NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

UNIVERSITY OF ALBERTA

MICROPROCESSOR BASED REAL-TIME GRAIN LOSS MONITORING AND PREDICTION SYSTEM FOR AN AXIAL-FLOW COMBINE

by

CHENGQI LIU

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

DEPARTMENT OF AGRICULTURAL ENGINEERING

EDMONTON, ALBERTA

FALL 1990

ISBN 0-315-65108-3

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR CHENGQI LIU

TITLE OF THESIS MICROPROCESSOR BASED REAL-TIME GRAIN LOSS MONITORING AND PREDICTION SYSTEM FOR AN AXIAL-FLOW COMBINE

DEGREE FOR WHICH THESIS WAS PRESENTED MASTER OF SCIENCE

YEAR THIS DEGREE GRANTED FALL 1990

(SIGNED) ...Chengqi Liu.....

PERMANENT ADDRESS:

....8406 104 st.......

....Apt. 35...............

....Edmonton, AB.....

DATED ..October 8,...1990

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled MICROPROCESSOR BASED REAL-TIME GRAIN LOSS MONITORING AND PREDICTION SYSTEM FOR AN AXIAL-FLOW COMBINE submitted by CHENGQI LIU in partial fulfilment of the requirements for the degree of MASTER OF SCIENCE.

....
N. Durdle

....
H. P. Harrison

....
J. J. Leonard

Date....05 OCT 90....................

ABSTRACT

The objective of this project was to develop and test a high-performance, microprocessor-based data acquisition system to monitor the grain loss from an axial-flow combine in real time.

To achieve this objective nine acoustic sensors were located underneath the separating grate section of an axial flow harvester to detect the grain impacts.

Signal conditioners were built to distinguish between grain and non-grain impacts and were suitably interfaced to a data acquisition system.

Data acquisition interfaces were built which were connected to the outputs of the sensor signal conditioners. These consisted of counters which could count the grain impacts accurately. Under the control of the software, multiple channel data acquisition could be conducted simultaneously.

A real time regression analysis program was developed in assembly language. Using the least square method, separation distribution curves were obtained in longitudinal and arc directions in real time. Care was taken to minimize the interference of real time calculation with data acquisition, so that the data acquisition and processing could be conducted independently.

Laboratory facilities were built for sensor calibration and simulation of rotor separation distribution.

Threshing experiments were carried out and the actual loss calculated in real time. The actual loss data could be displayed on a LCD (liquid crystal display unit) or printed by a printer. The results were compared with measured losses and confirmed the feasibility of the design of hardware and software.

A linear regression of actual versus predicted losses gave a high correlation (R^2=0.92) and a slope of close to one (1.02). The comparison of calculated to measured losses showed a statistically significant correlation (α=0.05) with the data over all the runs.

ACKNOWLEDGEMENTS

I wish to acknowledge my supervisor Dr Jeremy J. Leonard, Department of Agricultural Engineering University of Alberta, for his advice, encouragement, patience and support for fulfilment of this thesis.

I wish to express special thanks to Dr Page Harrison, Department of Agricultural Engineering, University of Alberta, for the supply of facilities, materials and assistance which enabled the threshing tests to be conducted smoothly.

I wish to extend my gratitude toward Department of Agricultural Engineering staff Mr Sandy Yakimchuk, and Mr Ray Hollowach, former graduate student Mr Anders Bjork and alumni Mr Willem Groeneveld and Mr Gregory Snaith for all their patience and help during hardware-software development and threshing experiments.

I wish to express appreciation to my wife, my mother and brother in China, and to my teachers and colleagues, at Beijing Agricultural Engineering University for their support and understanding.

Funding for this project was provided by Alberta Agriculture.

TABLE OF CONTENTS

Chapter | Page

LIST OF TABLES

LIST OF FIGURES

Figure page

LIST OF PLATES

LIST OF SYMBOLS

Symbol	Description	Units
A	constant	
B	attenuation coefficient of exponential curve	m^{-1}
a, b	edge locations of apertured part of the grate in the grate arc direction	m
A_j, B_j, C_j	coefficients determining shape of curves	
C	width of the rotor apertures part in grate section	m
count1, count2	initial counts loaded in counter registers CR1 and CR2.	
$\bar{E}$	mean of the errors (%)	
e	base of the natural logarithm	
f_p	PCLOCK frequency	MHz
G_0	grain initially passed onto the walkers	kg
GL	actual grain loss number of kernels	
H_0	null hypothesis of Student's 'T' test	
H_1	alternative hypothesis of Student's 'T' test	
i	location number of a sensor in a row	
j	row number of sensor matrix	
K	coefficient of the predicted grain loss	m^{-1}
K_1	reciprocal of the effective sampling area of each sensor	m^{-2}
K_2	a calibration coefficient obtained in threshing calibration test	
L	distance to the end of the straw walkers	m

L_w	straw walker grain loss	kg
1	axial distance from the front of the rotor	m
M	mean of loss ratios (actual/predicted) sampled from the population	
N	number of kernels	
N_a	number of kernels (actual)	
N_t	number of kernels (tested)	
n	sample size	
R^2	regression correlation	
$R_n(b)$	the truncation error in Taylor Series	
S	the sum of the squares of deviations from the curve	
S_p	Point Separation	kg $m^{-2}s^{-1}$
$S_w(L)$	the separation rate occurring at the end of the straw walkers	kg $m^{-2}s^{-1}$
$S_w(X)$	separation rate at a distance from the front of the straw walkers	kg $m^{-2}s^{-1}$
S1-S9	sensor counts stored in memory	
s	standared deviation of samples	
Ts	sampling duration	s
T_1	Timer/Counter1 (TC1) 1 bit counting time	s
W	the weight of kernels in each collection box	g
W_{tk}	weight of thousand kernels	
g		
X	coordinate along walker	m
X_i	location coordinate of the i_{th} sensor around the grate in a same sensor row	m
Y_j	location of the sensor rows	m

y_0	coordinate of the rotor end	m
Z	point separation rate (impact count)	
Z_0	analytical initial separation rate (impact count)	
Z_i	impact count on the i_{th} (i=1,2,3) sensor at axial position j (j=1, 2, 3)	
$\bar{Z}_i$	mean of impact counts on j_{th} sensor row (j=1, 2, 3)	
$\bar{Z}$	mean separation distribution in the axial direction	
α	level of significance	
$\beta_0, \beta_1, \beta_2$	parameters determining shape of the curve	
ϵ_i	curve deviation from the true value	
μ_0	specified population mean	

1. INTRODUCTION

Modern grain harvesters, or combines, are complex, efficient and expensive machines that are widely used in grain farming operations. To cope with changing harvesting conditions, manufacturers and farmers have attempted to achieve optimum performance from their grain harvesters for years.

The performance of a combine is affected by many factors such as grain and straw yield, crop type and variety, cutting width and crop height, moisture content and even local climatic conditions (PAMI 1980). This complexity gives rise to the problem of how to properly adjust a combine to harvest a crop at the maximum rate while maintaining an acceptable grain loss level. Many suggestions have been well documented since the 1960's.

