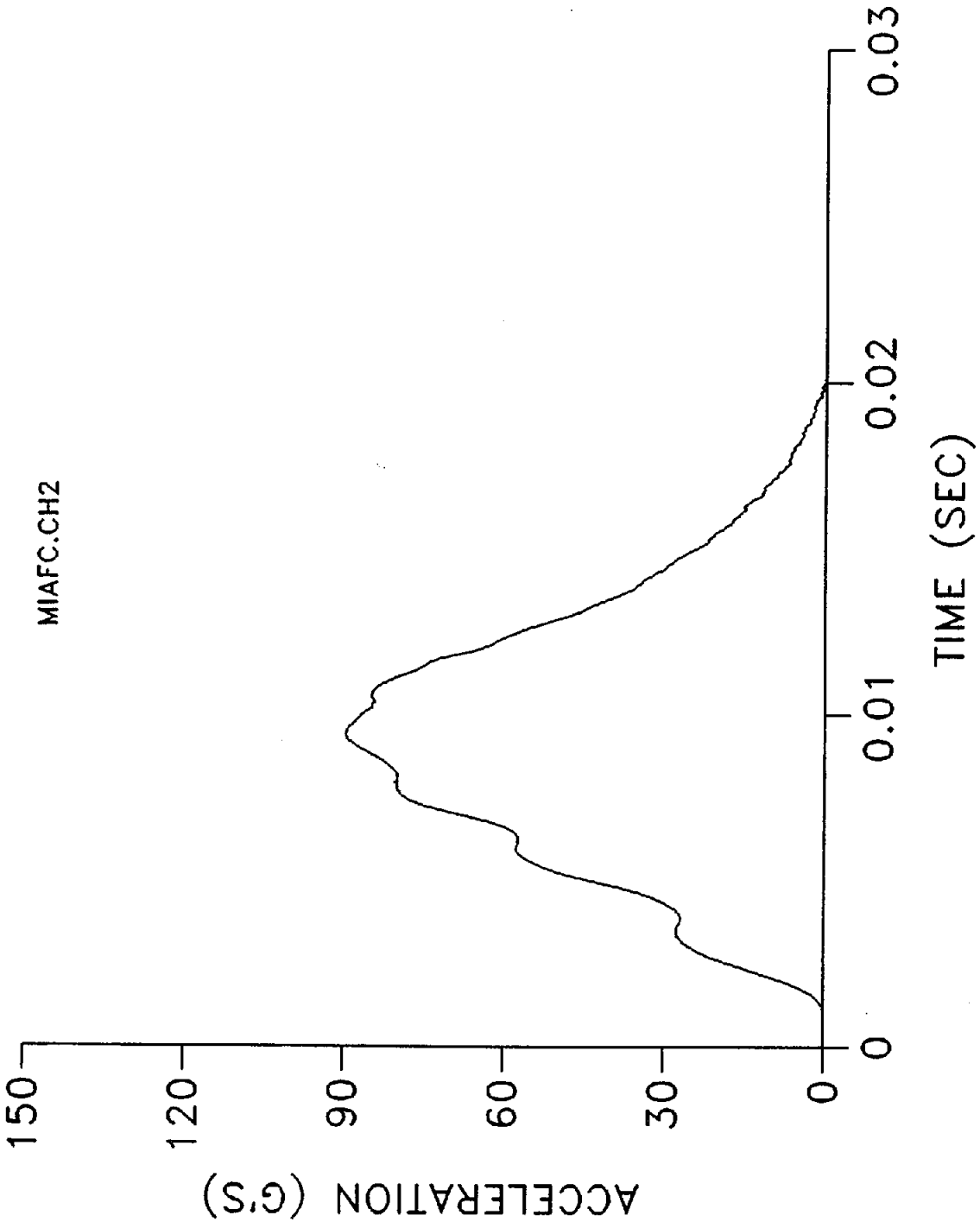

EIAFC.CH1
LOAD (LBS)
1500
1200
900
600
300
0
0
0.01
0.02
0.03
TIME (SEC)

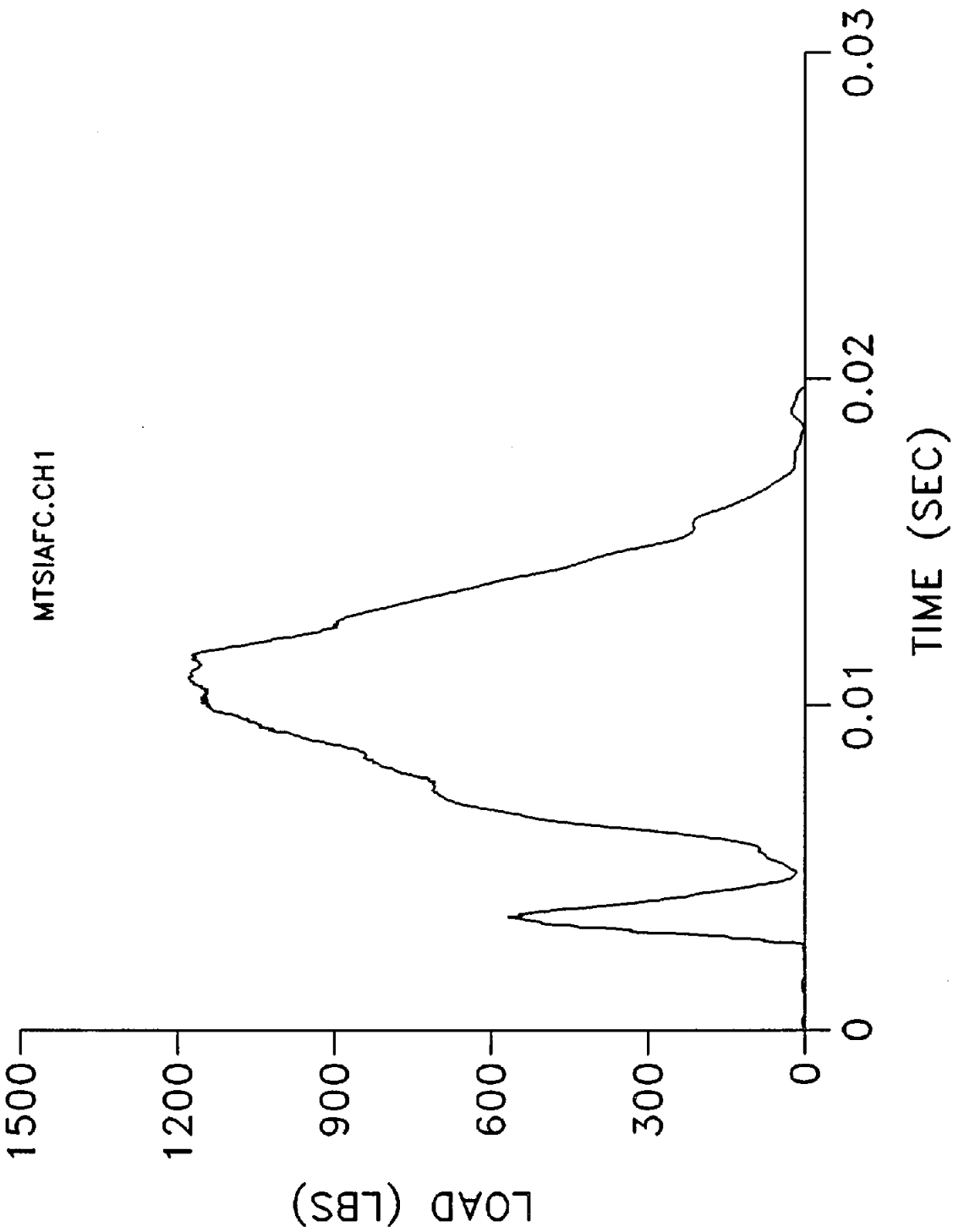

EIAFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)

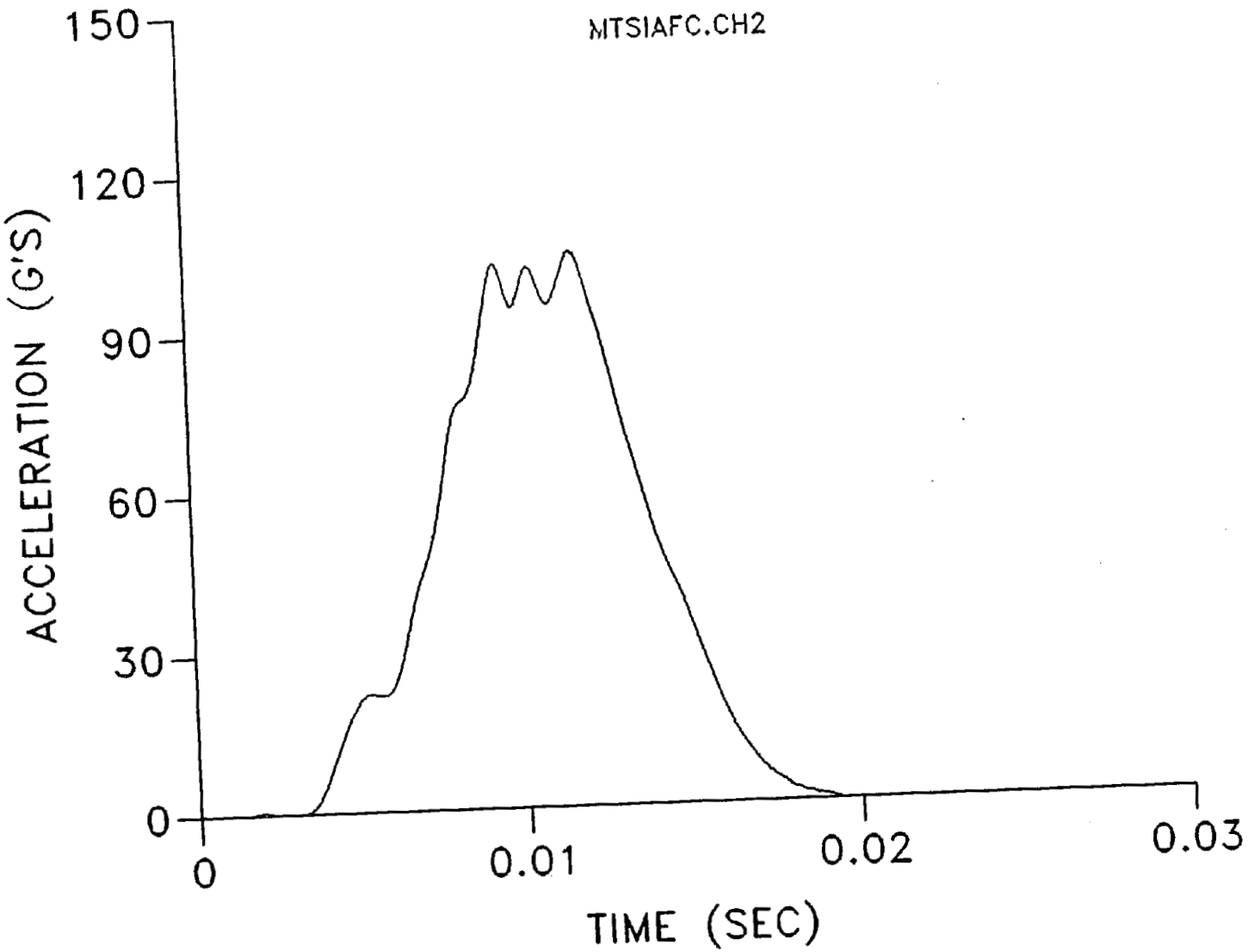

JIAFC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
0
0.01
0.02
0.03
TIME (SEC)

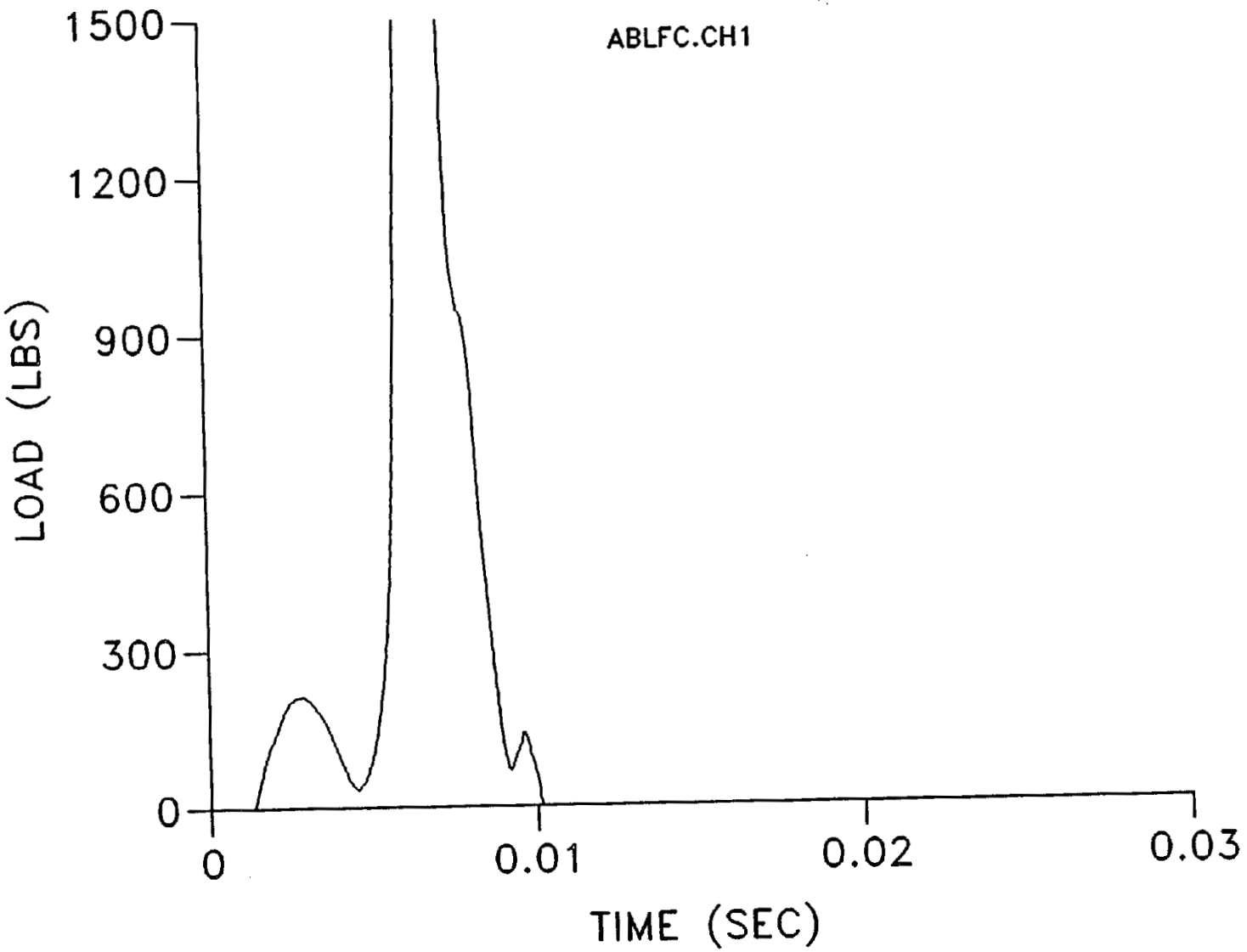

JIAFC.CH2
150
120
90
60
30
0
ACCELERATION (G'S)
0
0.01
0.02
0.03
TIME (SEC)

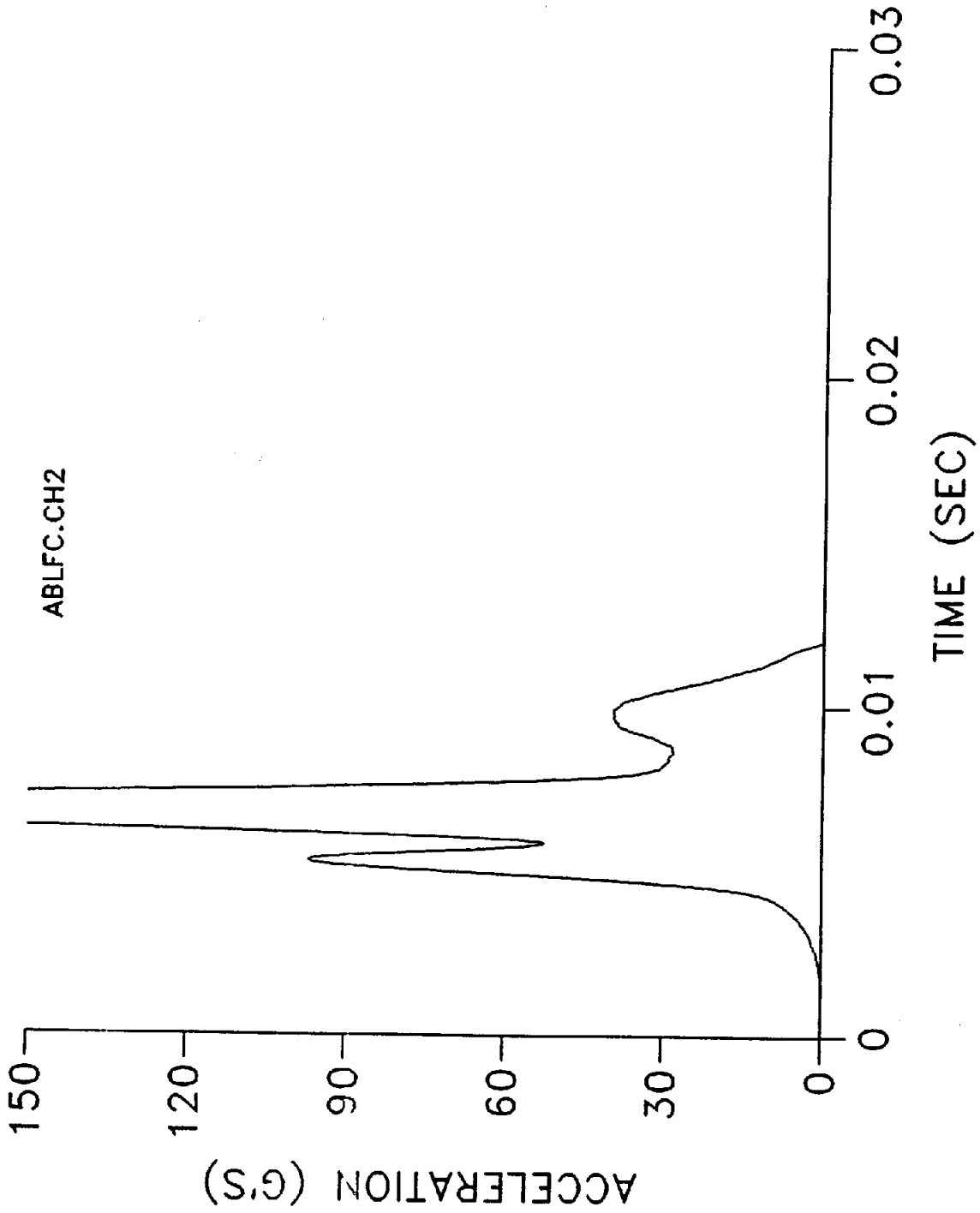

MIAFC.CH1
LOAD (LBS)
1500
1200
900
600
300
0
0
0.01
0.02
0.03
TIME (SEC)

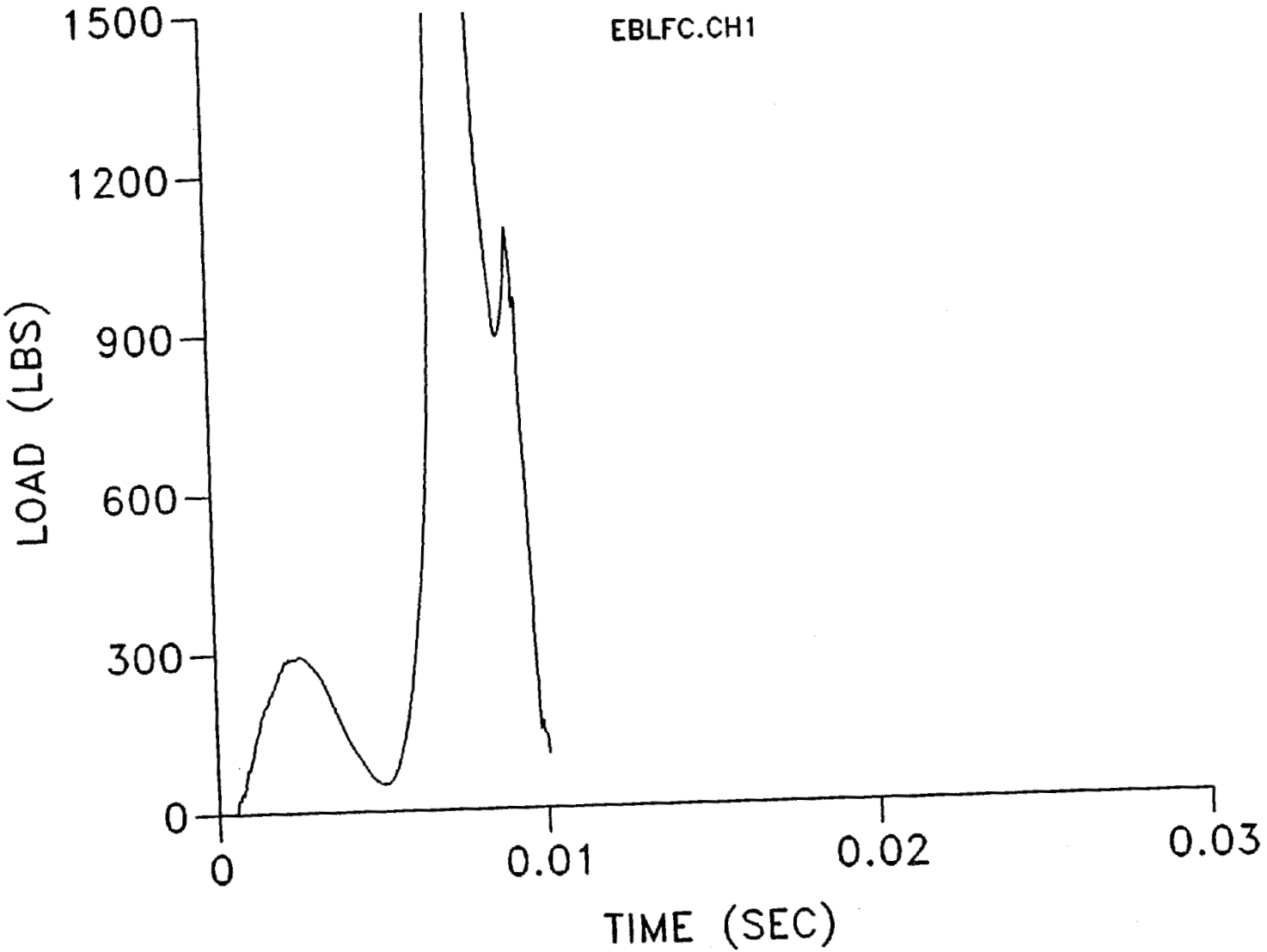

MIAFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)