One method used to realize optimum operation is to adjust the ground speed in response to the perceived grain loss level. A grain loss monitor would be able to provide useful information for the combine operator to control the ground speed.

Users reported that the most valuable aspect provided by monitors was that they indicated when grain loss began due to overloading or misadjustment and that sudden changes in loss rate indicated the need for corrective action to be taken (Reed 1977). Without monitors it was virtually impossible for the combine operator to know whether the combine was operating near or above maximum separating capacity. The careful operator also became aware of the need and the benefit derived from adjusting threshing cylinder speed, concave clearance and other parts to improve the separating performance.

1.1 Grain Loss Terminology

Grain losses are classified according to their sources and include all field losses attributable to the machine (ASAE 1990). For clarity, grain loss terminology is reviewed below:

Gathering Loss Rate:

The weight of grain per unit of time which has been missed or dropped by the header or pickup. Gathering losses are expressed as a percentage of the sum of the grain feed rate and gathering loss rate.

Processing Loss Rate:

The weight per unit of time of detached grain and grain in unthreshed heads remaining in the straw and chaff after completion of the threshing, separating and cleaning processes. Processing losses are expressed as a percentage of the grain feed rate.

Leakage Loss:

Any involuntary loss of grain from the combine other than those described above. This is expressed as a percentage of the grain feed rate.

ASAE (1987) define the combine capacity as the maximum sustained total feed rate at which the **processing loss**, with the machine in field operation on level ground, is between one to three percent, depened on the crops and conditions. Apparently, the processing loss is a significant loss of the three kinds of losses.

Three terminologies (MOG feed rate, grain feed rate and total feed rate) mentioned above also are explained below:

MOG Feed Rate:

The weight of material-other-than-grain (MOG) passing through the machine per unit of time (tonnes/hour).

Grain Feed rate:

The weight of grain, including processing loss, passing through the machine per unit of time (tonnes/hour).

Total Feed Rate:

The sum of grain feed rate and material-other-than-grain (MOG) feed rate (tonnes/hour).

1.2 Harvesting Principle

Harvesting processes consist of several steps. Crop threshing and separation distribution are different when different types of harvesters are selected. In order to design a suitable grain loss monitor for these complex machines, harvesting principles and grain separation distributions in the harvesters must be reviewed.

1.2.1 Conventional Combines

Conventional combines (i.e., those using straw walkers) are very popular in Western Canada as well as the other regions in North America. Most grain loss monitors are used to measure grain loss from conventional combines.

1.2.1.1 Conventional Combine Operating Principles

Figure 1.1 illustrates the operation of a typical conventional grain combine. Crop is fed tangentially into a cross-mounted cylinder and concave assembly. Threshing occurs largely by impact of the cylinder bars on the incoming crop, while considerable separation occurs through the open grate concave. Separation of the remaining grain from the straw is accomplished with straw walkers, while a cleaning shoe with chaffer and sieve, is used for scalping and final cleaning.

1.2.1.2 Straw Walker Grain Separation Distribution

The separation distribution along the straw walker was reported as a decaying exponential curve (Huisman, 1983) and the amount of grain separated at a distance X along the straw walker, $S_W(X)$, could be described with the following equation:

$$S_w(X) = -BG_0e^{-BX} \qquad (kg/m.s) \qquad 1.1$$

where:

B = constant related to feed rate, G/MOG, walker design, moisture content and other crop properties.

G_0 = grain initially passed onto the walkers (kg).

e = the base of the natural logarithm.

X = coordinate along walker (m).

The separation distribution curve underneath the straw walker is shown in Figure 1.2. The origin of coordinate X was not at the front of the straw walkers but, actually, at a theoretical point where the separation process became exponential.

The actual grain loss could be obtained by integrating this exponential curve from the end of the walkers to infinity.

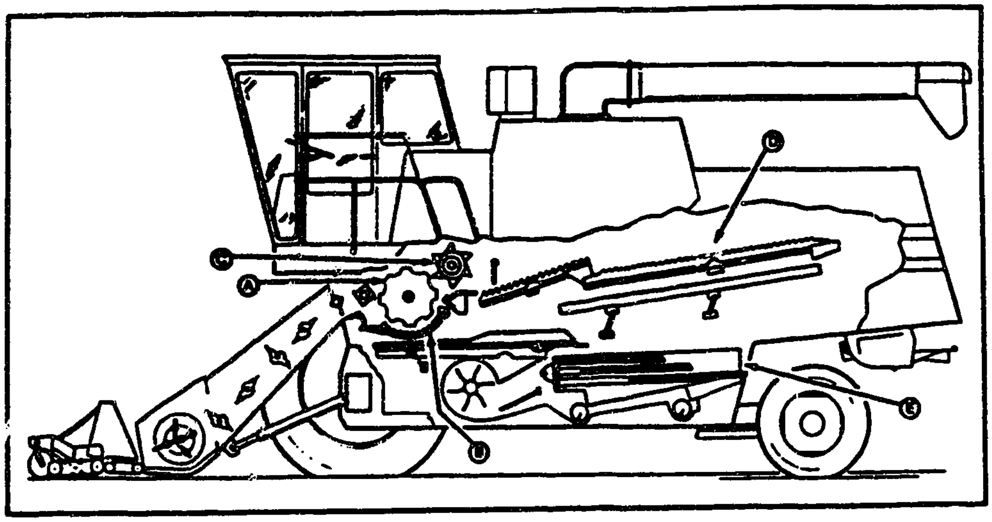


Figure 1.1 A Typical Conventional Combine:
(A) Cylinder, (B) Concave, (C) Beater,
(D) Straw Walkers, (E) Shoe.
(adapted from PAMI 1980)

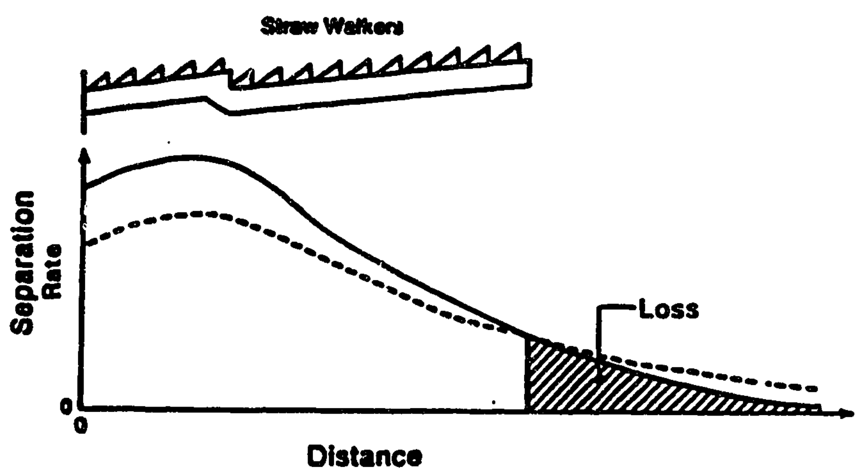


Figure 1.2 The Separation Curve of the Straw Walkers. (Adapted from Huisman, 1983)

1.2.1.3 Conventional Combine Grain Loss

As stated in section 1.1, the processing losses are significant losses. For a conventional combine, processing losses include walker, shoe and cylinder losses. As shown in Figure 1.3, the straw walker loss not only makes up the greatest portion of the total losses, but also increases more rapidly than other losses when the feed rate increases. If walker loss could be measured and controlled, by reading a loss monitor and adjusting the ground speed, the total loss could be controlled at an acceptable level.