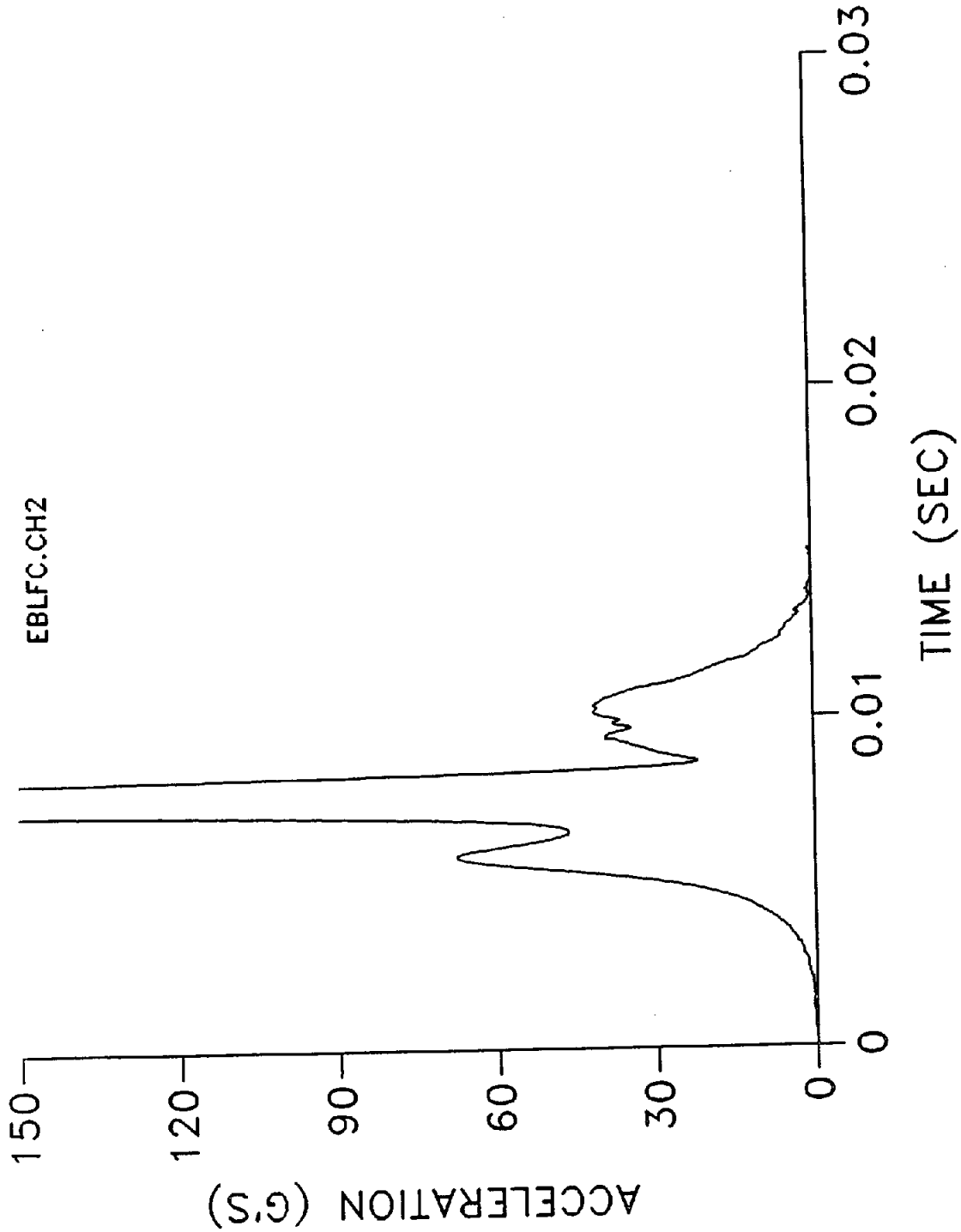

MTSIAFC.CH1
LOAD (LBS)
1500
1200
900
600
300
0
0
0.01
0.02
0.03
TIME (SEC)

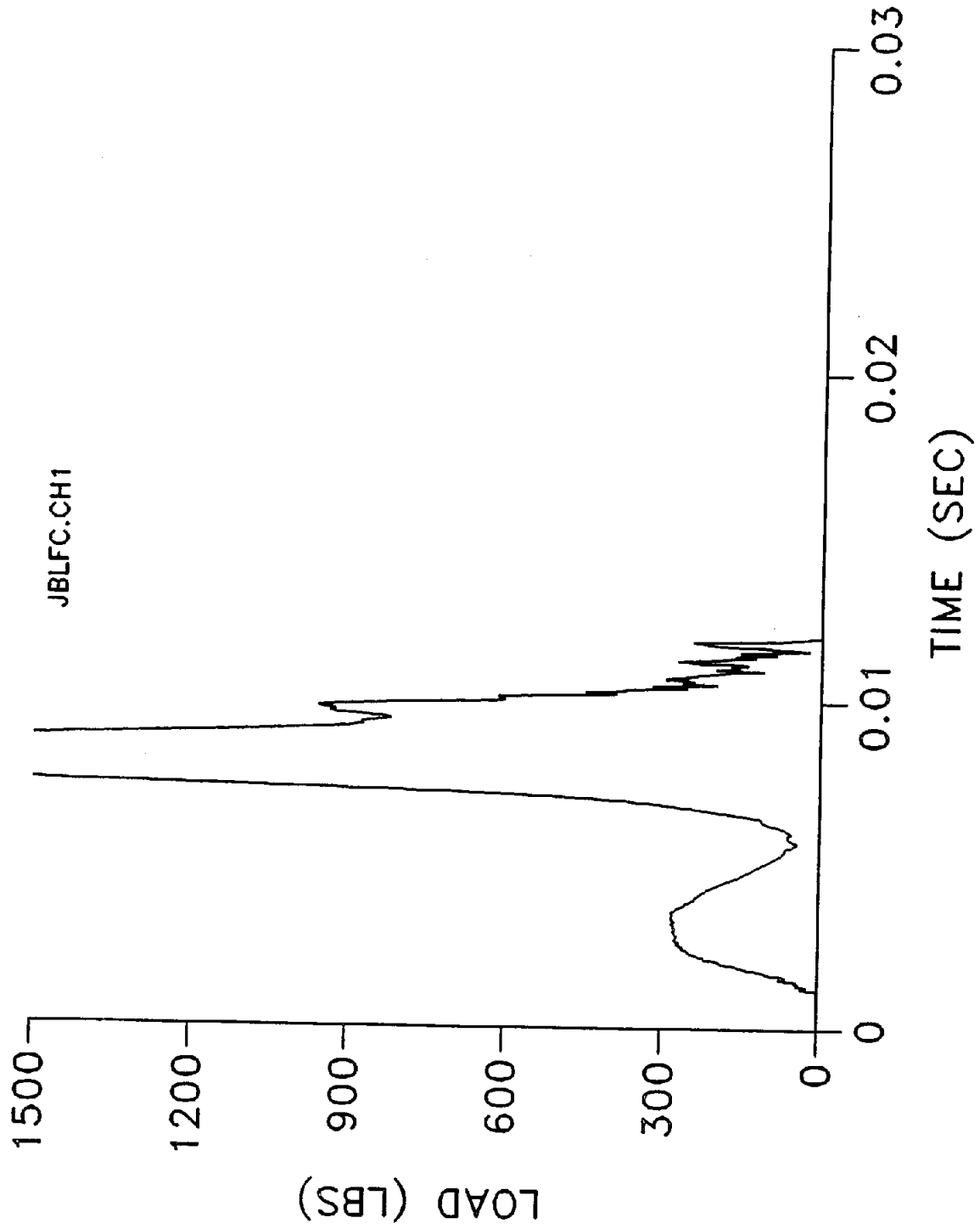

MTSIAFC.CH2
150
120
90
60
30
0
ACCELERATION (G'S)
0
0.01
0.02
0.03
TIME (SEC)

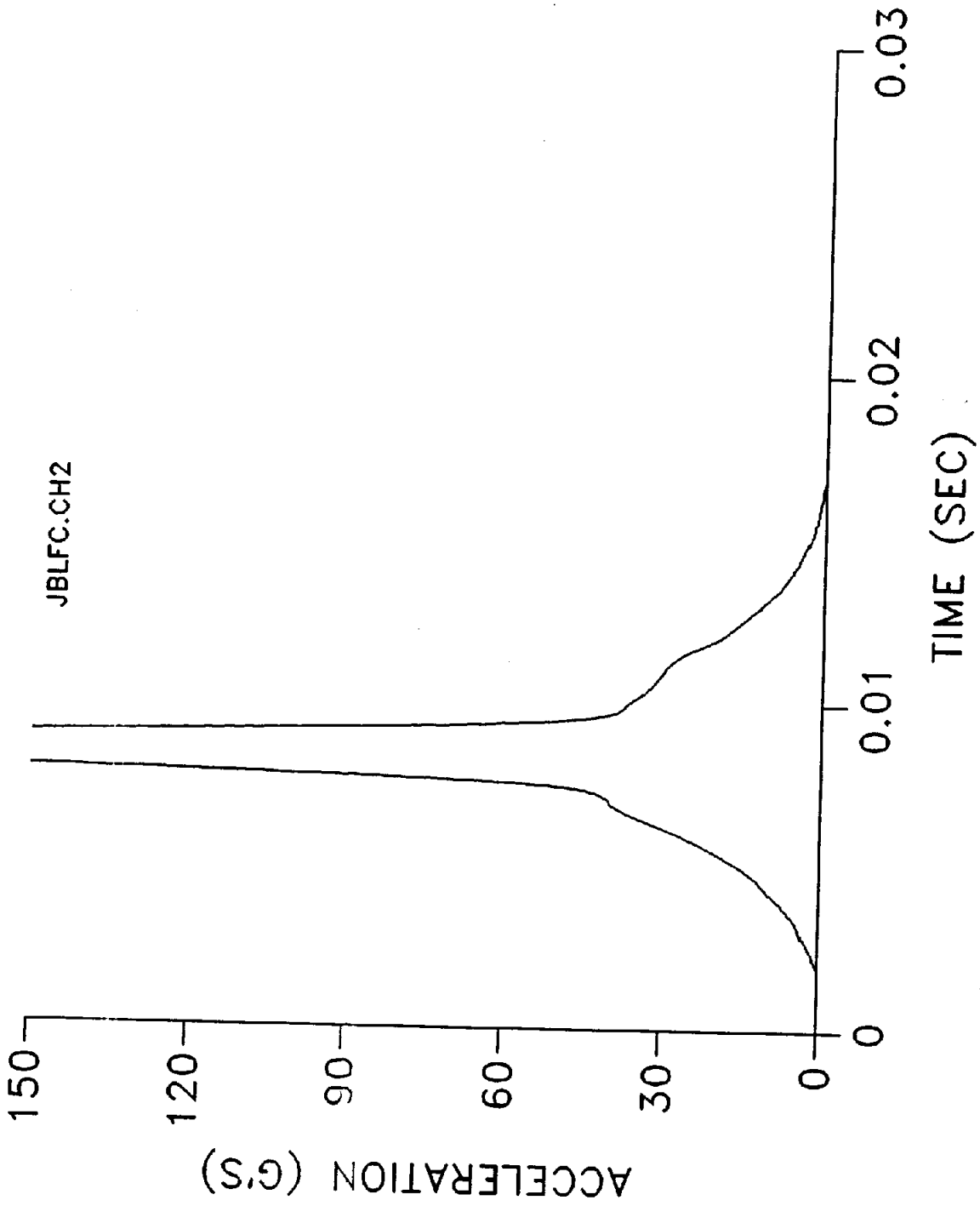

ABLFC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
0
0.01
0.02
0.03
TIME (SEC)

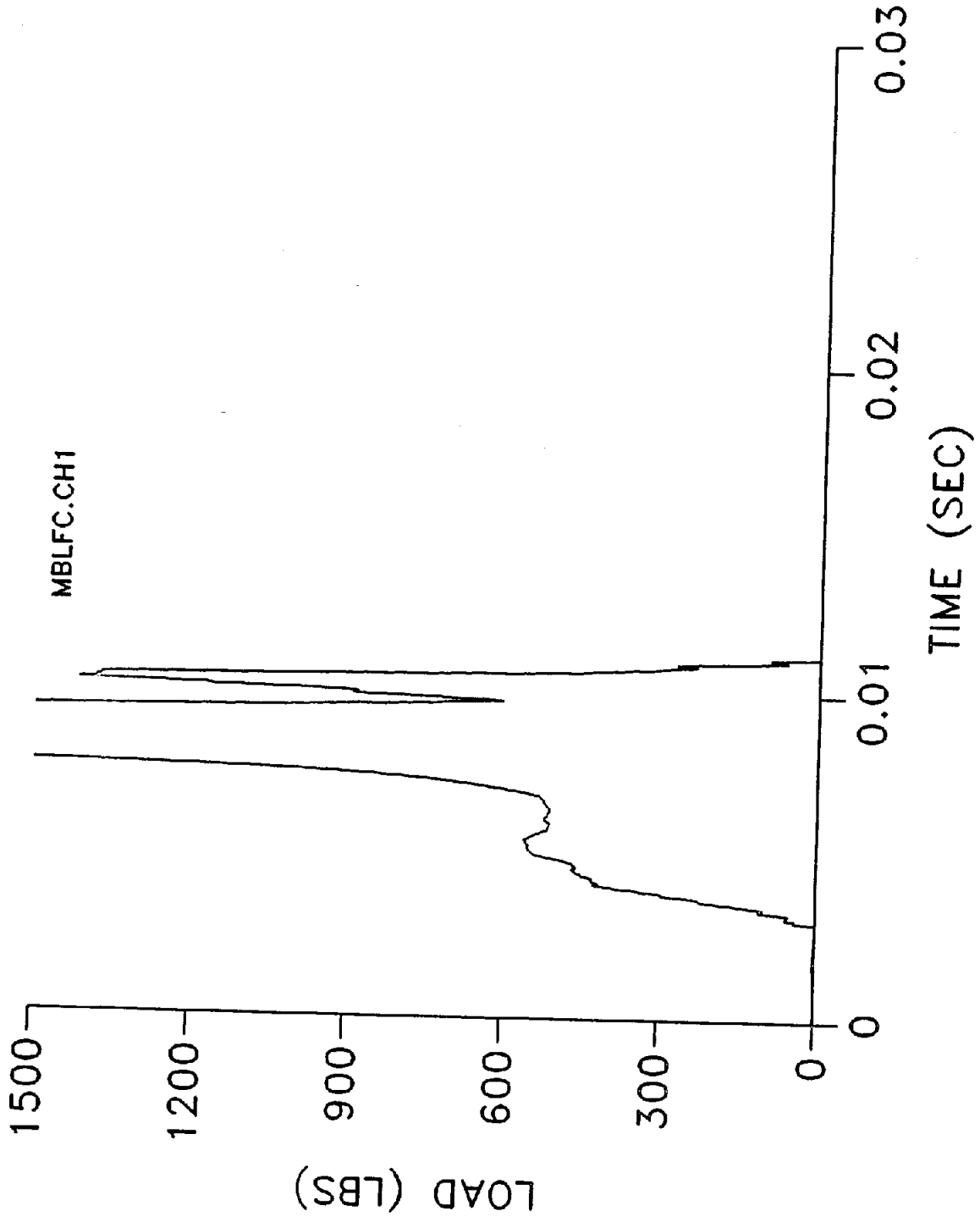

ABLFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)

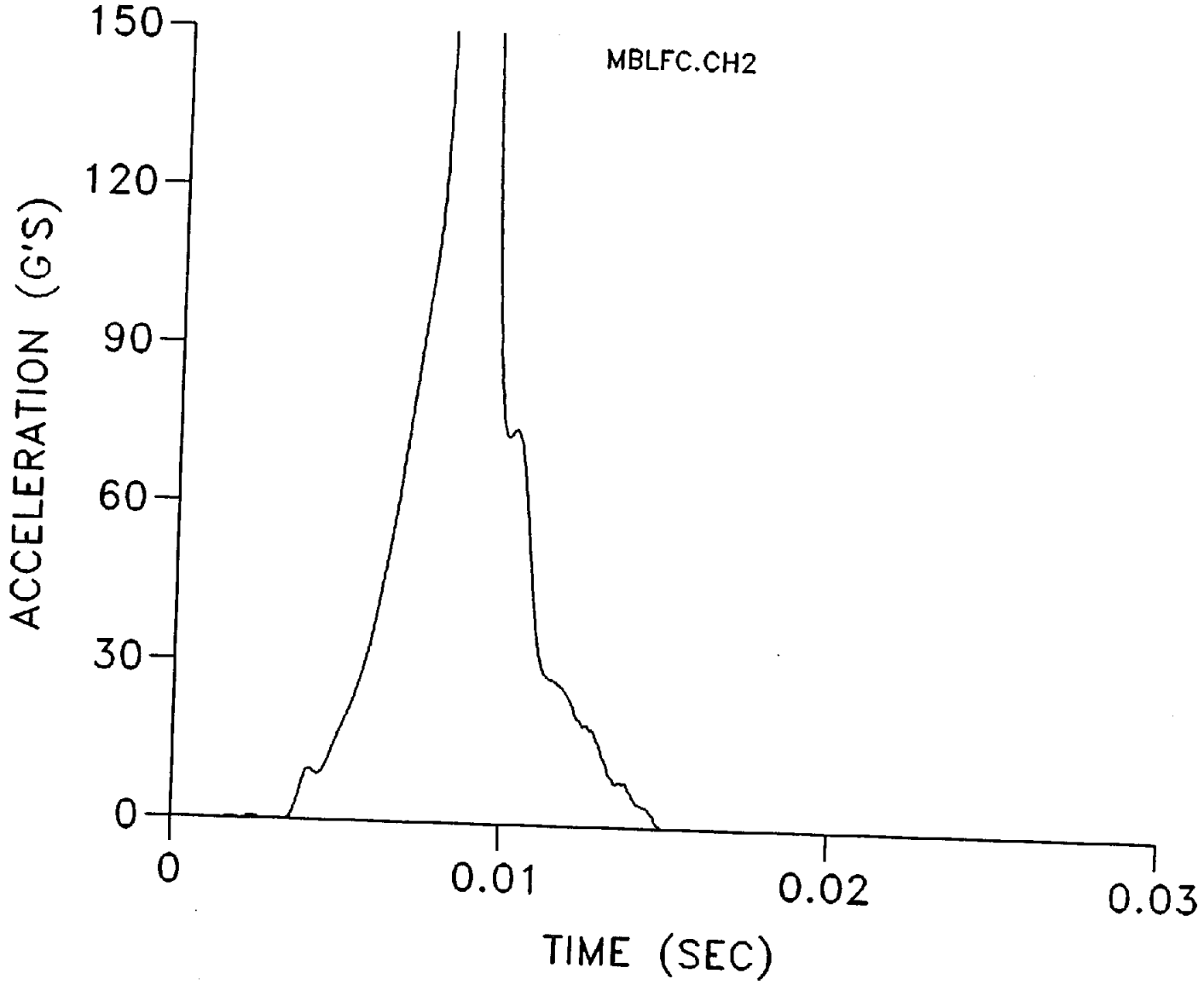

EBLFC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
0
0.01
0.02
0.03
TIME (SEC)

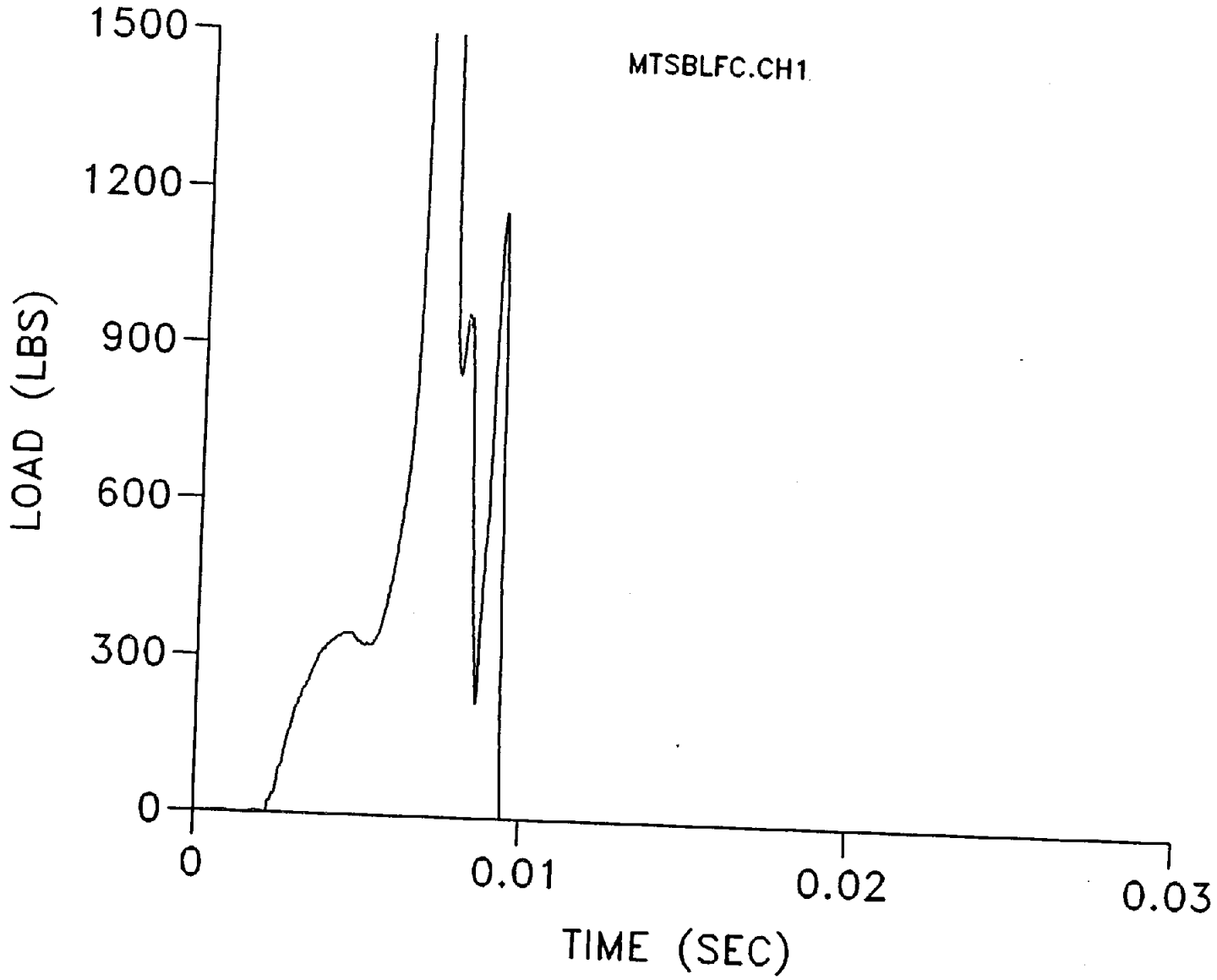

EBLFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)


JBLFC.CH1
LOAD (LBS)
1500
1200
900
600
300
0
0
0.01
0.02
0.03
TIME (SEC)


JBLFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)


MBLFC.CH1
LOAD (LBS)
1500
1200
900
600
300
0
0
0.01
0.02
0.03
TIME (SEC)


MBLFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)


MTSBLFC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
0
0.01
0.02
0.03
TIME (SEC)