1.2.2 Axial Flow Combines

A number of new types of combines, incorporating different threshing and separating concepts, have recently been introduced. The axial flow harvesters have become popular in recent years because they give less threshing damage and lower losses at normal feed rates than conventional combines (PAMI 1978). However, monitoring grain loss for an axial flow type combine is still necessary.

1.2.2.1 Axial Flow Combine Operation Principles

Figure 1.4 shows the operation of the International Harvester 1460, single-rotor, axial-flow combine. Threshing occurs at the front section of the rotor, while the separa-

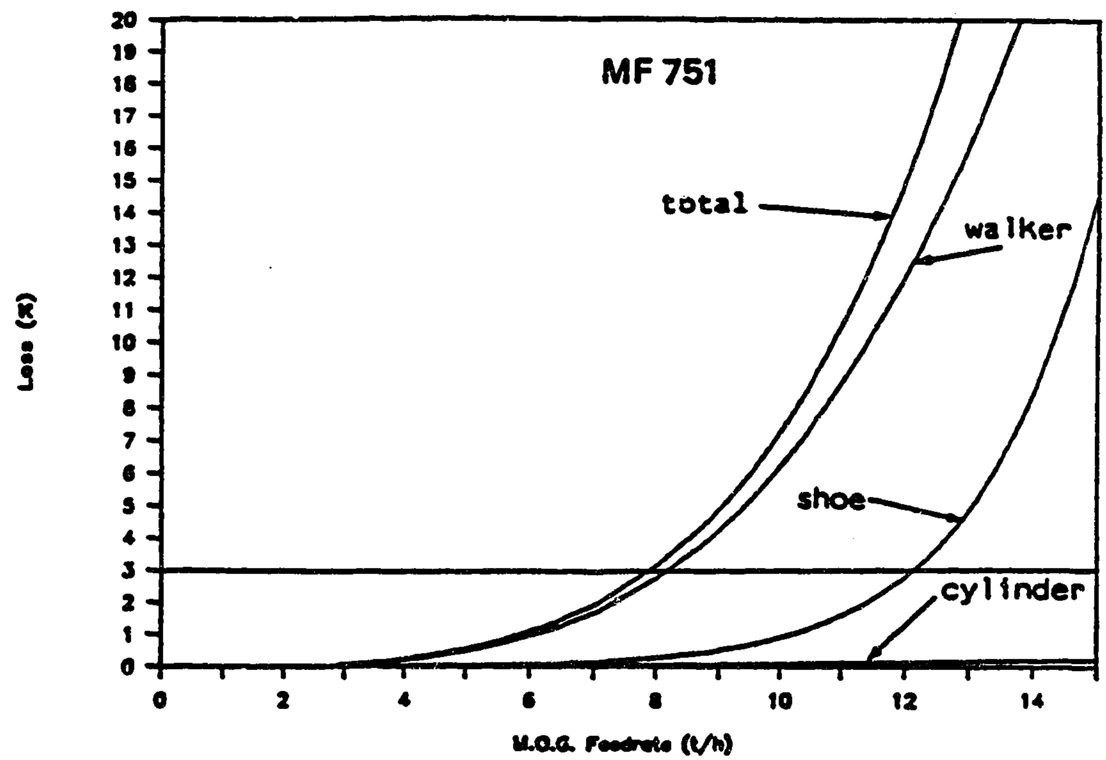


Figure 1.3 Conventional Combine Performance Curves in Barley (adapted from PAMI, 1979, 1980).

tion of the grain from straw is accomplished along the full rotor length in both the threshing and separation concave. A rear beater aids in straw discharge. A conventional cleaning shoe is used for scalping and final cleaning.

1.2.2.2 Grain Separation in Axial Flow Combine

The structure of axial flow combines is different from that of conventional types. Thus, the movement of material inside the rotor and the separation distribution underneath the rotor have their own characteristics. High speed rotation and the circular shape of the rotor causes the separation distribution around the rotor to be non-uniform.

This fact made separation distribution measurement difficult. However, if the distribution curves around the rotor at different points along axial direction could be found, and the mean distribution values at these points could be derived from these curves, the separation distribution along the axial direction would be similar to that underneath straw walkers (decaying exponential curve, equation 1.1). Therefore, the actual grain loss could be predicted by integrating this curve from the rotor end to infinity.

Bjork (1988) suggested the following expression for separation distribution around the threshing concave and separation grate for a single rotor combine (International Harvester 1460).

$$S_p = A + Bx + Cx^2 \qquad 1.2$$

where:

S_p = Point Separation (kg $m^{-2}s^{-1}$),

A, B, C = coefficients determining shape of curve, and

X = coordinate around concave or grate arc (degrees).

Depending on the signs of coefficients B and C, this function will give an initial peak separation that decays in either a concave or convex manner and a peak separation occurring somewhere in the mid-region of the concave or grate.

1.2.2.3 Axial Flow Combine Grain Loss

Grain losses from an axial flow harvester are of two main types, unthreshed grain still in the head (threshing loss) and threshed grain or seed which is discharged with the straw and chaff (separation and cleaning loss). In field tests (PAMI 1978), grain losses for the International Harvester 1460 were low in wheat crops, but became significant in barley crops at high feed rates. Figure 1.5 shows

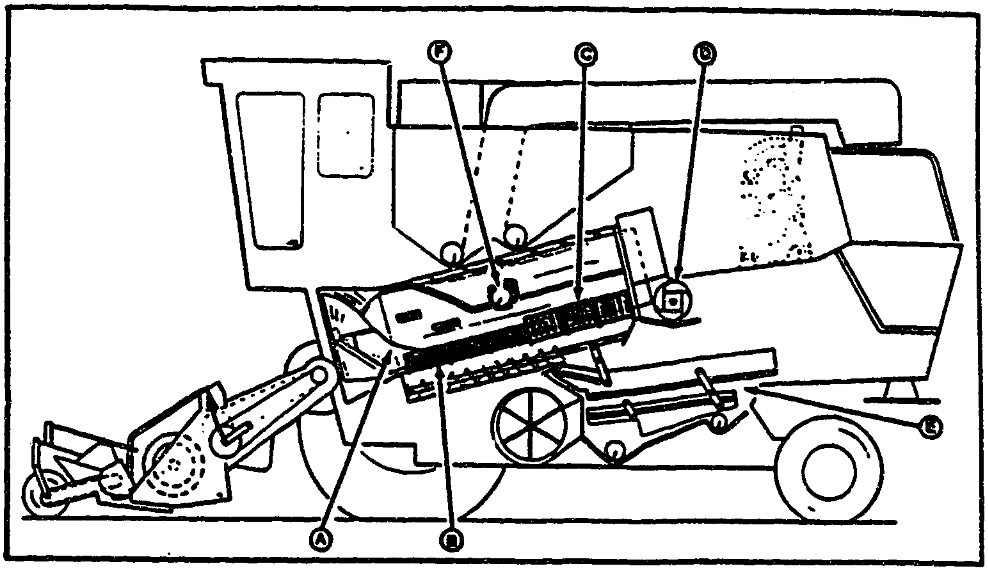


Figure 1.4 International Harvester 1460. (A) rotor, (B) threshing concaves, (C) separation concaves, (D) back beater, (E) shoe, (F) tailings return. (adapted from PAMI, 1980)

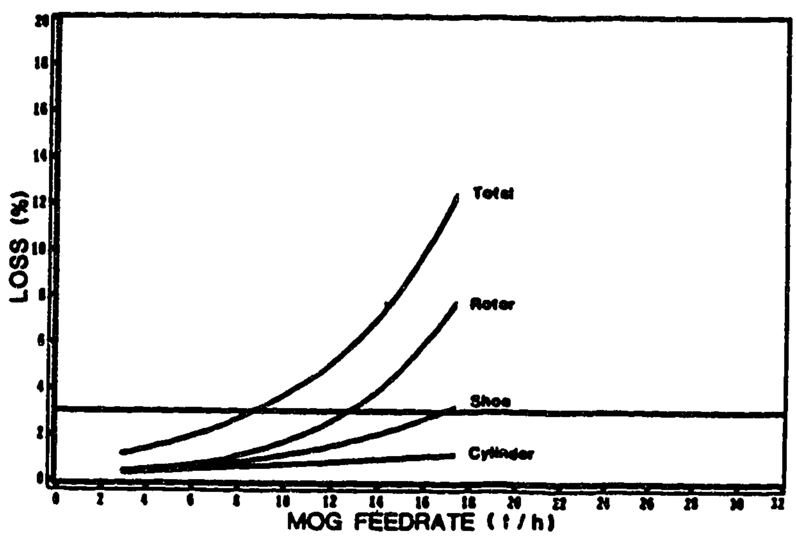


Figure 1.5 International Harvester 1460 Grain loss in Bonanza Barley (adapted from PAMI 1978).

the grain loss from an International Harvester 1460 combine in Bonanza Barley. The rotor (separation) loss is the most significant of all losses.

As with conventional combines, loss monitors on axial flow machines do not give an indication of actual grain loss. However, since the grain separation distribution in these machines could be measured, it seemed reasonable to postulate that an actual grain loss monitoring system could be built. This would be similar in concept to that built by Larson (1987) for conventional combines but would need to be more powerful in order to handle the two-dimensional variation in separation rate.

2. LITERATURE REVIEW

Grain loss monitors are electronic instruments designed to indicate combine grain loss. Sensors, sensitive to seed impacts, are mounted behind the straw walkers, the cleaning shoe and, sometimes, a third sensor is placed at the end of chaffer (Cox, 1982). The signal from these sensors is displayed on a meter at the operator's station. Most loss monitors are not designed to measure the actual amount of grain loss but only to warn the operator of increases or decreases in grain loss rate (PAMI 1978). Figure 2.1 shows a typical grain loss monitor system installed in a conventional combine with straw walkers and cleaning shoe.

An electronic grain loss monitoring device was developed by Feiffer et al. of the German Democratic Republic in 1965 (Reed, 1977). This device removed the chaff from the sample mixture of chaff and lost grain by an air blast before detecting the grain kernels with an acoustic sensor (Reed, 1977).

A device or monitor for detection of grain loss that was believed to be the first grain loss monitor marketed commercially and used by combine operators was developed by Reed et al. (1965) and manufactured by Smith - Roles Limited

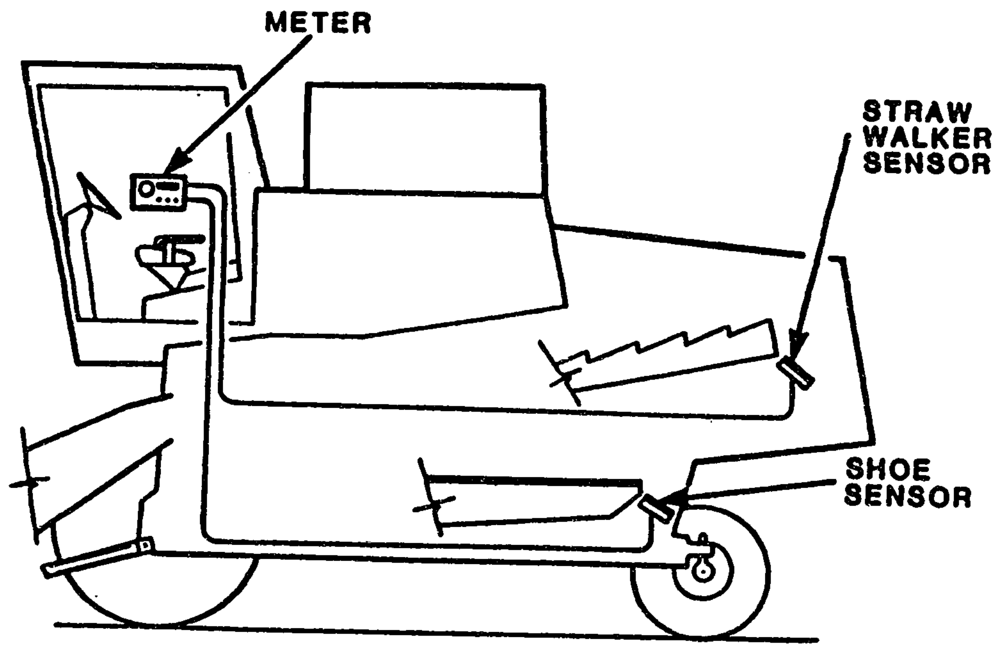


Figure 2.1 Typical Grain Loss Monitor (adapted from PAMI 1978)

of Saskatoon, Saskatchewan, Canada (1968). The monitor was able to determine the grain and non-grain impacts on acoustic sensors (Reed, 1977).

One or two sensor systems could display the decrease or increase of grain loss for conventional or axial-flow combines, but accurate determination of actual grain loss (kg/hectare) could not be obtained.

Huisman (1983) proposed a method of measuring walker loss by measuring the separation rates along the entire walker length (see Figure 1.2). Walker loss then could be obtained by integrating the distribution curve from the walker end to infinity.

Lunty (1986) used a laboratory straw walker assembly and associated data acquisition system to verify Huisman's suggestion.

Sensors were installed at four different positions of the underside of open-bottomed straw walkers to measure grain impacts. The locations of the sensors were selected at the rear section of the straw walkers (for the first sensor X = 0.66m for the fourth one X = 2.76m). By using the data of each run, the natural logarithms of the sensed separation rates were regressed linearly with the position of the sensors solving for A and B in a separation equation of the following form:

$$S_w(X) = Ae^{-BX} \qquad for \qquad X > 0.66m \qquad 2.1$$

where:

$S_w(X)$ = separation rate ($kg/m^2.s$) at a distance X (m) from the front of the straw walkers.