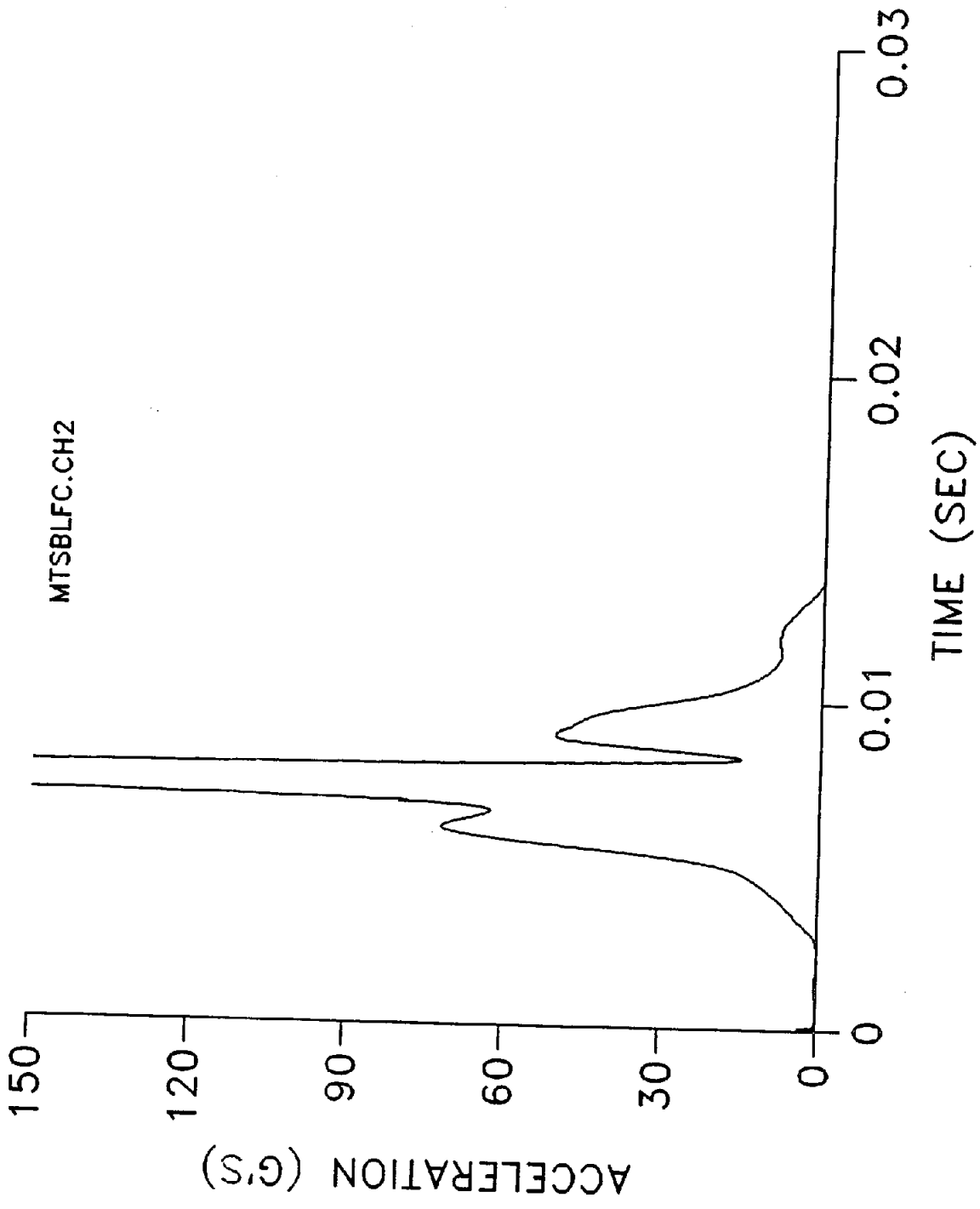
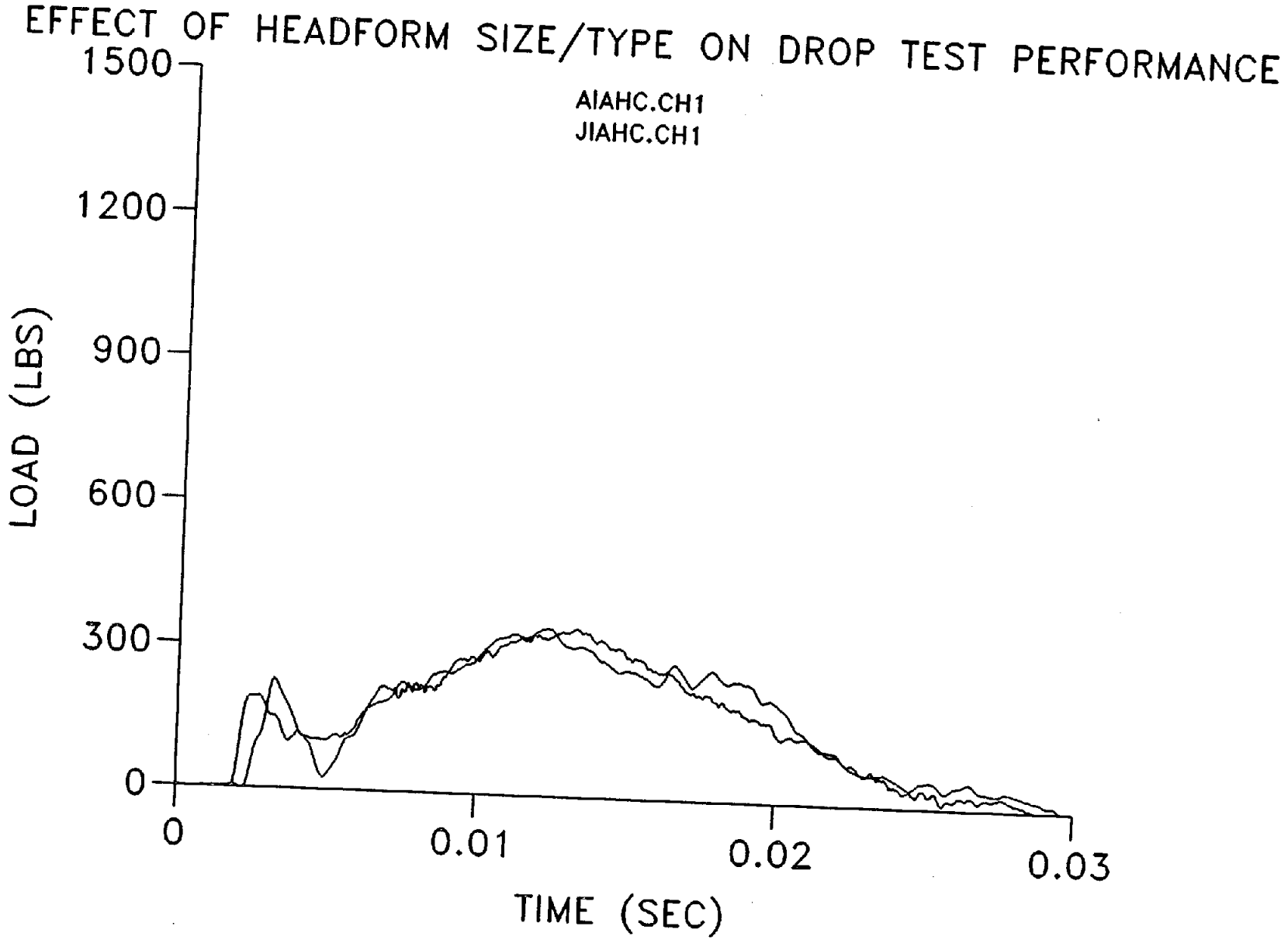
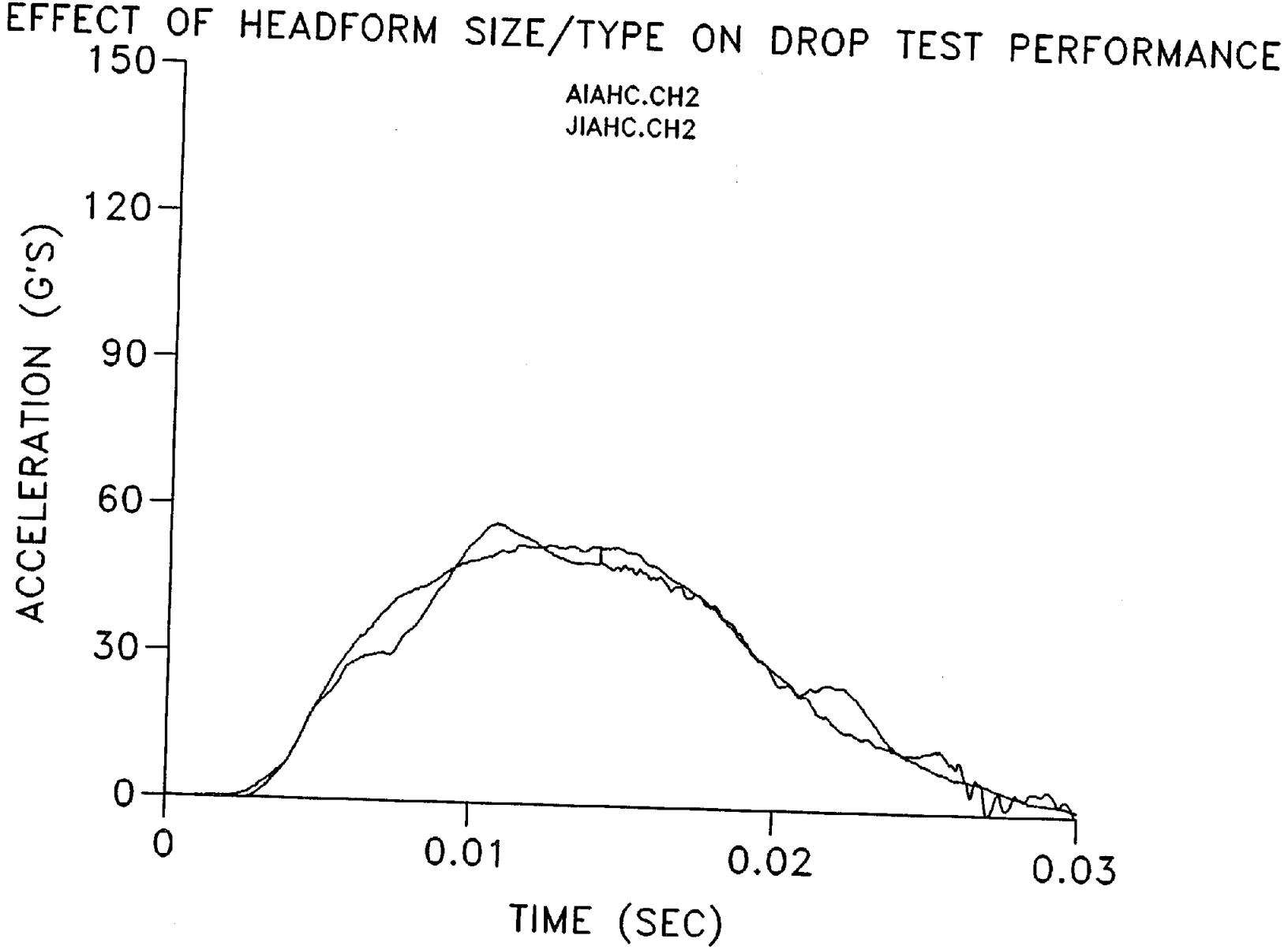

MTSBLFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)

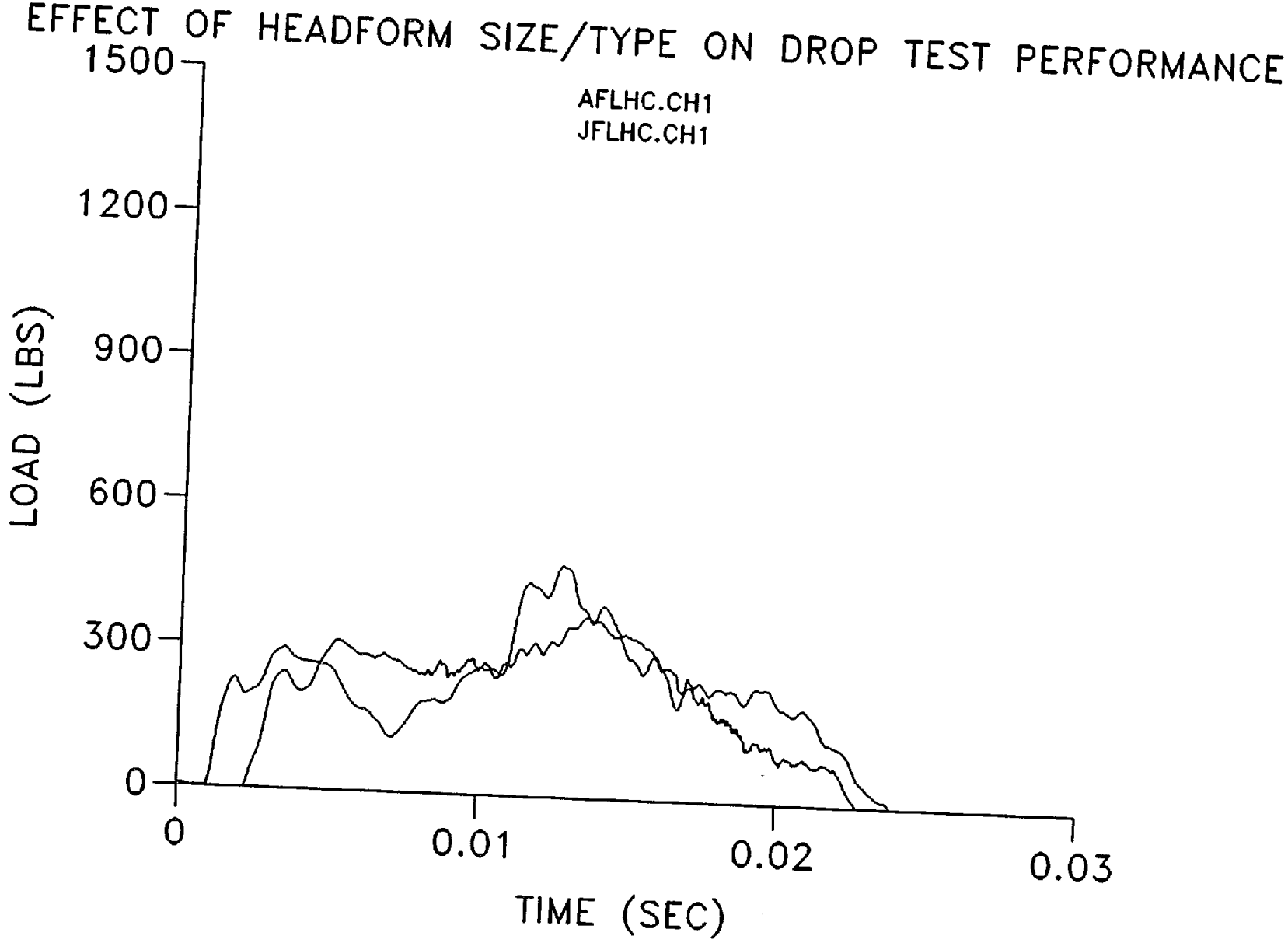
TABLE C.3

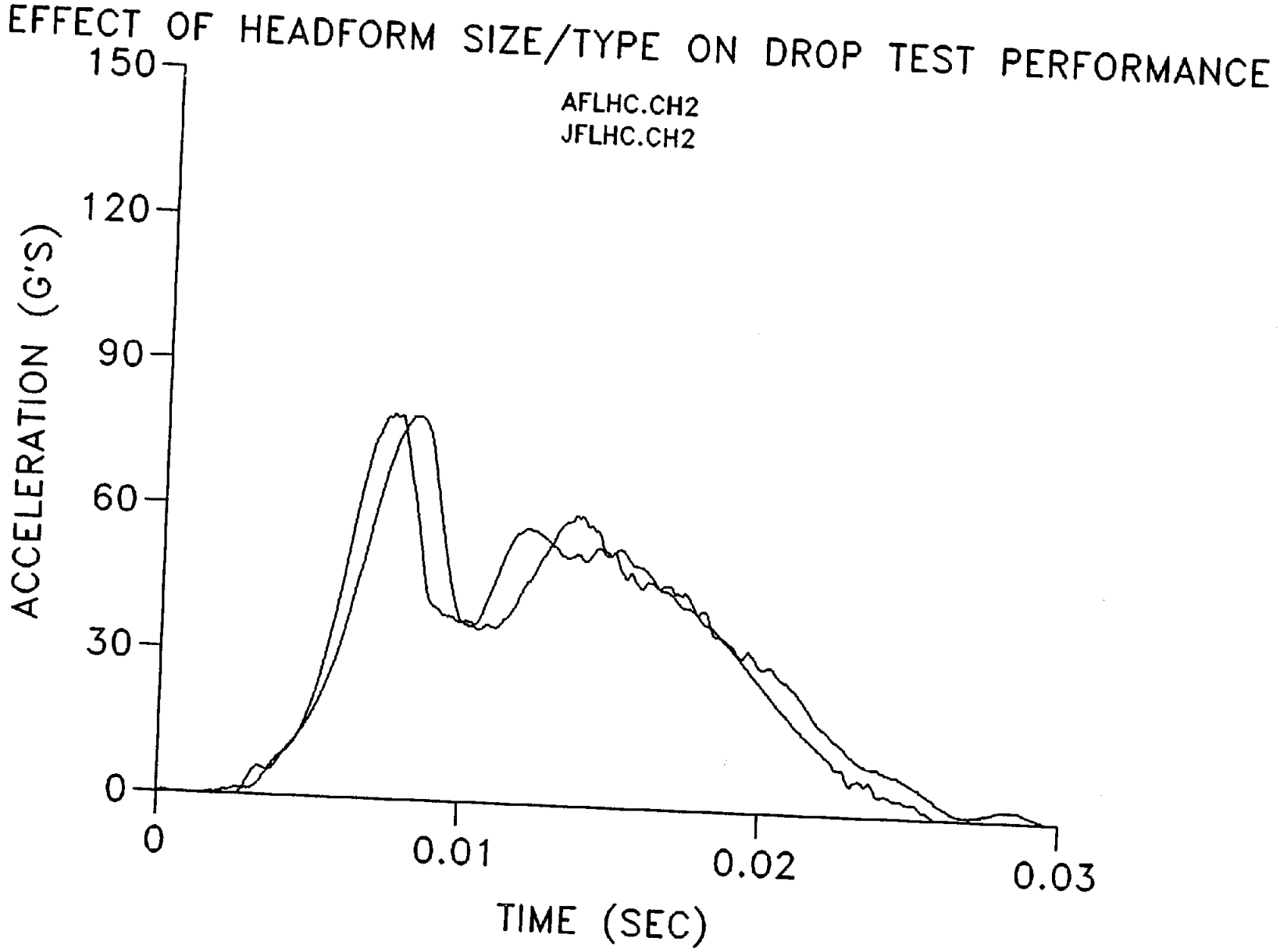
EFFECT OF HEADFORM SIZE/TYPE ON DROP TEST PERFORMANCE

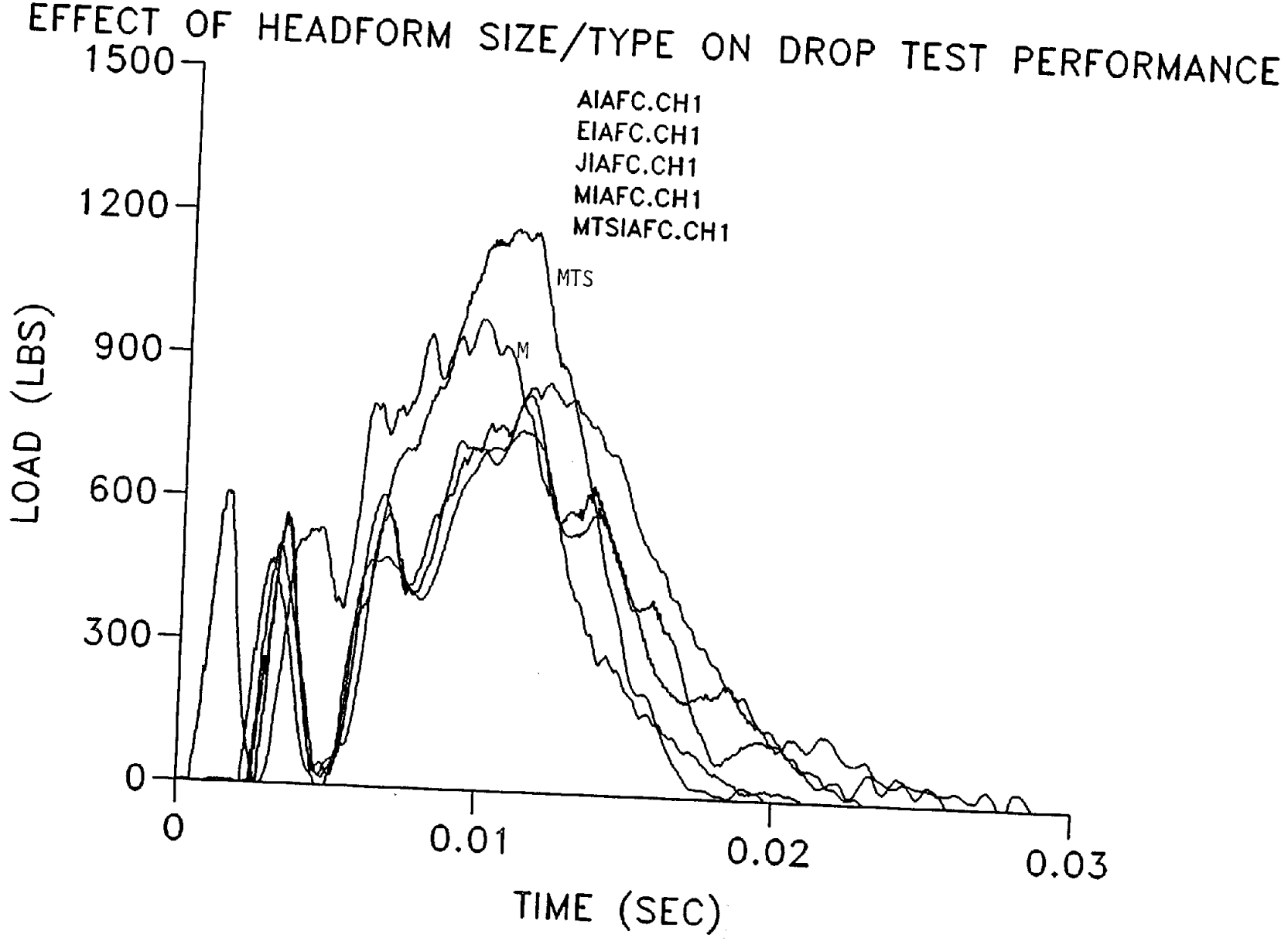
FILE	POINTS	NAME	CHANNEL 1 CAL (LBS/V)	CHANNEL 1 OFFSET (V)	CHANNEL 1 MIN (LBS)	CHANNEL 1 MAX (LBS)	NAME	CHANNEL 2 CAL (G'S/V)	CHANNEL 2 OFFSET (V)	CHANNEL 2 MIN (G'S)	CHANNEL 2 MAX (G'S)
AIAHC	16384	LOAD	258.000	0.020	-27.756	350.174	SI ACCEL	-50.000	0.034	-1.981	57.345
JIAHC	16384	LOAD	258.000	0.021	-11.661	351.151	SI ACCEL	-50.000	0.034	-0.760	53.195
AFLHC	16384	LOAD	258.000	0.015	-45.402	482.440	SI ACCEL	-50.000	0.037	-1.083	79.727
JFLHC	16384	LOAD	258.000	0.016	-79.603	376.432	SI ACCEL	-50.000	0.022	-1.828	79.471
AIAFC	16384	LOAD	258.000	0.020	-40.484	759.468	SI ACCEL	-50.000	0.029	-3.681	72.735
EIAFC	8192	LOAD	258.000	0.018	-12.192	833.111	SI ACCEL	-50.000	0.025	-147.686	77.240
JIAFC	16384	LOAD	258.000	0.021	-30.671	862.503	SI ACCEL	-50.000	0.020	-3.663	76.171
MIAFC	16384	LOAD	258.000	0.021	-19.172	988.640	SI ACCEL	-50.000	0.034	-1.714	89.839
MTSIAFC	8192	LOAD	258.000	0.017	-18.211	1182.345	SI ACCEL	-50.000	0.028	-2.732	104.446
AFLFC	16384	LOAD	258.000	0.020	-27.750	1147.611	SI ACCEL	-50.000	0.040	-4.095	104.792
EFLFC	16384	LOAD	258.000	0.018	-3.307	1164.496	SI ACCEL	-50.000	0.048	-236.606	101.508
JFLFC	16384	LOAD	258.000	0.020	-55.627	1173.905	SI ACCEL	-50.000	0.033	-3.008	103.926
MFLFC	16384	LOAD	258.000	-0.041	-39.871	1213.595	SI ACCEL	-50.000	0.033	-4.934	109.079
MTSFLFC	8192	LOAD	258.000	0.019	-38.855	1185.637	SI ACCEL	-50.000	0.030	-444.812	102.277
ABLFC	16384	LOAD	258.000	0.369	-146.901	2483.490	SI ACCEL	-50.000	0.027	-4.261	271.130
EBLFC	16384	LOAD	258.000	-0.030	-2.313	2586.506	SI ACCEL	-50.000	0.040	-6.814	333.274
JBLFC	16384	LOAD	258.000	0.018	-148.136	2574.218	SI ACCEL	-50.000	0.034	-1.209	261.975
MBLFC	16384	LOAD	258.000	0.021	-84.868	2573.238	SI ACCEL	-50.000	0.019	-6.869	406.705
MTSBLFC	8192	LOAD	258.000	0.021	-237.270	2573.267	SI ACCEL	-50.000	0.038	-335.507	501.895

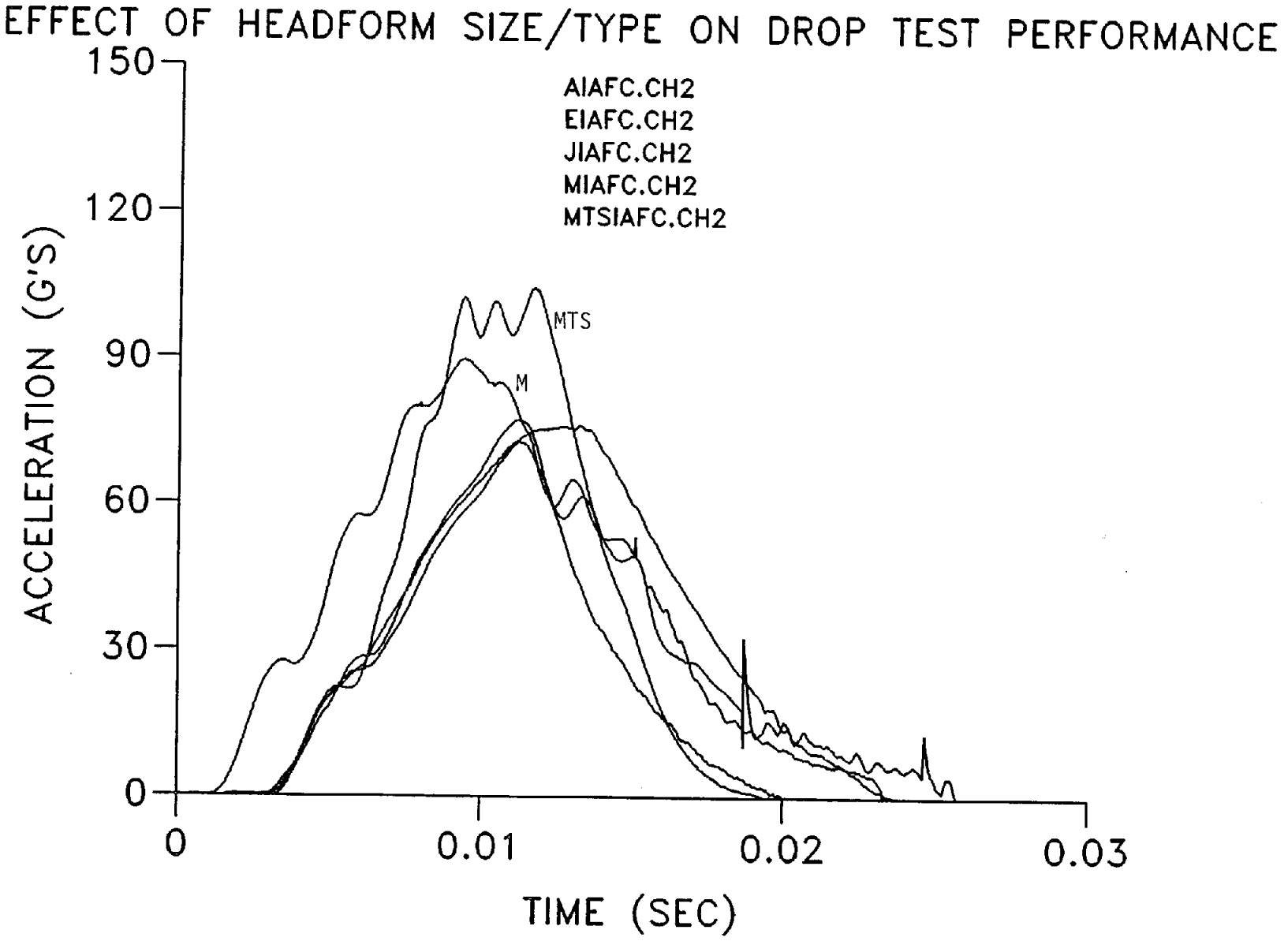

EFFECT OF HEADFORM SIZE/TYPE ON DROP TEST PERFORMANCE
AIAHC.CH1
JIAHC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
0
0.01
0.02
0.03
TIME (SEC)

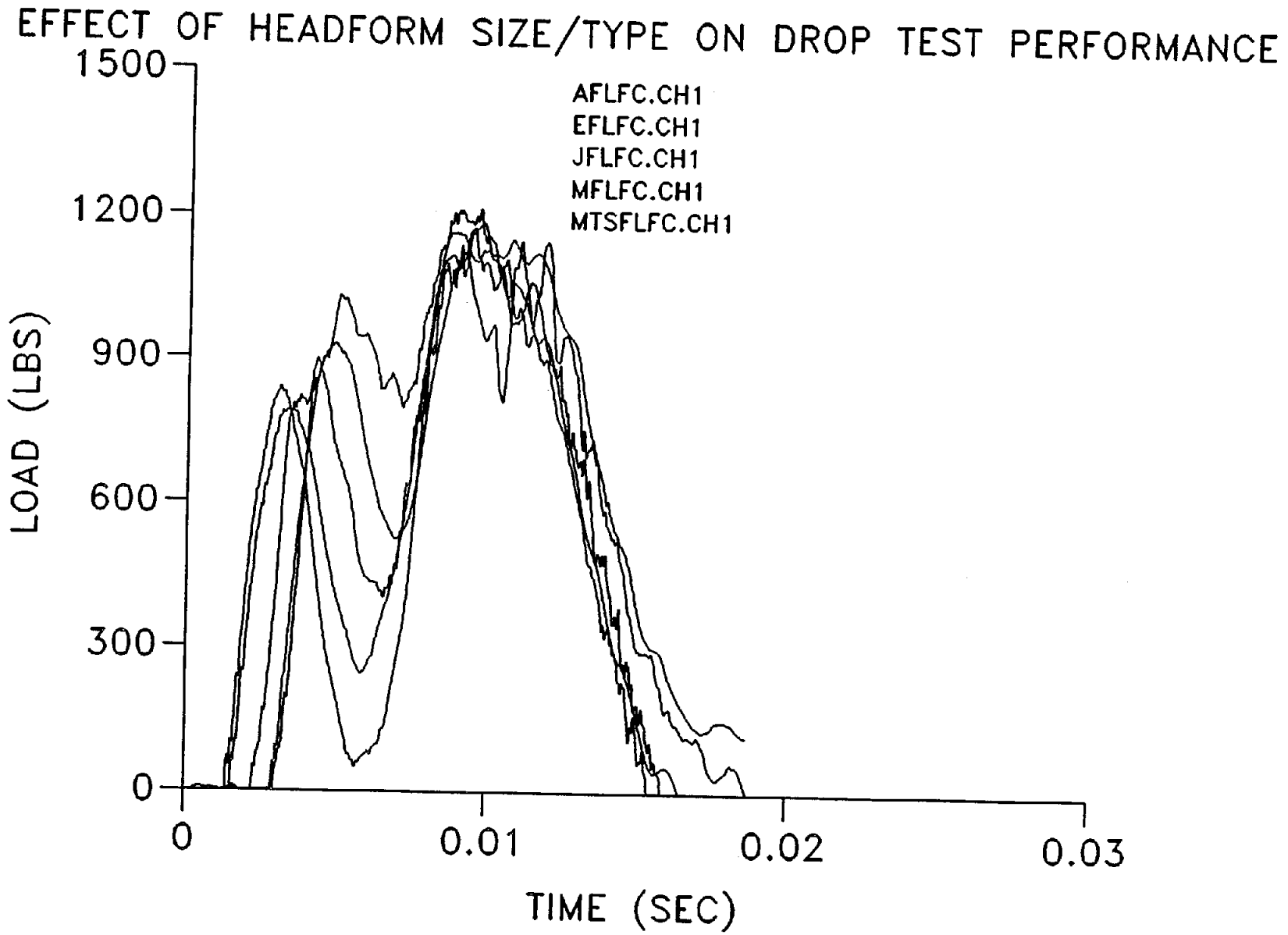

EFFECT OF HEADFORM SIZE/TYPE ON DROP TEST PERFORMANCE
AIAHC.CH2
JIAHC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)

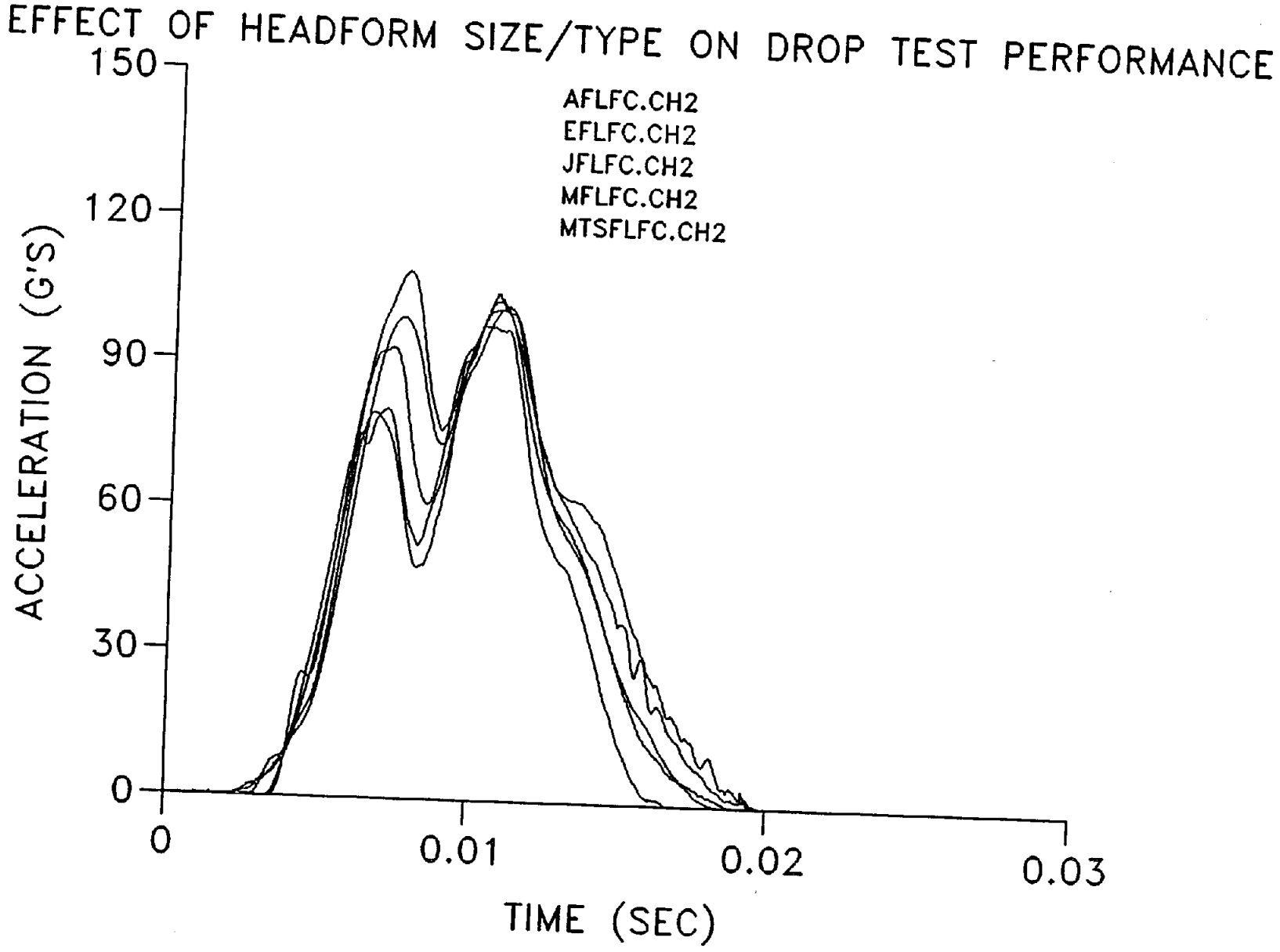

EFFECT OF HEADFORM SIZE/TYPE ON DROP TEST PERFORMANCE
AFLHC.CH1
JFLHC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
0
0.01
0.02
0.03
TIME (SEC)

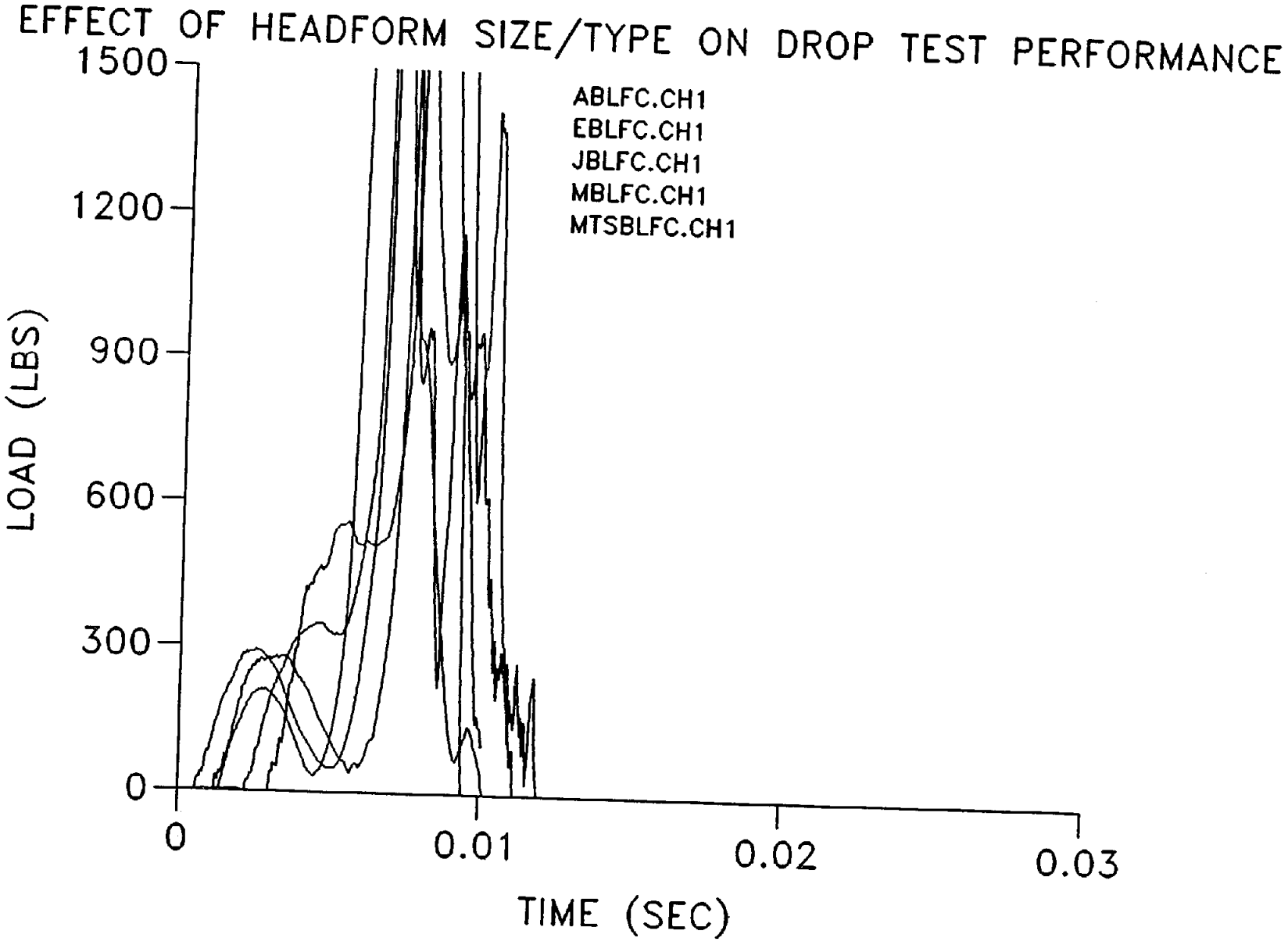

EFFECT OF HEADFORM SIZE/TYPE ON DROP TEST PERFORMANCE
AFLHC.CH2
JFLHC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)


EFFECT OF HEADFORM SIZE/TYPE ON DROP TEST PERFORMANCE
AIAFC.CH1
EIAFC.CH1
JIAFC.CH1
MIAFC.CH1
MTSIAFC.CH1
MTS
M
LOAD (LBS)
1500
1200
900
600
300
0
0
0.01
0.02
0.03
TIME (SEC)


EFFECT OF HEADFORM SIZE/TYPE ON DROP TEST PERFORMANCE
AIAFC.CH2
EIAFC.CH2
JIAFC.CH2
MIAFC.CH2
MTSIAFC.CH2
MTS
M
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)


EFFECT OF HEADFORM SIZE/TYPE ON DROP TEST PERFORMANCE
AFLFC.CH1
EFLFC.CH1
JFLFC.CH1
MFLFC.CH1
MTSFLFC.CH1
LOAD (LBS)
1500
1200
900
600
300
0
0
0.01
0.02
0.03
TIME (SEC)


EFFECT OF HEADFORM SIZE/TYPE ON DROP TEST PERFORMANCE
AFLFC.CH2
EFLFC.CH2
JFLFC.CH2
MFLFC.CH2
MTSFLFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)


EFFECT OF HEADFORM SIZE/TYPE ON DROP TEST PERFORMANCE
ABLFC.CH1
EBLFC.CH1
JBLFC.CH1
MBLFC.CH1
MTSBLFC.CH1
LOAD (LBS)
1500
1200
900
600
300
0
0
0.01
0.02
0.03
TIME (SEC)