A, B = equation parameters.

The actual walker loss could be obtained by integrating the separation curve from the end of the walkers to infinity (see Figure 1.2):

$$L_w = \int_L^\infty S_w(x)dx \qquad 2.2$$

$$= \int_L^\infty Ae^{-Bx}dx$$

$$= \frac{A}{B}e^{-BL}$$

$$L_w = S_w(L)/B \qquad 2.3$$

where:

$S_w(L)$ = the rate of separation occurring at the end of the straw walkers ($kg/m^2.s$), and

L = distance to the end of the straw walkers (m).

The separation data from the laboratory apparatus showed that actual loss could be predicted from separation measurements at four points below the walkers.

On the basis of Huisman's suggestion and the results of Lunty's attempt, Larson (1988) developed an 8-bit (MC6800) microprocessor-based system capable of rapid data acquisition under the harsh conditions of harvest environment. This system was tested in both laboratory and field conditions and was able to provide separation data for the calculation of walker loss. Four sensors were located at 0.686m, 1.373m, 2.059m and 2.745m from the front end of the walkers. Bandpass filters were implemented to reject non-grain impacts, mechanical and electrical noise. A monostable trigger (555 timer module) was used for obtaining single pulses for a single kernel.

The field test measurements of separation curves were performed successfully. The curves of all valid runs were of high-correlation exponential decays with calculated losses statistically close to the actual loss rate. The worst fit had an R^2 of 0.72, while the best was 0.99.

Loss calculations based on separation rate curves were accurate, provided the samples are point-samples of known area and location. The comparison of calculated to measured loss showed a statistically significant correlation at the level of $\alpha = 0.10$ with limited data. The calculation of walker loss was simple once the exponentially decaying curve was determined (Larson, 1987).

Due to the limitations of the microprocessor, loss calibration could not be done in real time. Larson (1987) recommended that software could be developed to calculate the actual loss from the sensor data in real time. To do this, a more powerful 16-bit microprocessor would be needed.

Since the velocity of straw and grain discharged from the end of the combine was high, PAMI (1978) suggested that loss monitors were not suitably designed for use on the new axial-flow threshing combines. When hit by grain and straw at high velocity, most loss monitors could not distinguish grain from straw and they read in error. In 1983 the British company RDS Farm Electronic Ltd., developed the MK81 combine monitor. The MK81 was designed especially for use on axial-flow combines, but could also be used on traditional types of machines (The Green Book, 1983).

Wang (1984) studied the separation characteristics of a Sperry New Holland TR85 rotary combine. The TR85 uses two longitudinally-mounted, axial threshing and separation rotors. Discussions were based on the separation distribution of the left side rotor-concave assembly. Ten acoustic sensing pads were used as grain impact transducers to study the separation characteristics in the longitudinal direction and, also, the lateral direction around the rotor. The sensors were mounted on a metal frame underneath the threshing and separation concaves. Grain impact signals were recorded on a tape recorder. The results showed that the largest proportion of the separation took place at the threshing concaves. The peak separation rate occurred near the entrance of the thresher and at the right side of the thresher when wheat was combined. By contrast, the peak separation rate occurred at the left side of the thresher and was a little further from the entrance when barley was combined (Wang, 1984).

Bjork (1988) suggested a separation distribution model (equation 1.2) in an axial-flow combine arc direction and this model would give an initial peak separation rate occurring somewhere in the mid-region of the concave or grate followed by exponential decaying in axial direction.

The existence of exponentially decaying separation curves in axial flow machines indicated that the method of Huisman (1983) could be used to determine actual grain loss from these machines. As Larson (1987) suggested a 16-bit microcomputer would be required to do this in real time.

3. OBJECTIVES

The object of this project was to develop and test a 16-bit (Intel 8088), microprocessor-based, nine-sensor grain loss monitor system for an axial flow combine (International Harvester 1460). As a multi-channel data acquisition and processing system, this monitor would be able to distinguish between grain and non-grain impacts on the sensors, count the seed impacts, fit separation curves to the data around the separation arc and in the axial direction, and predict the actual grain loss in real time.

4. SOUNDING BOARD SIGNAL ANALYSIS AND SIGNAL CONDITIONING

Acoustic grain impact sensors were available from several manufacturers. Two kinds of sensors made by different companies were chosen for a comparison experiment and the Baker Electronic Enterprises (Edmonton, AB) product was selected as the key sensing component on the basis of its even sensitivity at different points (except the side edge) of the sensing surface.

The sensor was a rectangular plastic box measuring approximately $160 \times 60 \times 20 mm^3$ composed of two parts: an acoustic plate and a base. Between the acoustic plate and the base, there was a rectangular plastic foam buffer to reduce the duration of decaying grain impact signals, so that the maximum impact frequency could be increased. In each sensor, a circular membrane piezoelectric sensor (R=25.4mm) was attached to the acoustic plate. The whole inside surface of the acoustic plate (sounding board) was glued to the buffer foam so that sounding board response to grain impacts at different points tended to be identical. The other side of the buffer foam was also fixed to the plastic base which was fitted with mounting screws for attachment to the combine. A second pair of screws on the base served as output terminals, which would eventually be connected to sensor signal conditioners. Plate 4.1 shows a photograph of a typical sensor and its inside structure.

4.1 Sounding Board Signal Analysis

Figure 4.1 shows the frequency spectrum from the impacts of rape and its straw and chaff (Baker, 1988). The power spectrum shows that rape kernel impacts caused a peak value near 15KHz. Also, two peak values were produced by the straw and chaff, one close to 15KHz and the other close to 16KHz, but their amplitudes were much lower than that caused by rape kernels. In the low frequency band (0-5KHz), there was another peak point generated by straw and chaff and its amplitude was higher than the peak produced by kernels.

The rape frequency spectrum illustrated that there were two choices to identify grain and non-grain impacts using filtering techniques. In the low frequency band (0-5KHz), a band reject filter may be used to reject the non-grain impact signals and leave the grain impact signals for further use. In the high frequency band, a band pass filter may be designed to amplify grain impact signals and reject non-grain signals. The band pass filter central frequency should equal the frequency of the peak amplitude. The band-

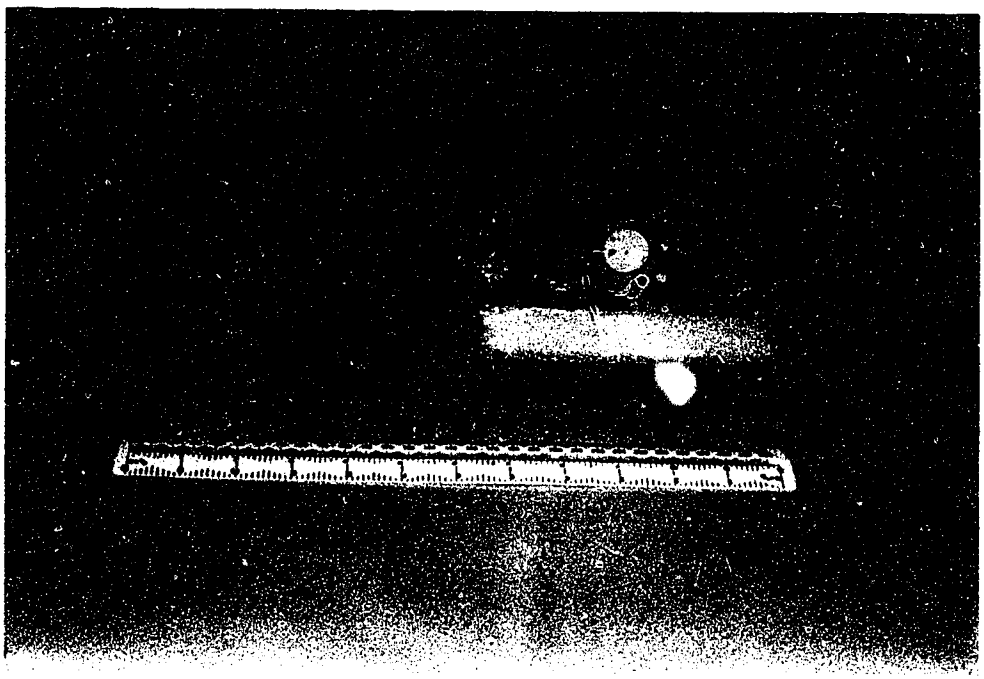


Plate 4.1 A Grain Impact Sensor (left) and
the Sensor inside Structure (right).

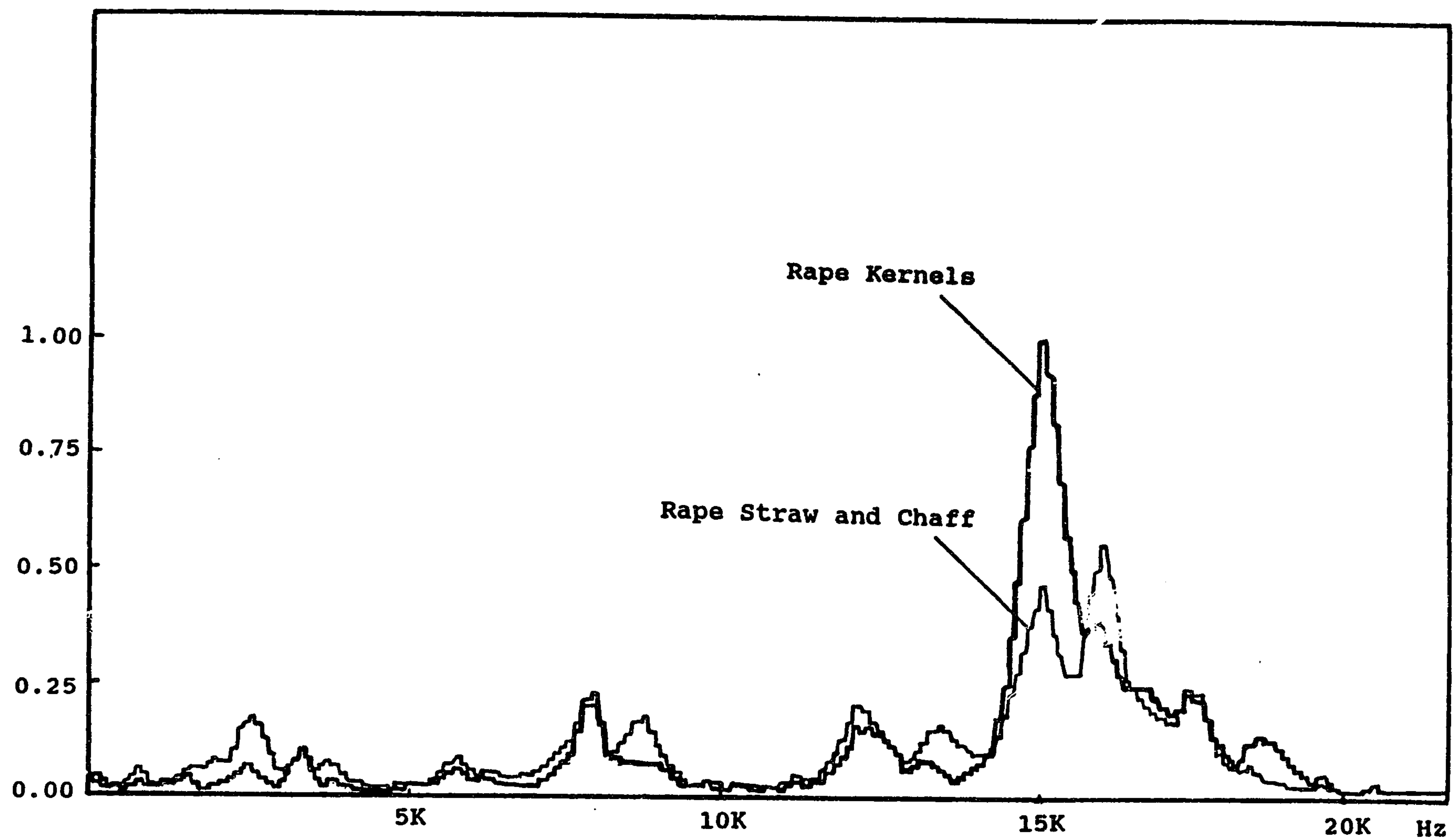


Figure 4.1 The Rape Kernel, Straw and Chaff Signal

FFT Frequency Spectrum (adapted from Baker,1988).

pass filter method was preferred because of the stronger signal and obvious amplitude difference between grain and non-grain signals.

Grain impact signals were enhanced and non-grain signals were attenuated by using band-pass filters. Thus, grain impact signals were conditioned to have higher amplitudes and non-grain signals to have lower amplitudes. Thus, the signals were much easier to process using a comparator. The reference voltage of the comparator was so selected that it was lower than the amplitude of the initial part of grain impact signals and higher than all amplitudes caused by non-grain impacts. Therefore non-grain impact signals were rejected and grain impact signals were selected by changing a comparator's output from one extreme value to the other.

Similarly, barley and wheat kernel impacts also have their characteristic peak frequencies.

Preliminary experiments to find out the central frequencies for wheat and barley kernels were made. The signals from the sounding board were amplified by a simple operational amplifier circuit with a gain of 33. The output signals from the circuit were measured with a storage oscilloscope (Tektronix 464). The wheat kernels were quite dry (10% w.b.) but the barley kernels had a moisture content of

13% w.b. Kernels were dropped from different heights to measure the signals. To obtain the best sensitivity, the orientation of the sounding board was 45° to the horizontal.

The waveform decay envelope curves were recorded by hand (see Figure 4.2). The time for the signal amplitude to decay to 2V was almost independent of the height from which kernels were dropped.