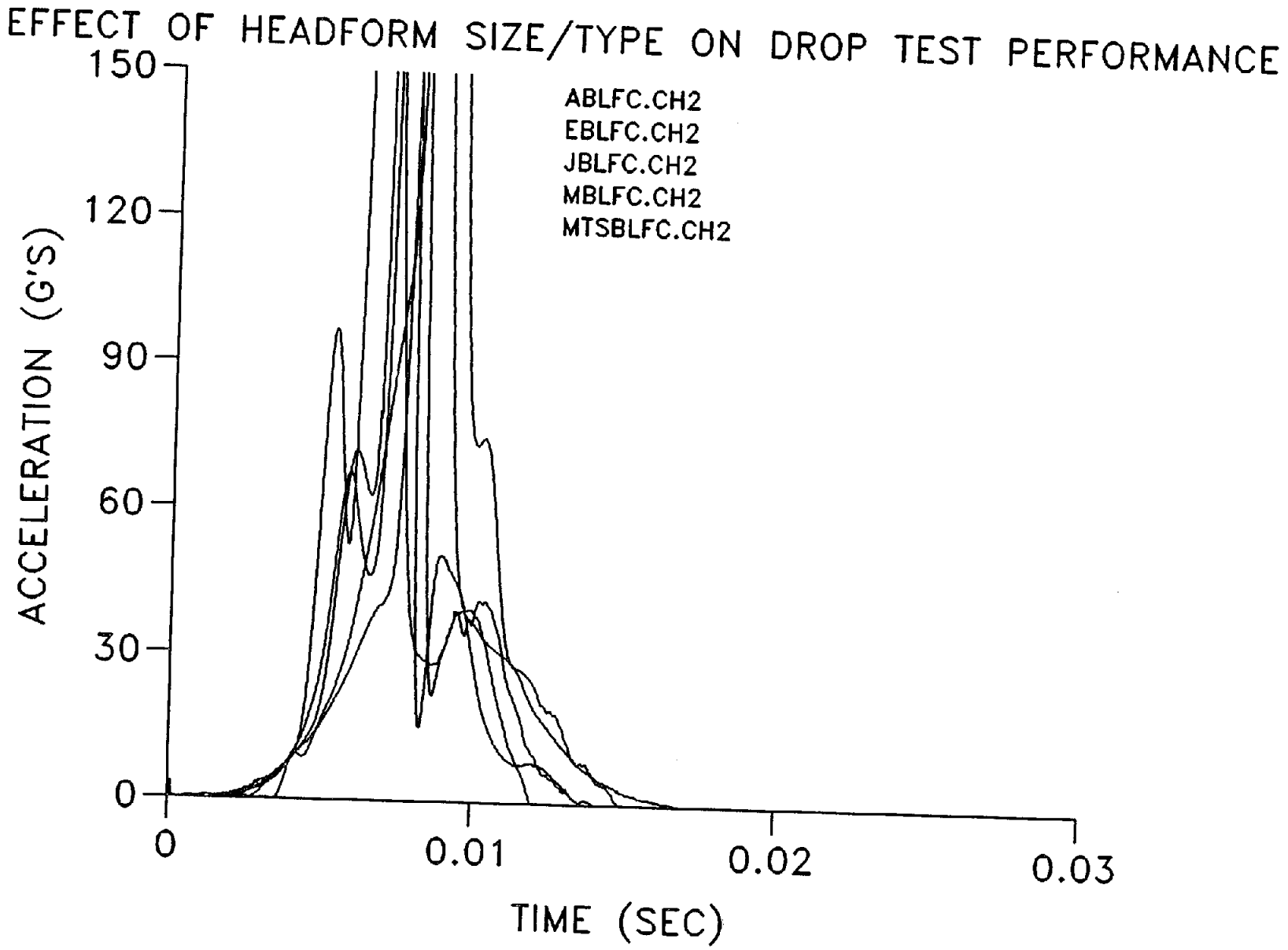
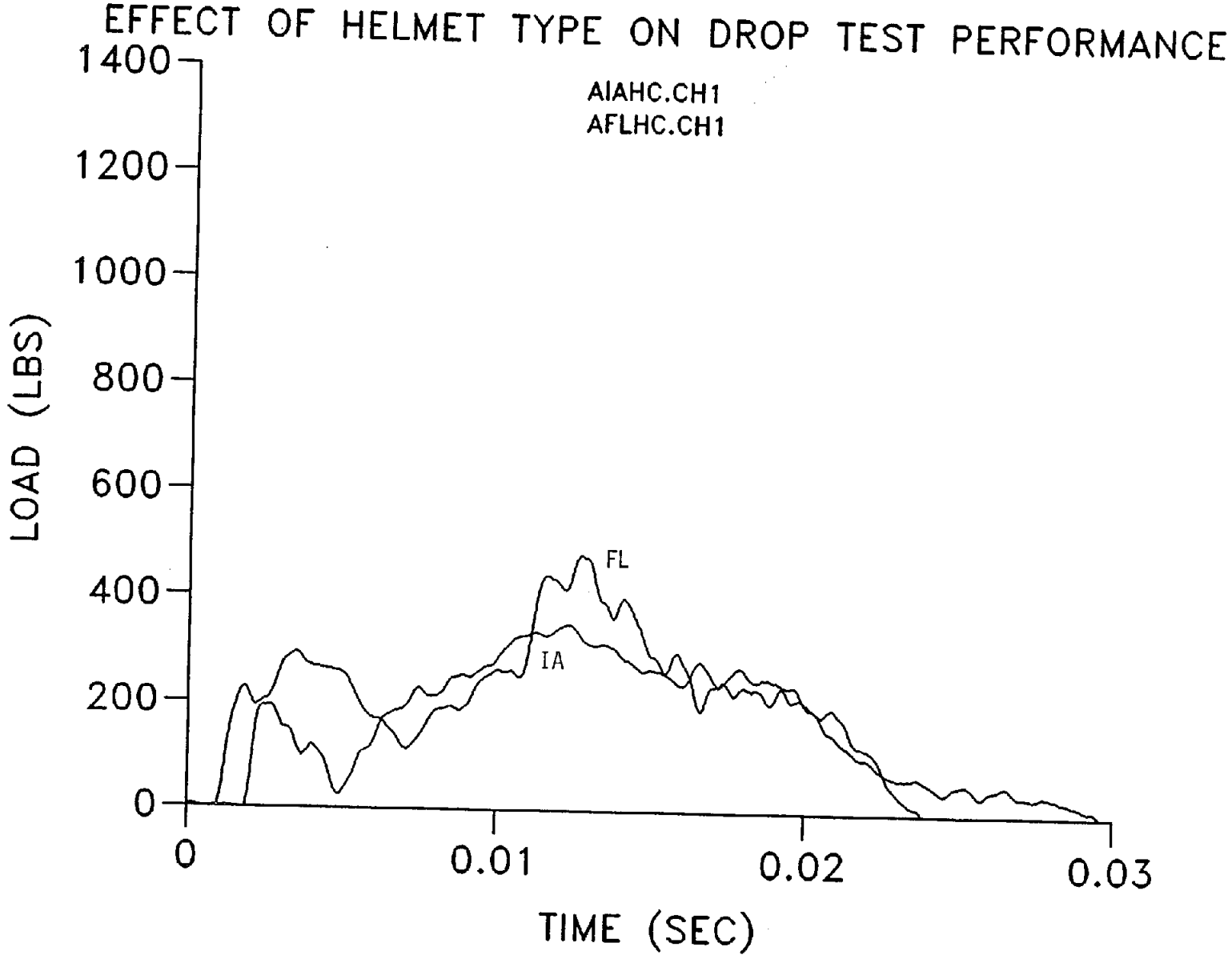
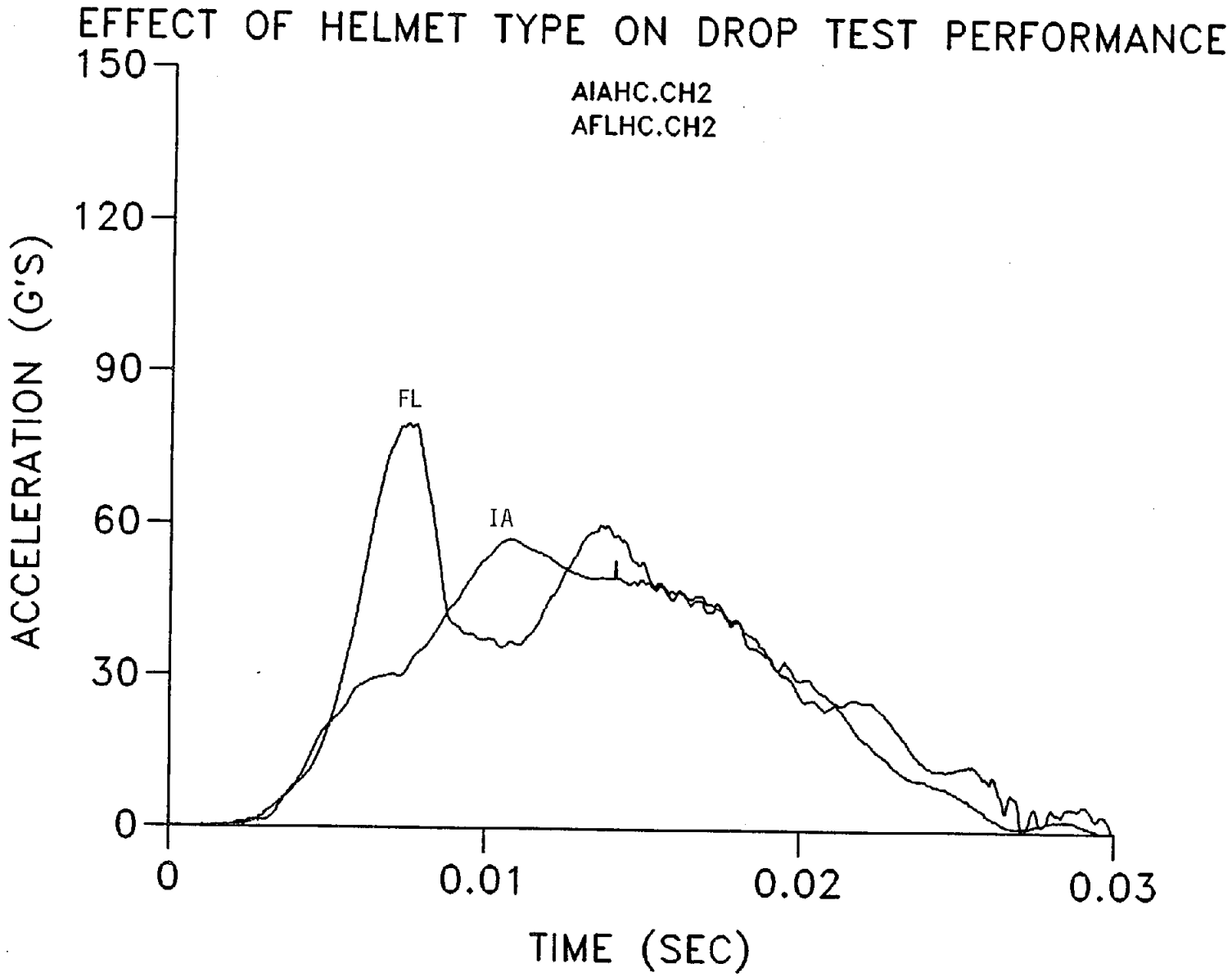
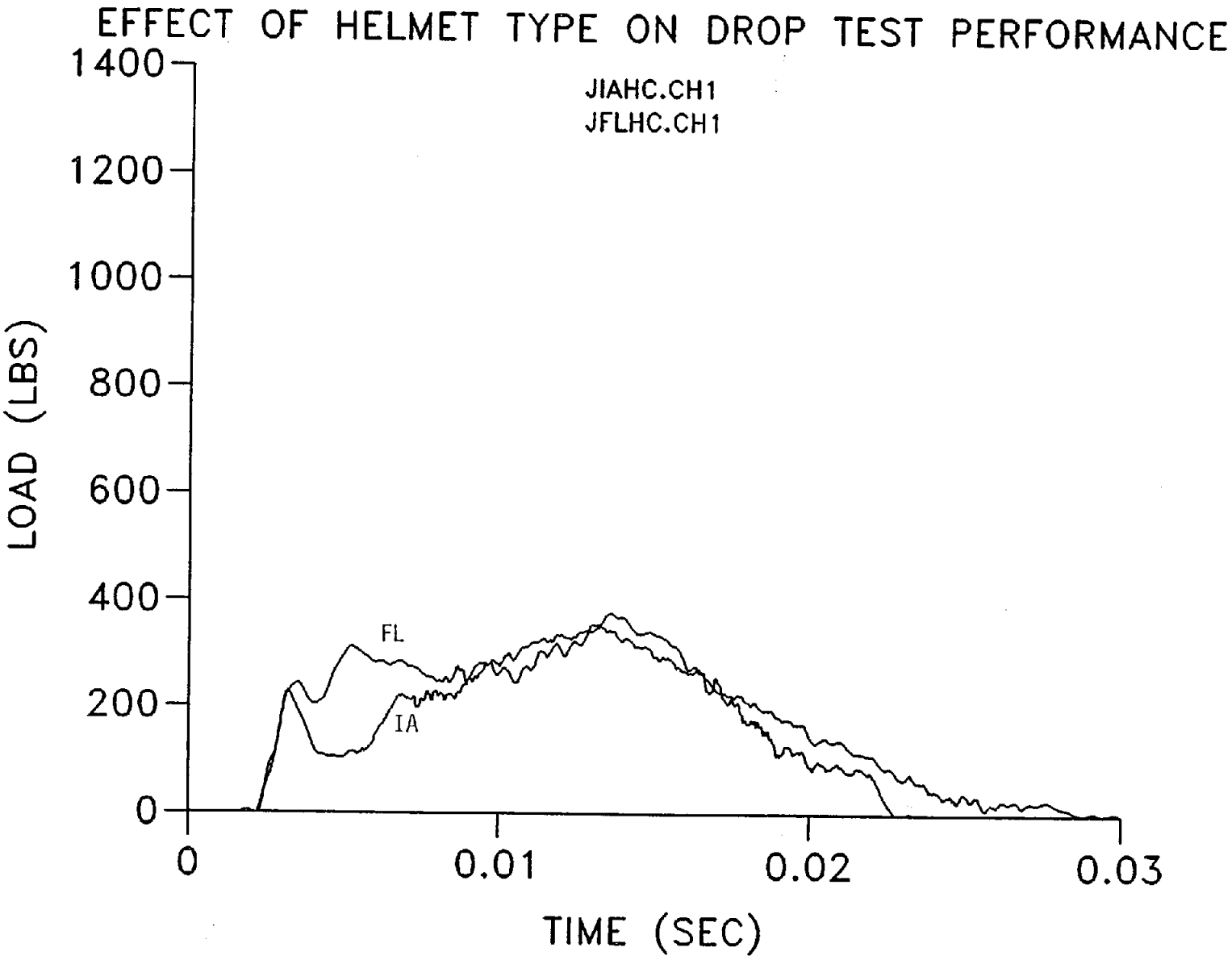

EFFECT OF HEADFORM SIZE/TYPE ON DROP TEST PERFORMANCE
ABLFC.CH2
EBLFC.CH2
JBLFC.CH2
MBLFC.CH2
MTSBLFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)

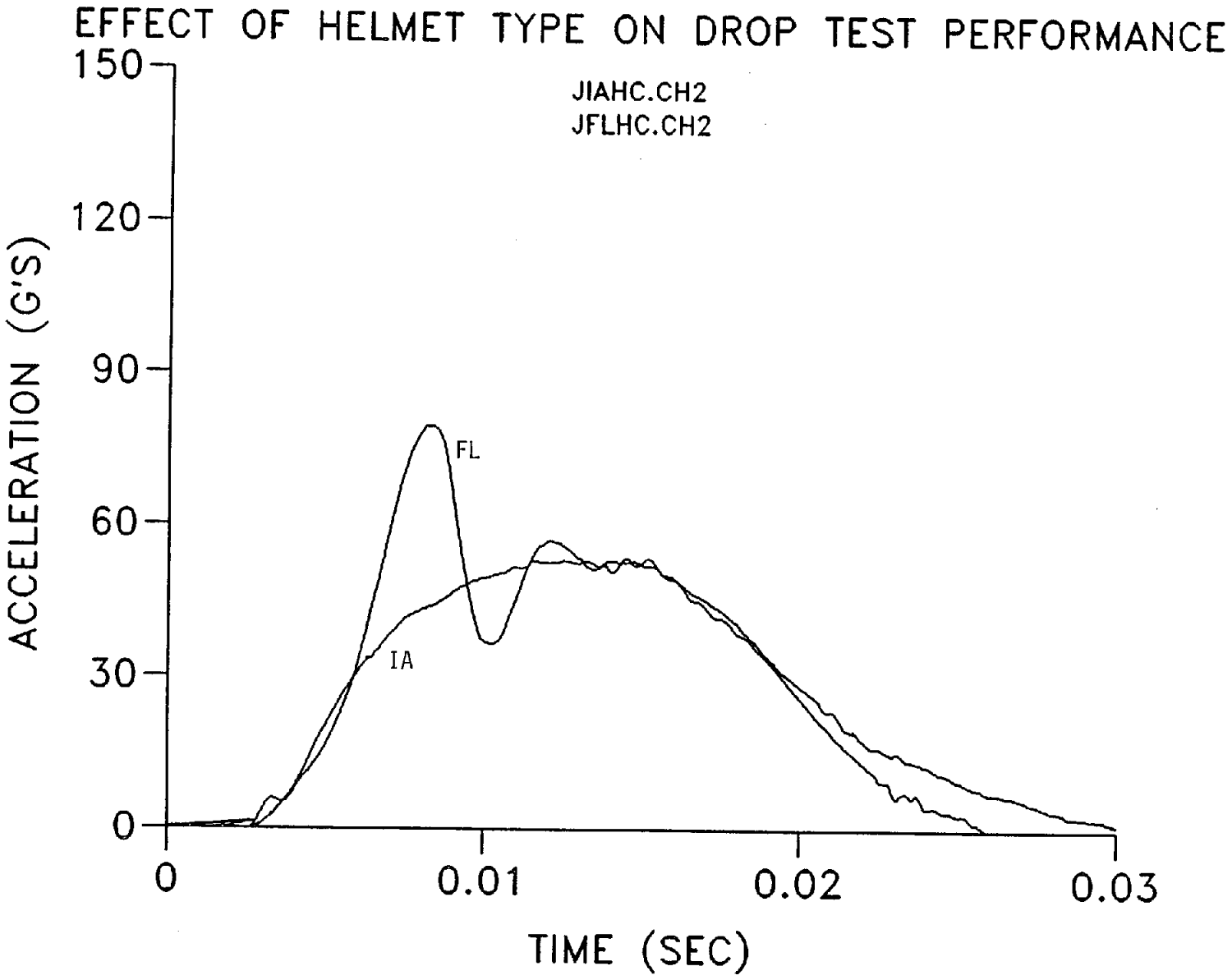
TABLE C.4
EFFECT OF HELMET TYPE ON DROP TEST PERFORMANCE

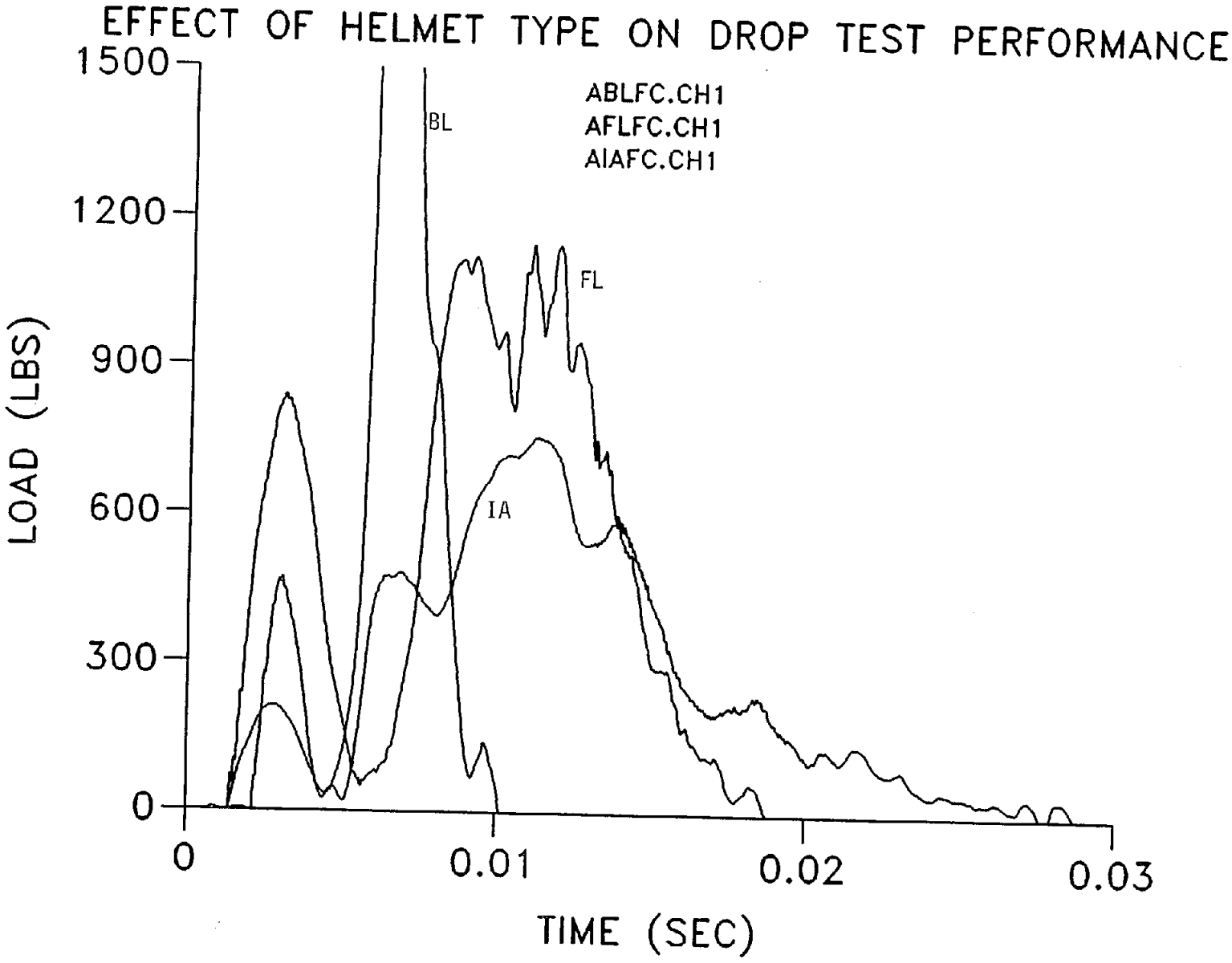
FILE	POINTS	CHANNEL 1 NAME	CAL (LBS/V)	OFFSET (V)	MIN (LBS)	MAX (LBS)	CHANNEL 2 NAME	CAL (G'S/V)	OFFSET (V)	MIN (G'S)	MAX (G'S)
AIAHC	16384	LOAD	258.000	0.020	-27.756	350.174	SI ACCEL	-50.000	0.034	-1.981	57.345
AFLHC	16384	LOAD	258.000	0.015	-45.402	482.440	SI ACCEL	-50.000	0.037	-1.083	79.727
JIAHC	16384	LOAD	258.000	0.021	-11.661	351.151	SI ACCEL	-50.000	0.034	-0.760	53.195
JFLHC	16384	LOAD	258.000	0.016	-79.603	376.432	SI ACCEL	-50.000	0.022	-1.828	79.471
AIAFC	16384	LOAD	258.000	0.020	-40.484	759.468	SI ACCEL	-50.000	0.029	-3.681	72.735
AFLFC	16384	LOAD	258.000	0.020	-27.750	1147.611	SI ACCEL	-50.000	0.040	-4.095	104.792
ABLFC	16384	LOAD	258.000	0.369	-146.901	2483.490	SI ACCEL	-50.000	0.027	-4.261	271.130
EIAFC	8192	LOAD	258.000	0.018	-12.192	833.111	SI ACCEL	-50.000	0.025	-147.686	77.240
EFLFC	16384	LOAD	258.000	0.018	-3.307	1164.496	SI ACCEL	-50.000	0.048	-236.606	101.508
EBLFC	16384	LOAD	258.000	-0.030	-2.313	2586.506	SI ACCEL	-50.000	0.040	-6.814	333.274
JIAFC	16384	LOAD	258.000	0.021	-30.671	862.503	SI ACCEL	-50.000	0.020	-3.663	76.171
JFLFC	16384	LOAD	258.000	0.020	-55.627	1173.905	SI ACCEL	-50.000	0.033	-3.008	103.926
JBLFC	16384	LOAD	258.000	0.018	-148.136	2574.218	SI ACCEL	-50.000	0.034	-1.209	261.975
MIAFC	16384	LOAD	258.000	0.021	-19.172	988.640	SI ACCEL	-50.000	0.034	-1.714	89.839
MFLFC	16384	LOAD	258.000	-0.041	-39.871	1213.595	SI ACCEL	-50.000	0.033	-4.934	109.079
MBLFC	16384	LOAD	258.000	0.021	-84.868	2573.238	SI ACCEL	-50.000	0.019	-6.869	406.705
MTSIAFC	8192	LOAD	258.000	0.017	-18.211	1182.345	SI ACCEL	-50.000	0.028	-2.732	104.446
MTSFLFC	8192	LOAD	258.000	0.019	-38.855	1185.637	SI ACCEL	-50.000	0.030	-444.812	102.277
MTSBLFC	8192	LOAD	258.000	0.021	-237.270	2573.267	SI ACCEL	-50.000	0.038	-335.507	501.895