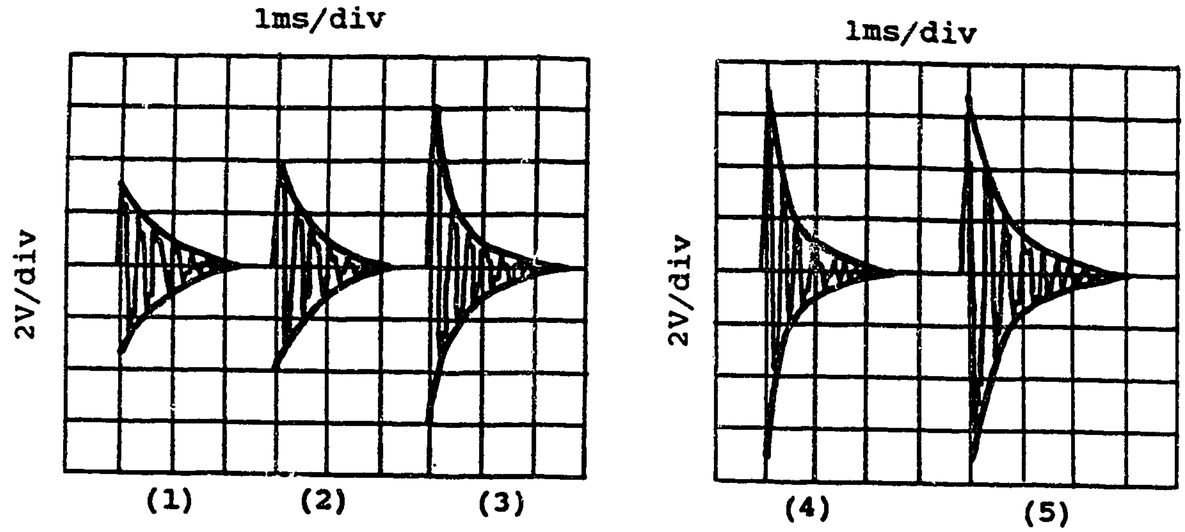
The barley kernels were dropped from 5, 10, and 30cm above the sounding board surface center, the signal decay time kept within 3-5ms (see Figure 4.2 Test 1 Barley) while the frequency of the impact signal was around 3500-5000Hz.

The wheat kernels were dropped from heights of 5, 10, 15, 20, 30, and 60cm and the signal amplitude decayed to 2V within 2-3ms (see Figure 4.2 Test 2 wheat) with a central frequency of about 3000Hz.

With both crops the frequency of signals from straw and chaff impacts was lower (1500-2000Hz) than that from kernel impacts and, thus, filters could be used for impact discrimination.

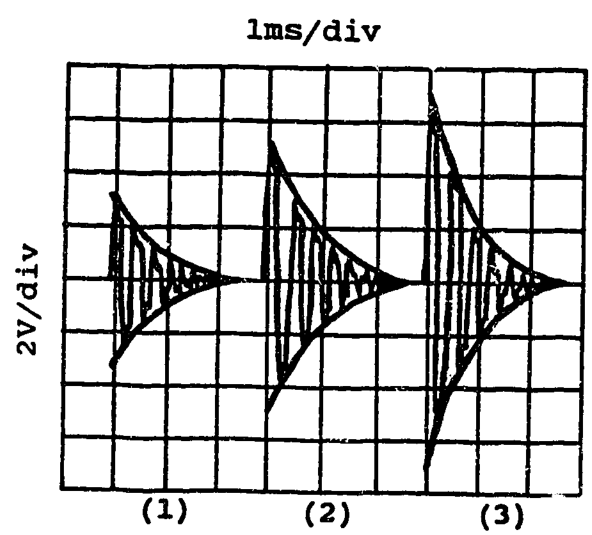
4.2 Signal Conditioning

The sensors and their associated signal conditioners were the key parts of the measurement system since the whole system performance depended on the quality of their signals.



Dorp Height: (1) 5-10cm (2) 15cm (3) 20cm (4) 30cm (5) 60cm

Test 1: Wheat



Dorp Height: (1) 5cm (2) 10cm (3) 30cm

Test 2: Barley

Figure 4.2 The Waveform Envelop Curves of Grain Impacts.

Hence, high quality components (operational amplifiers (op amps), transistors, triggers, resistors etc.) were selected for the circuit design. The signal conditioning circuit is shown in Figure 4.3.

In order to reduce noise interference, the feed wire for weak signals from the sensor outputs to signal conditioner inputs should be as short as possible and the signal conditioner output signals should be as strong as possible. This would increase signal-to-noise ratio when signals were fed along long cables.

The conditioners were designed as independent printed circuit boards attached to sensors so that the weak signal feed wires were only 5cm long. Noise transmitted by electromagnetic induction had few chances to be received by the short wires. Output signals, which consisted of a train of pulses, could be transmitted to a computer along 5 to 8m-long, plastic-insulated wires without significant distortion. The output pulses were able to drive photo-coupled isolators in a computer interface.

A schematic diagram of the signal conditioning circuit is shown in Figure 4.3. Each circuit included an amplifier, a band pass filter, a comparator, Schmitt-triggers, a monostable multivibrator and a voltage follower. Most of the digital integrated circuits (ICs) were CMOS components