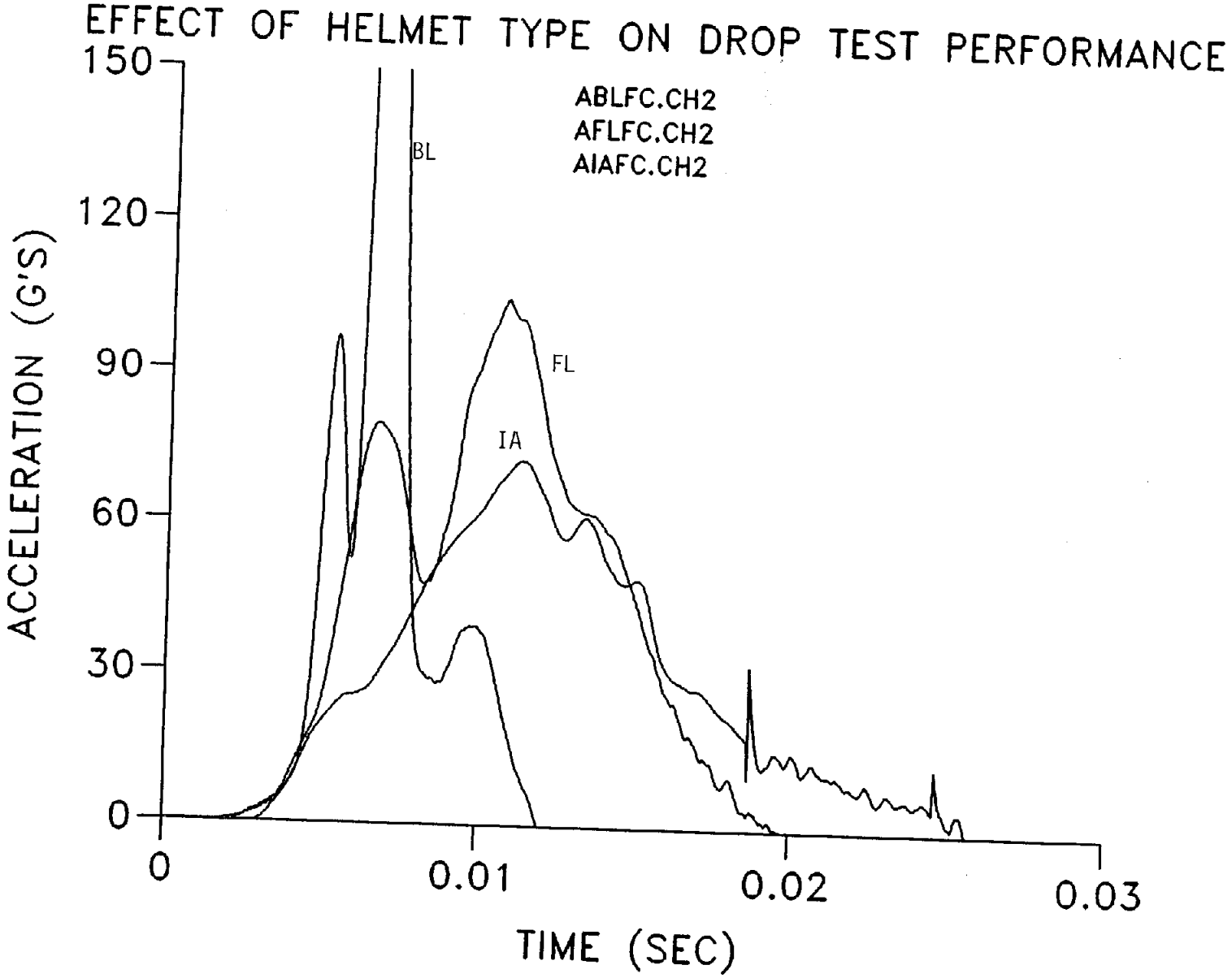

EFFECT OF HELMET TYPE ON DROP TEST PERFORMANCE
AIAHC.CH1
AFLHC.CH1
1400
1200
1000
800
600
400
200
0
LOAD (LBS)
FL
IA
0
0.01
0.02
0.03
TIME (SEC)

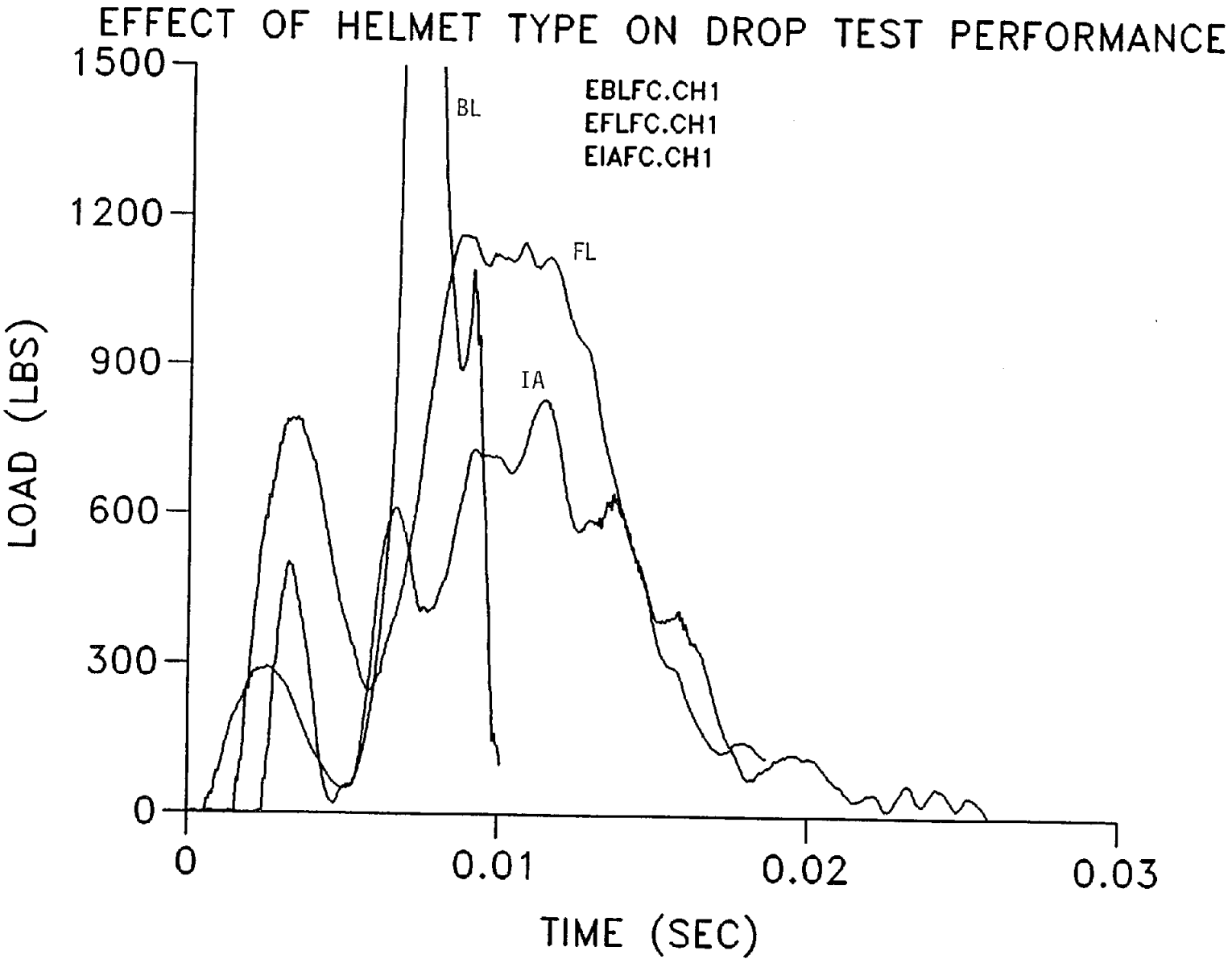

EFFECT OF HELMET TYPE ON DROP TEST PERFORMANCE
AIAHC.CH2
AFLHC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
FL
IA
0
0.01
0.02
0.03
TIME (SEC)

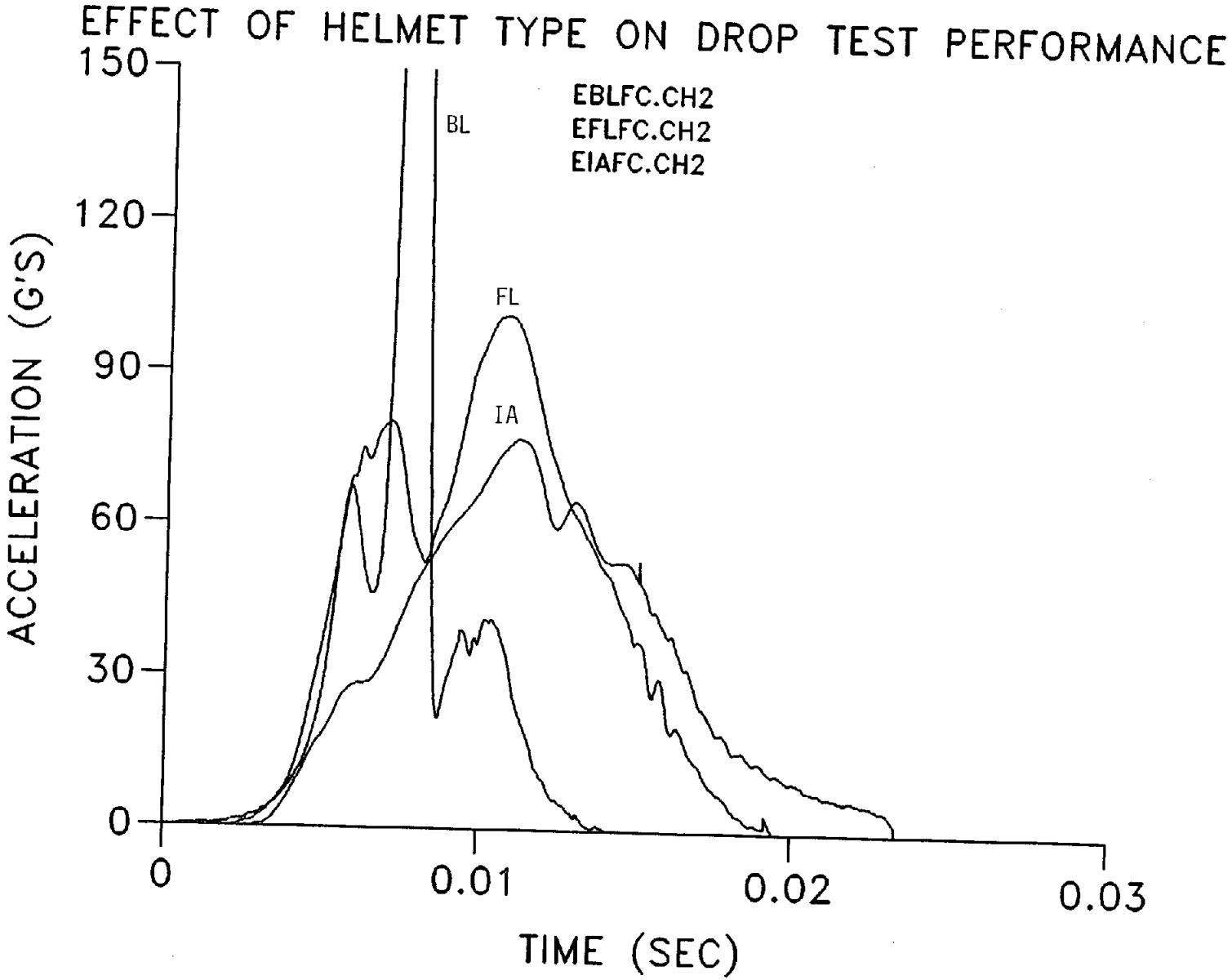

EFFECT OF HELMET TYPE ON DROP TEST PERFORMANCE
JIAHC.CH1
JFLHC.CH1
LOAD (LBS)
1400
1200
1000
800
600
400
200
0
FL
IA
0
0.01
0.02
0.03
TIME (SEC)

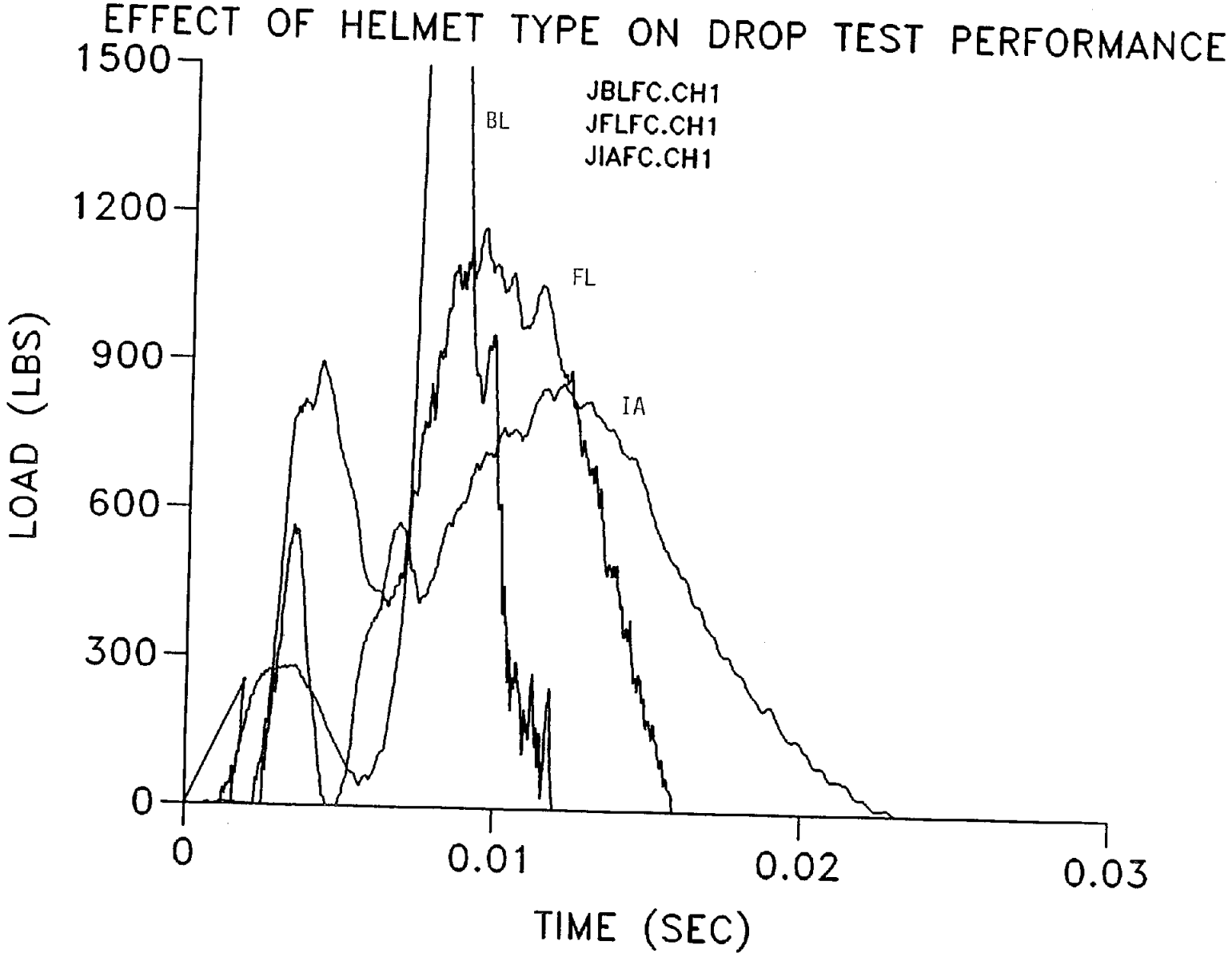

EFFECT OF HELMET TYPE ON DROP TEST PERFORMANCE
JIAHC.CH2
JFLHC.CH2
150
120
90
60
30
0
ACCELERATION (G'S)
FL
IA
0
0.01
0.02
0.03
TIME (SEC)

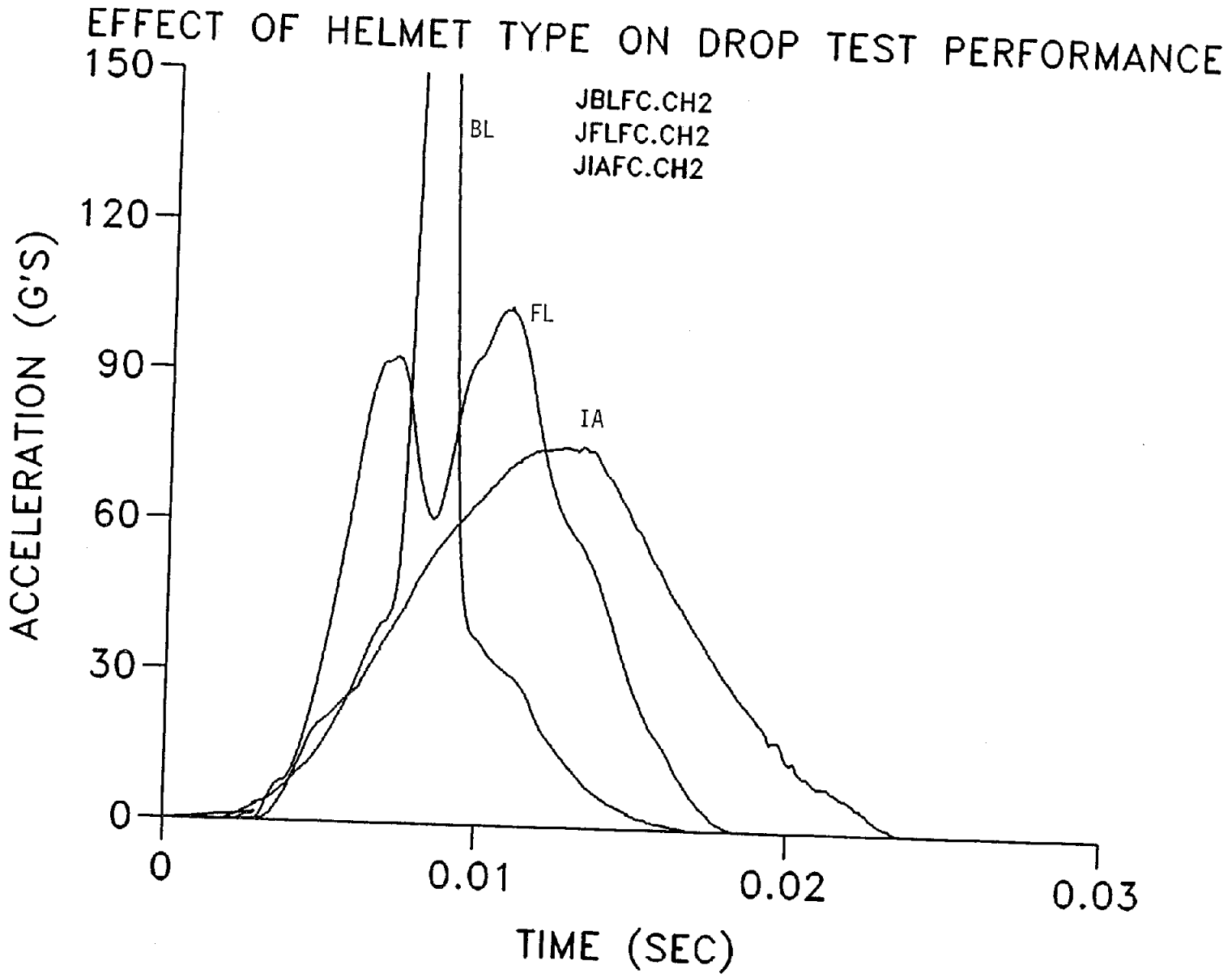

EFFECT OF HELMET TYPE ON DROP TEST PERFORMANCE
ABLFC.CH1
AFLFC.CH1
AIAFC.CH1
BL
FL
IA
LOAD (LBS)
1500
1200
900
600
300
0
0
0.01
0.02
0.03
TIME (SEC)

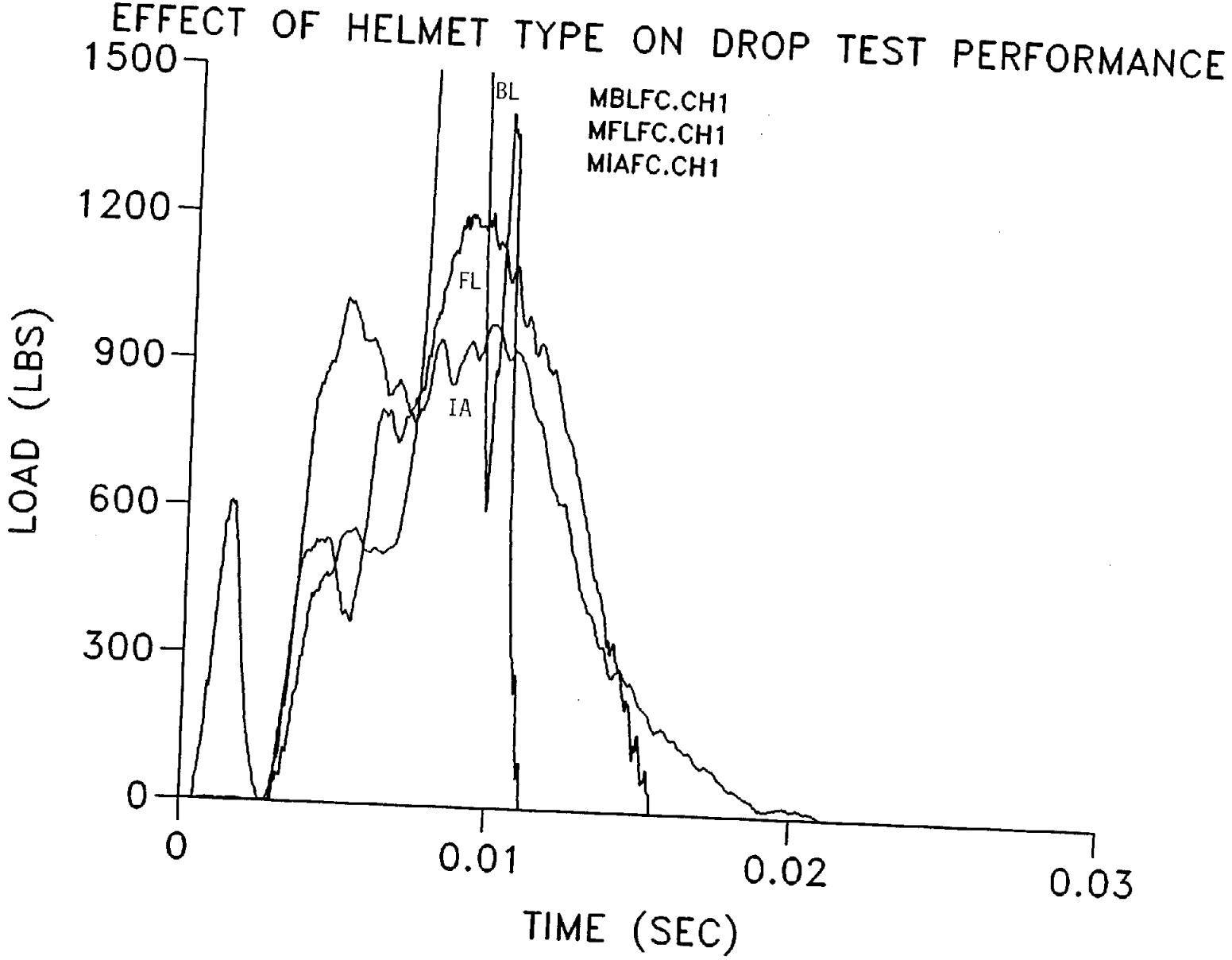

EFFECT OF HELMET TYPE ON DROP TEST PERFORMANCE
ABLFC.CH2
AFLFC.CH2
AIAFC.CH2
BL
FL
IA
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)

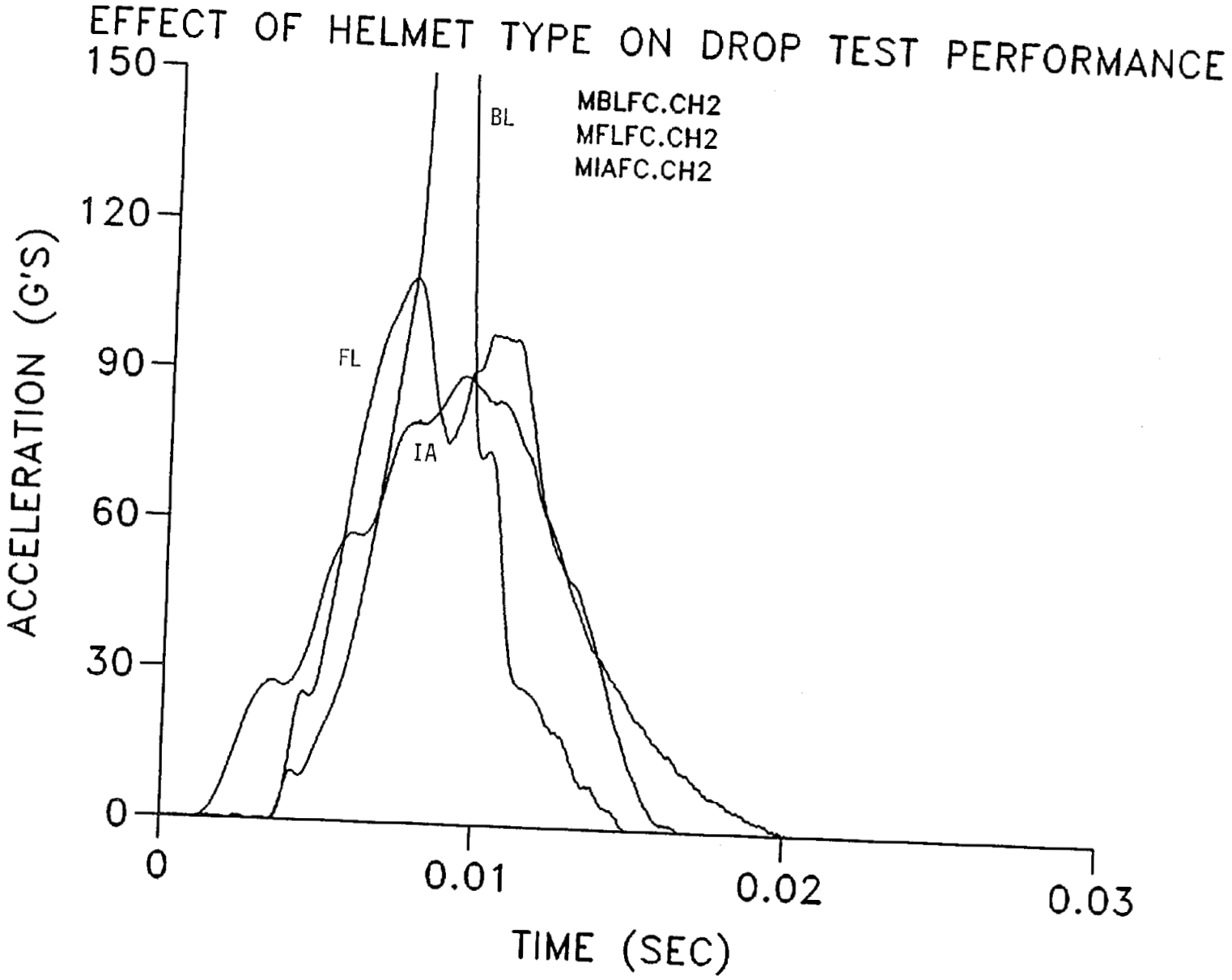

EFFECT OF HELMET TYPE ON DROP TEST PERFORMANCE
EBLFC.CH1
EFLFC.CH1
EIAFC.CH1
BL
FL
IA
LOAD (LBS)
1500
1200
900
600
300
0
0
0.01
0.02
0.03
TIME (SEC)

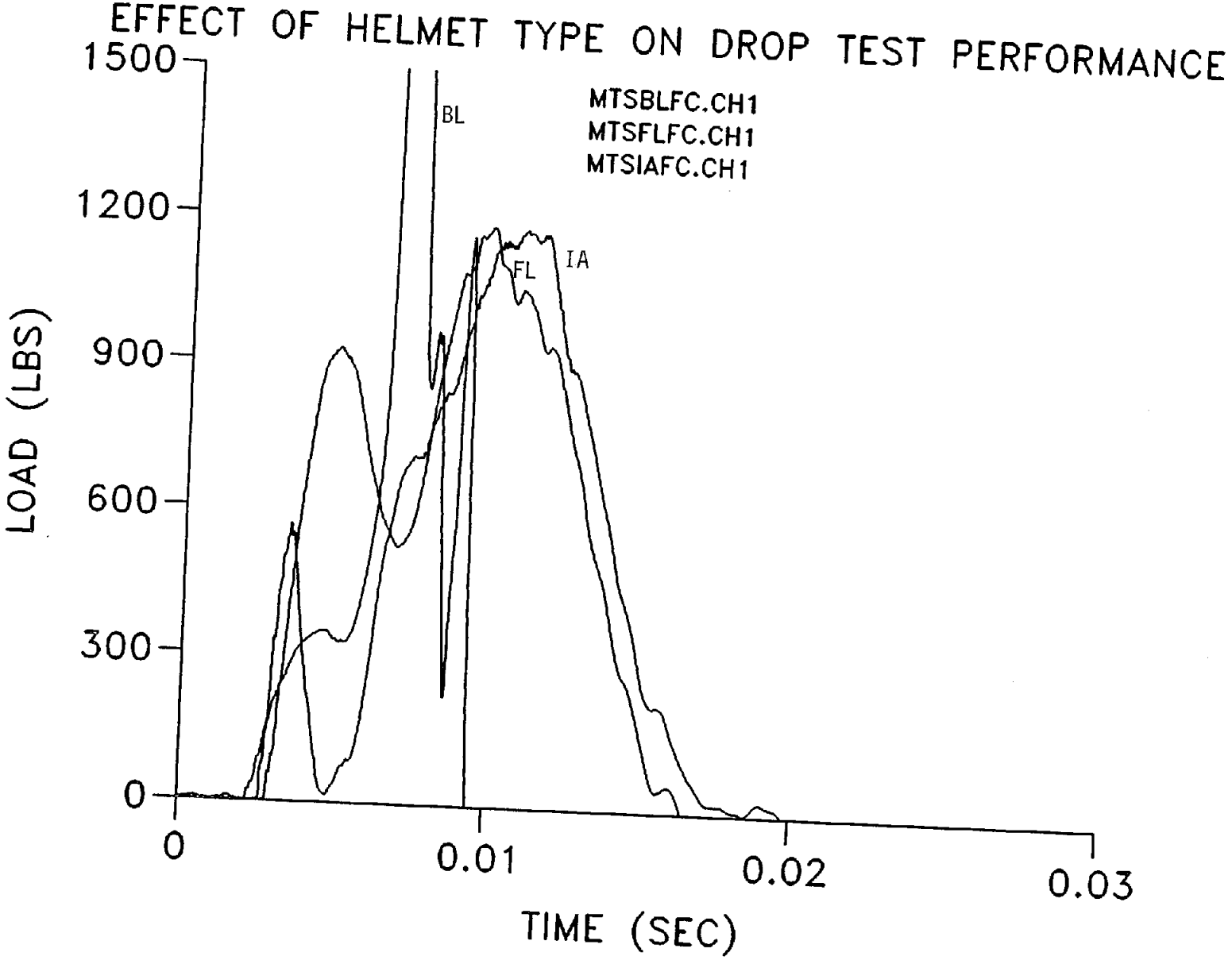

EFFECT OF HELMET TYPE ON DROP TEST PERFORMANCE
150
120
90
60
30
0
ACCELERATION (G'S)
EBLFC.CH2
EFLFC.CH2
EIAFC.CH2
BL
FL
IA
0
0.01
0.02
0.03
TIME (SEC)


EFFECT OF HELMET TYPE ON DROP TEST PERFORMANCE
1500
1200
900
600
300
0
LOAD (LBS)
JBLFC.CH1
JFLFC.CH1
JIAFC.CH1
BL
FL
IA
0
0.01
0.02
0.03
TIME (SEC)


EFFECT OF HELMET TYPE ON DROP TEST PERFORMANCE
JBLFC.CH2
JFLFC.CH2
JIAFC.CH2
BL
FL
IA
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)


EFFECT OF HELMET TYPE ON DROP TEST PERFORMANCE
1500
1200
900
600
300
0
LOAD (LBS)
BL
FL
IA
MBLFC.CH1
MFLFC.CH1
MIAFC.CH1
0
0.01
0.02
0.03
TIME (SEC)


EFFECT OF HELMET TYPE ON DROP TEST PERFORMANCE
MBLFC.CH2
MFLFC.CH2
MIAFC.CH2
BL
FL
IA
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)


EFFECT OF HELMET TYPE ON DROP TEST PERFORMANCE
MTSBLFC.CH1
MTSFLFC.CH1
MTSIAFC.CH1
BL
FL
IA
LOAD (LBS)
1500
1200
900
600
300
0
0
0.01
0.02
0.03
TIME (SEC)

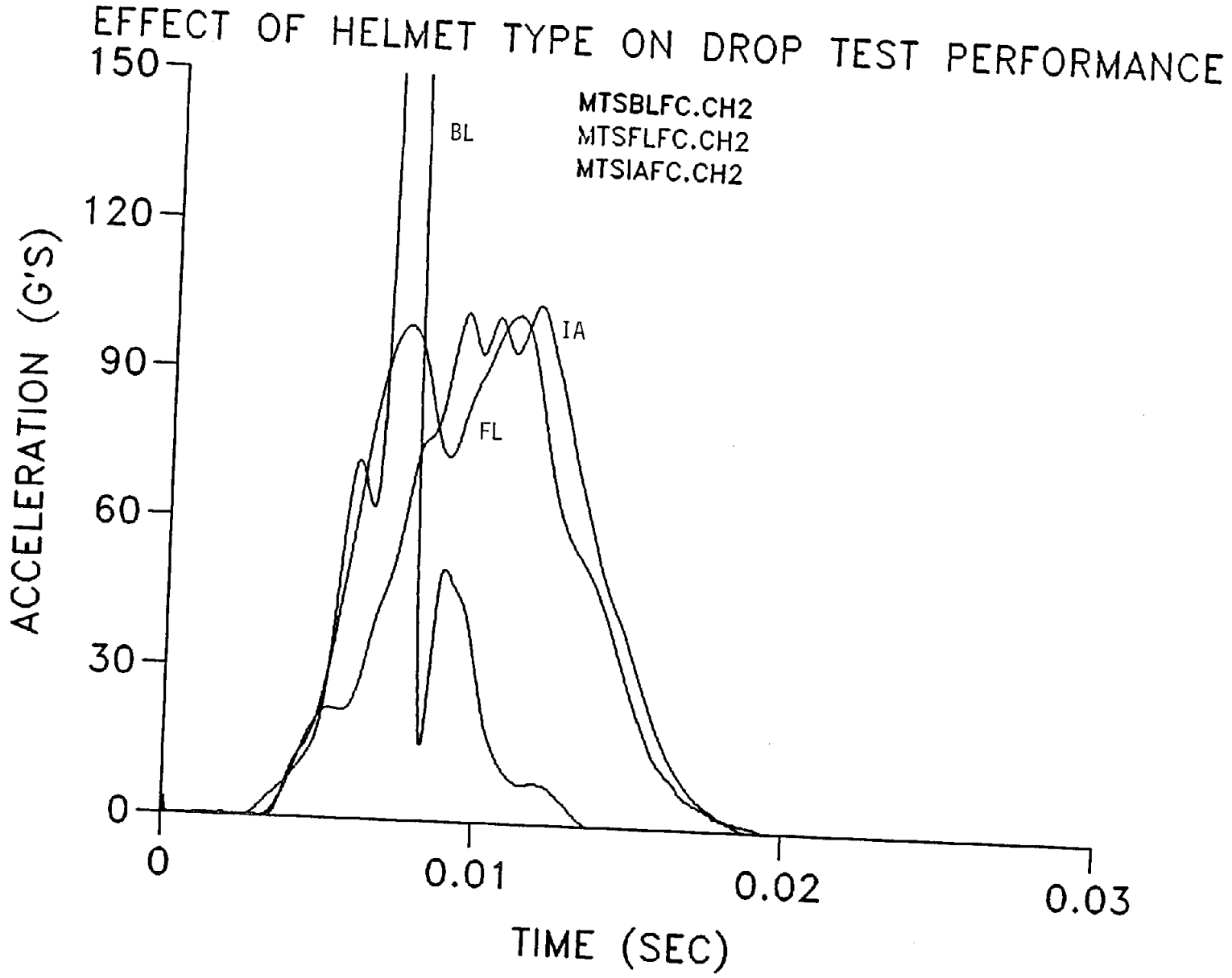
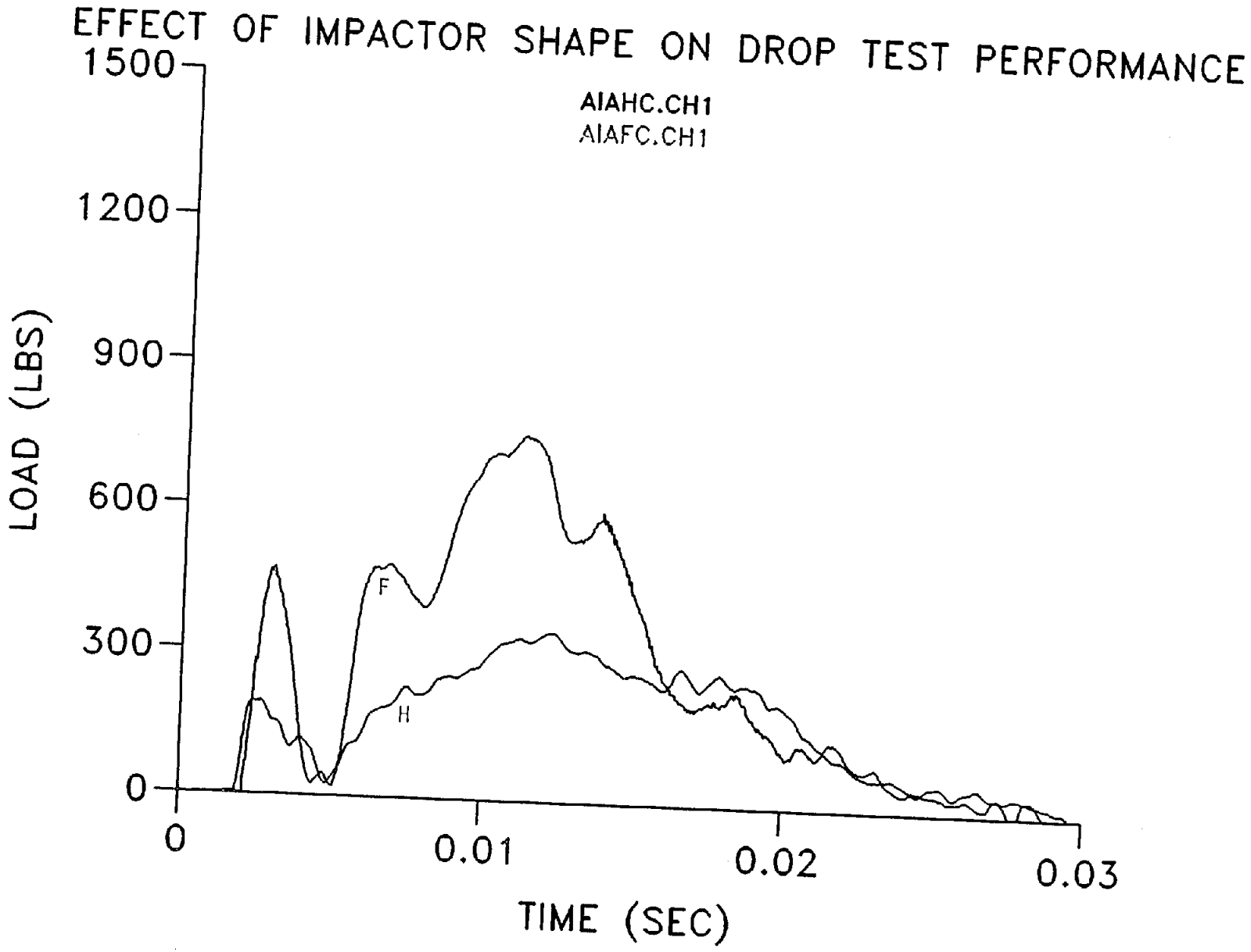
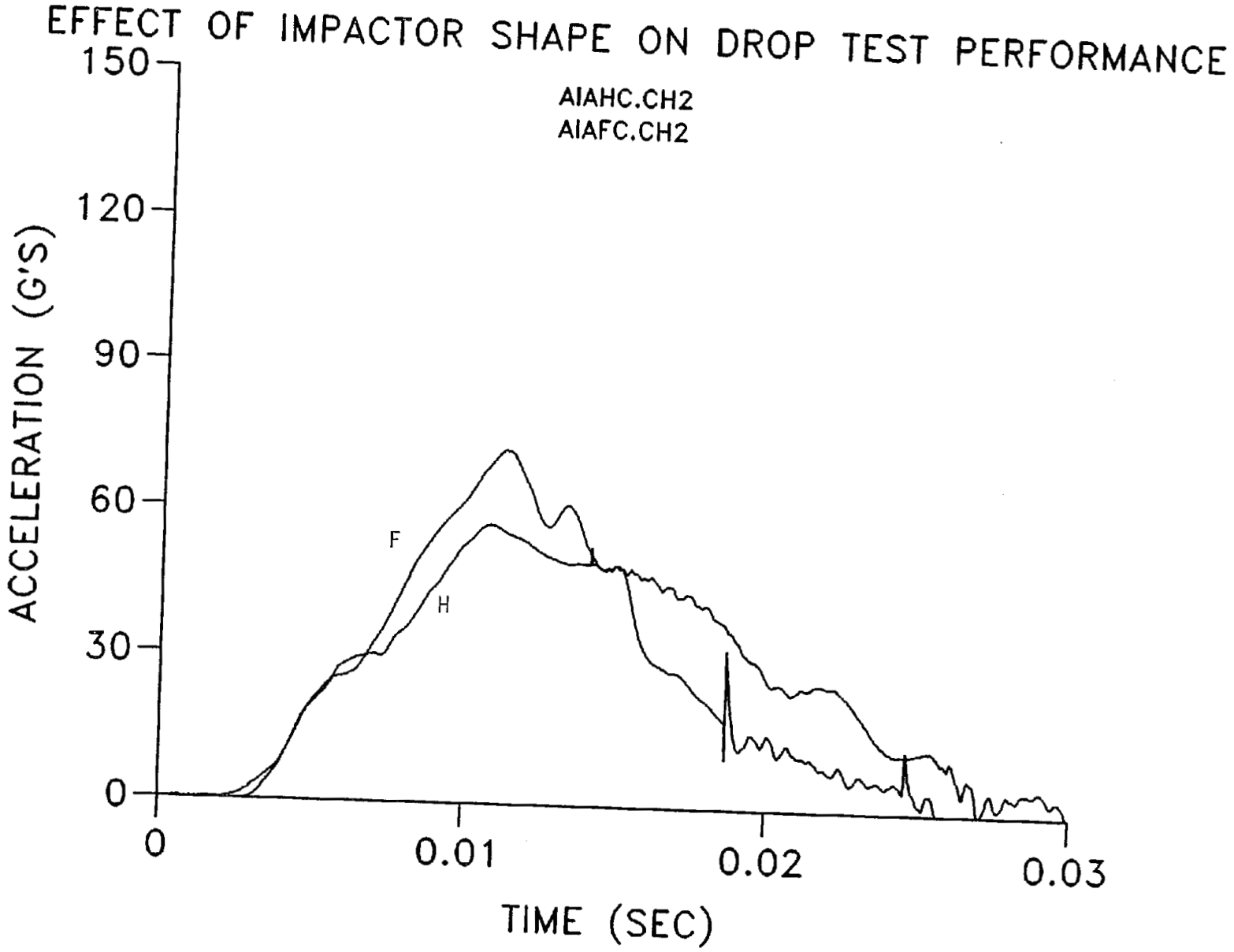
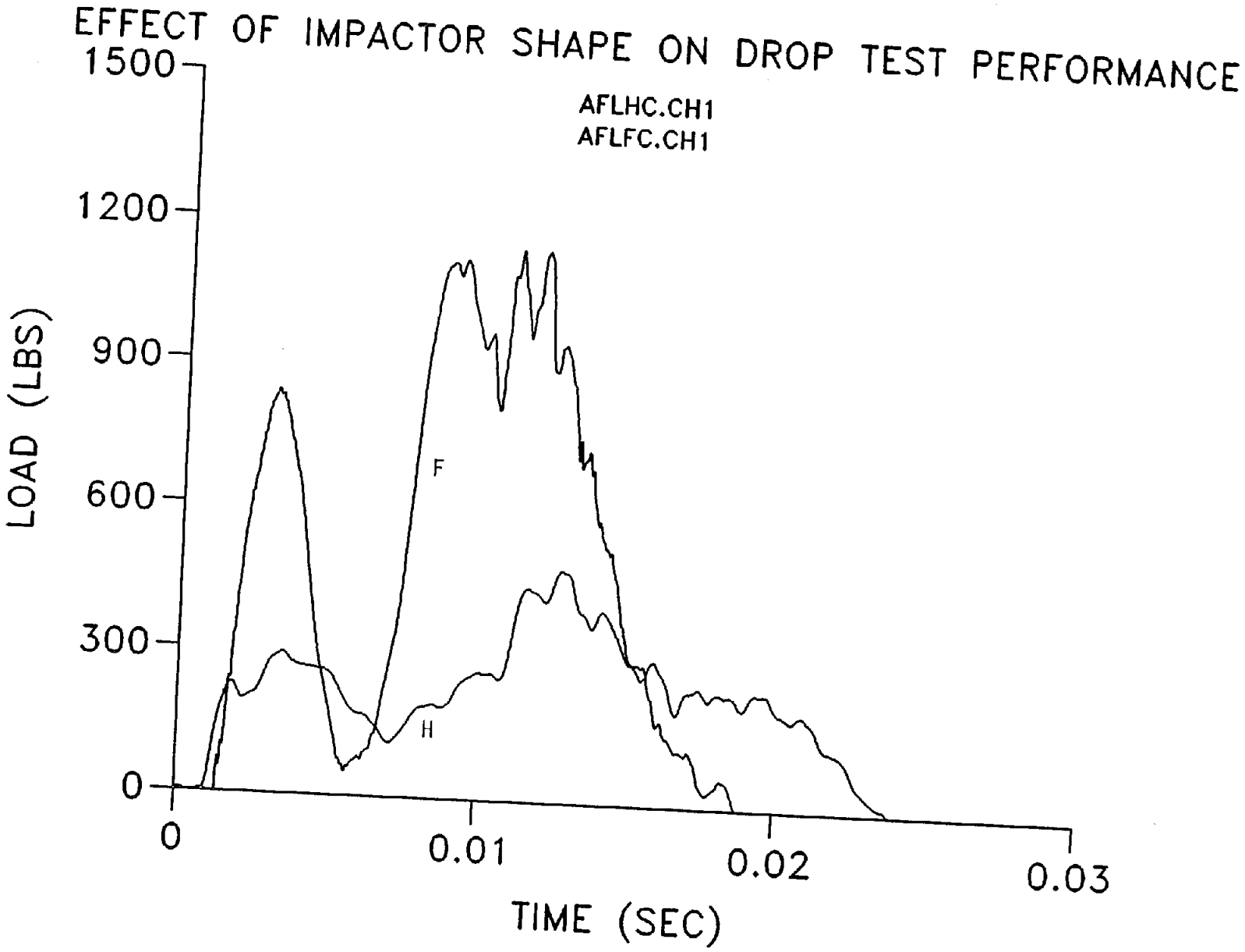

EFFECT OF HELMET TYPE ON DROP TEST PERFORMANCE
MTSBLFC.CH2
MTSFLFC.CH2
MTSIAFC.CH2
BL
IA
FL
ACCELERATION (G'S)
150
120
90
60
30
0
0
0.01
0.02
0.03
TIME (SEC)

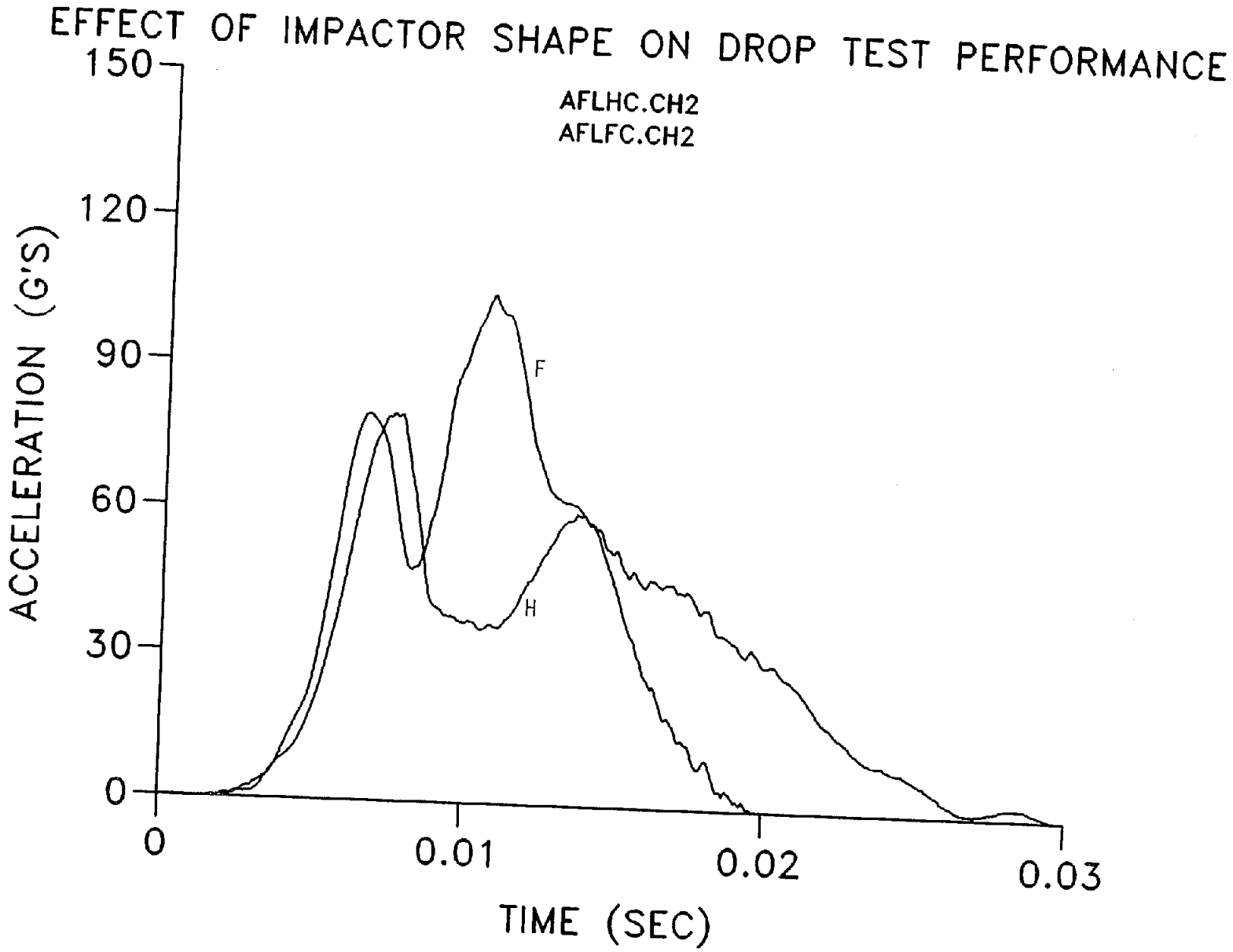
TABLE C.5
EFFECT OF IMPACTOR SHAPE ON DROP TEST PERFORMANCE

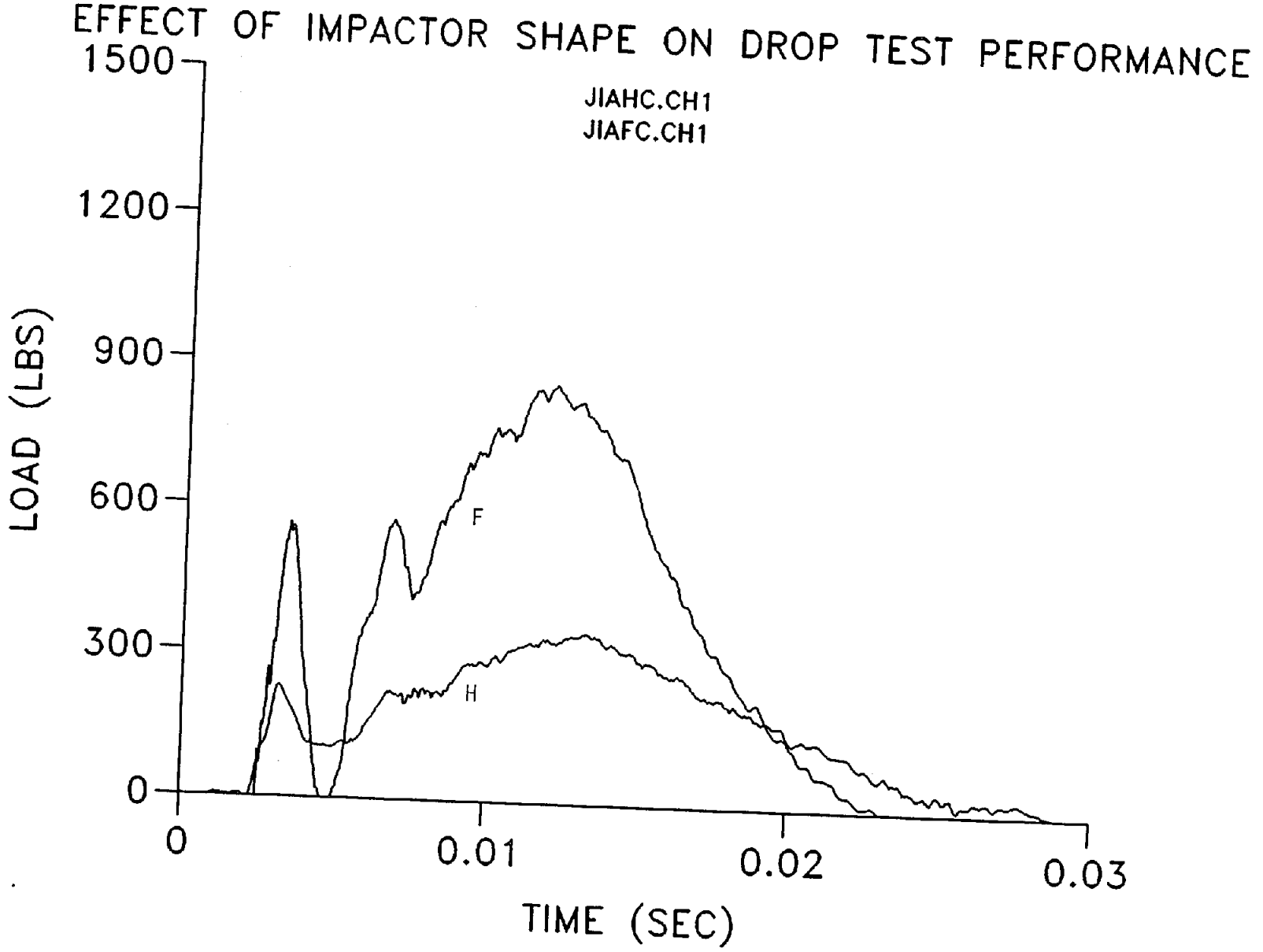
FILE	POINTS	CHANNEL 1 NAME	CAL (LBS/V)	OFFSET (V)	MIN (LBS)	MAX (LBS)	CHANNEL 2 NAME	CAL (G'S/V)	OFFSET (V)	MIN (G'S)	MAX (G'S)
AIAHC	16384	LOAD	258.000	0.020	-27.756	350.174	SI ACCEL	-50.000	0.034	-1.981	57.345
AIAFC	16384	LOAD	258.000	0.020	-40.484	759.468	SI ACCEL	-50.000	0.029	-3.681	72.735
AFLHC	16384	LOAD	258.000	0.015	-45.402	482.440	SI ACCEL	-50.000	0.037	-1.083	79.727
AFLFC	16384	LOAD	258.000	0.020	-27.750	1147.611	SI ACCEL	-50.000	0.040	-4.095	104.792
JIAHC	16384	LOAD	258.000	0.021	-11.661	351.151	SI ACCEL	-50.000	0.034	-0.760	53.195
JIAFC	16384	LOAD	258.000	0.021	-30.671	862.503	SI ACCEL	-50.000	0.020	-3.663	76.171
JFLHC	16384	LOAD	258.000	0.016	-79.603	376.432	SI ACCEL	-50.000	0.022	-1.828	79.471
JFLFC	16384	LOAD	258.000	0.020	-55.627	1173.905	SI ACCEL	-50.000	0.033	-3.008	103.926

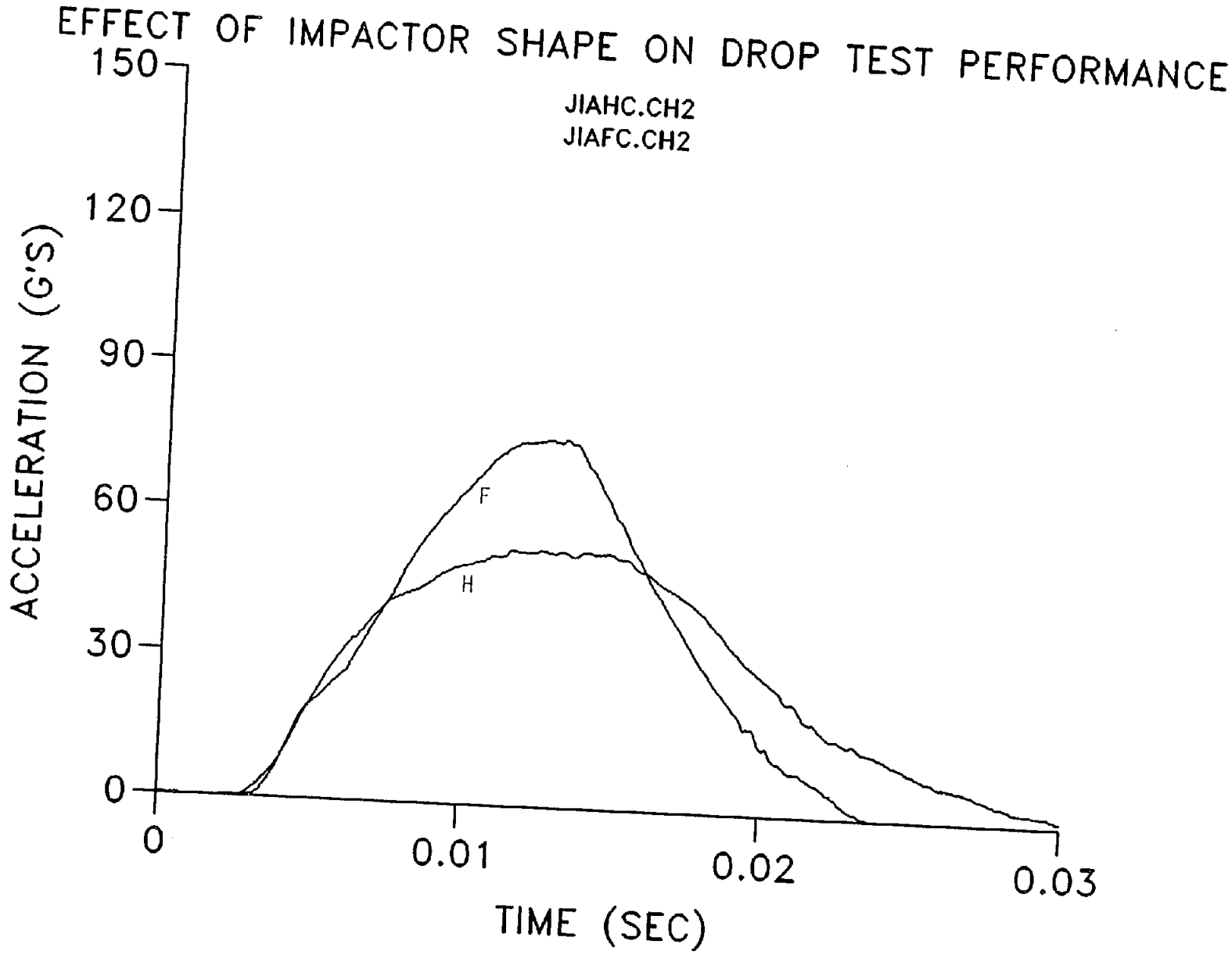

EFFECT OF IMPACTOR SHAPE ON DROP TEST PERFORMANCE
AIAHC.CH1
AIAFC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
F
H
0
0.01
0.02
0.03
TIME (SEC)

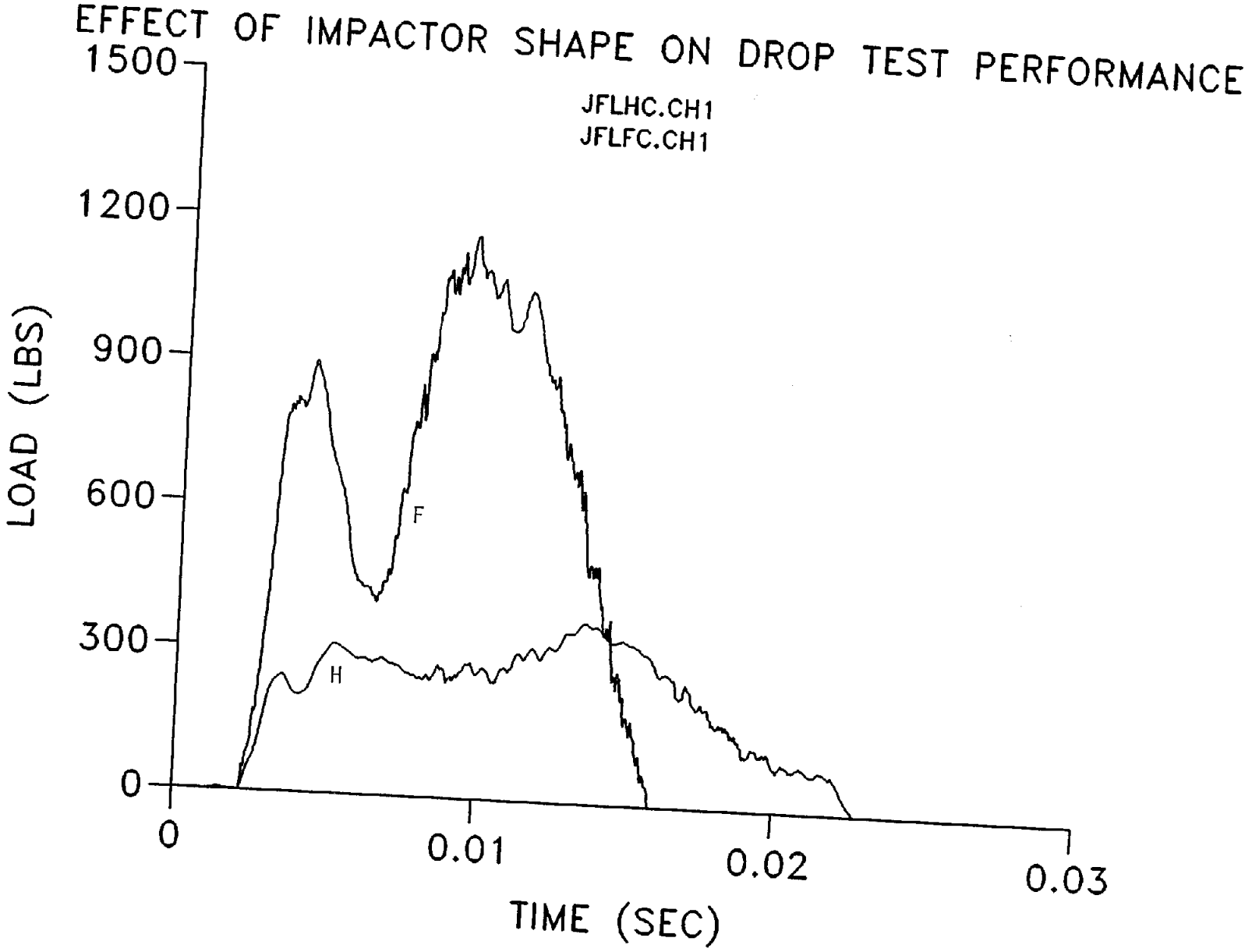

EFFECT OF IMPACTOR SHAPE ON DROP TEST PERFORMANCE
AIAHC.CH2
AIAFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
F
H
0
0.01
0.02
0.03
TIME (SEC)

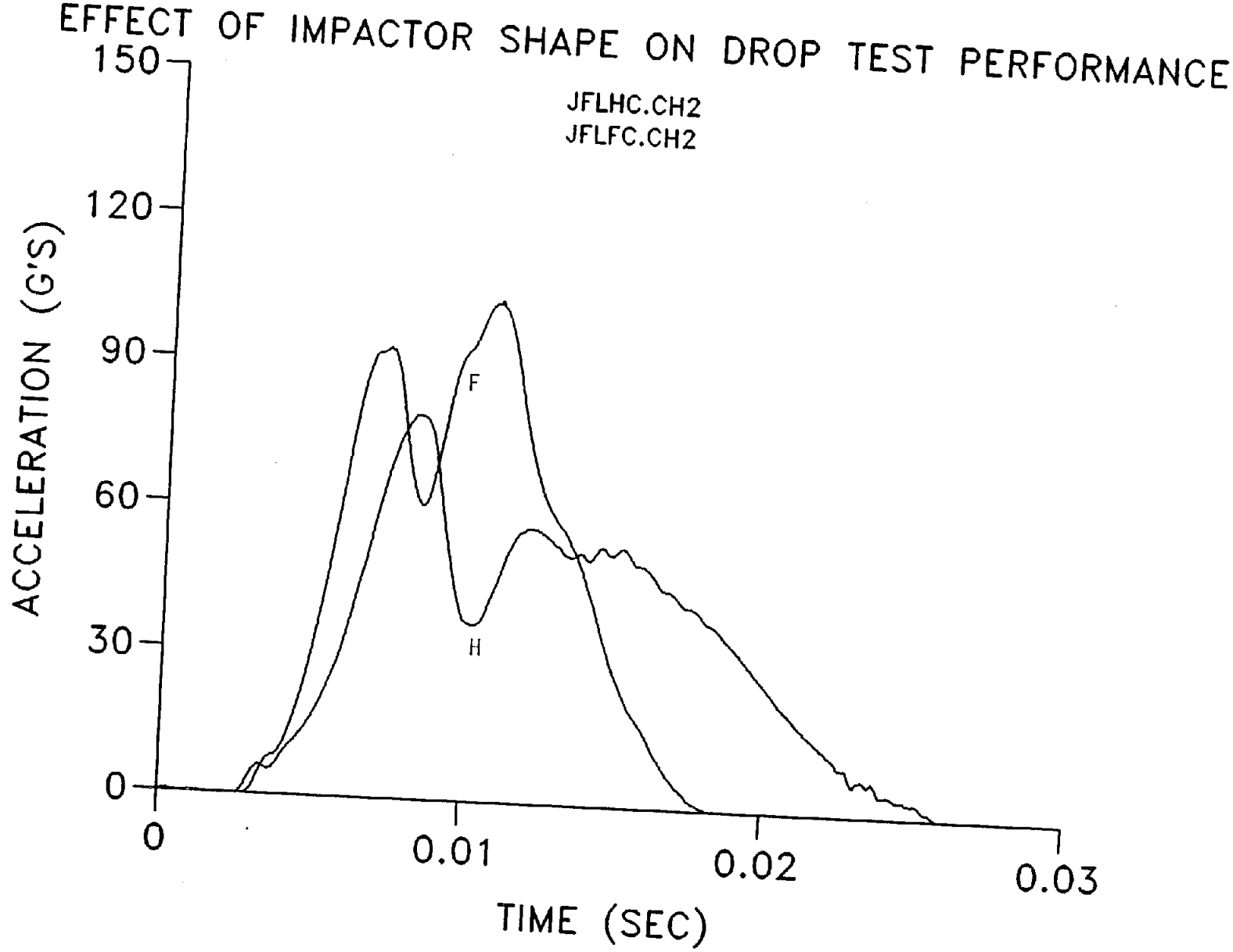

EFFECT OF IMPACTOR SHAPE ON DROP TEST PERFORMANCE
AFLHC.CH1
AFLFC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
F
H
0
0.01
0.02
0.03
TIME (SEC)


EFFECT OF IMPACTOR SHAPE ON DROP TEST PERFORMANCE
AFLHC.CH2
AFLFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
F
H
0
0.01
0.02
0.03
TIME (SEC)


EFFECT OF IMPACTOR SHAPE ON DROP TEST PERFORMANCE
JIAHC.CH1
JIAFC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
F
H
0
0.01
0.02
0.03
TIME (SEC)


EFFECT OF IMPACTOR SHAPE ON DROP TEST PERFORMANCE
JIAHC.CH2
JIAFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
F
H
0
0.01
0.02
0.03
TIME (SEC)


EFFECT OF IMPACTOR SHAPE ON DROP TEST PERFORMANCE
JFLHC.CH1
JFLFC.CH1
1500
1200
900
600
300
0
LOAD (LBS)
F
H
0
0.01
0.02
0.03
TIME (SEC)


EFFECT OF IMPACTOR SHAPE ON DROP TEST PERFORMANCE
JFLHC.CH2
JFLFC.CH2
ACCELERATION (G'S)
150
120
90
60
30
0
F
H
0
0.01
0.02
0.03
TIME (SEC)