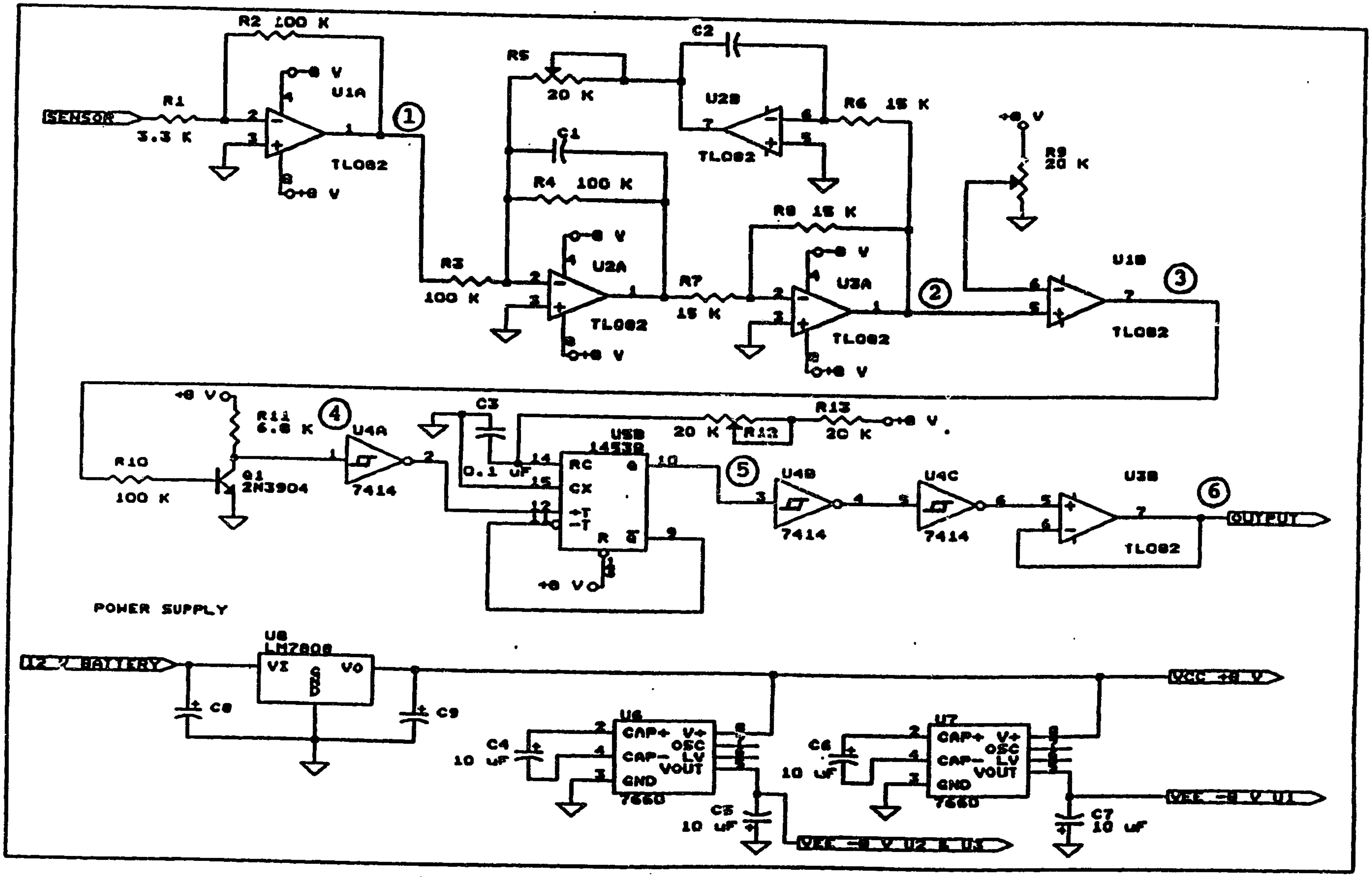


Figure 4.3 Signal Conditioner Circuit.

which were chosen for their flexible power supply requirement and higher level output voltage. A power regulation circuit converted 12V DC battery power to $\pm 8V\,DC$ as a power supply for op amps and other ICs. Thus, the digital CMOS ICs could provide output pulse signals as high as +8V.

The structure and the function of the key parts of the signal conditioning circuits are as follows:

4.2.1. Bandpass Filter

Biquad bandpass filters were used (Johnson et al. 1980). The selection of this bandpass filter was due to its popular design, and its excellent tuning features. This bandpass filter consisted of op amps (U2A, U2B, U3A in Figure 4.3), capacitors (C1, C2) and resistors (R3-R8). For testing with barley, these were set up to have a central frequency of 5000Hz and a bandwidth of 1000 Hz. The center frequency of the band could be shifted by adjusting a potentiometer (R5). Figure 4.4 shows the frequency response of the band pass filter.

The bandpass filter blocked low frequency straw and chaff signals as well as mechanical noise caused by the rotor rotation. Tests showed that the signal conditioners had excellent mechanical noise immunity for rotor speeds from about 800 to 1000rpm.

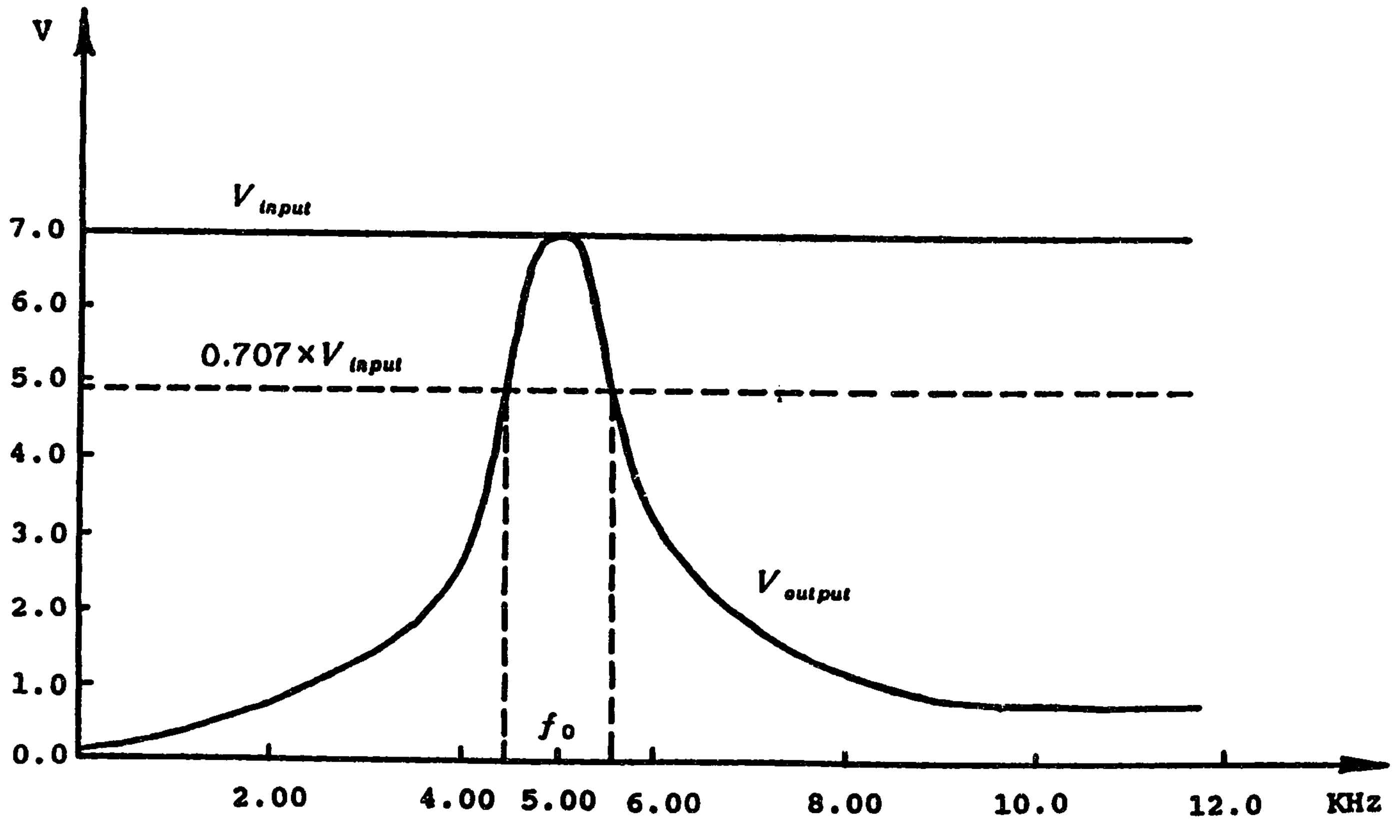
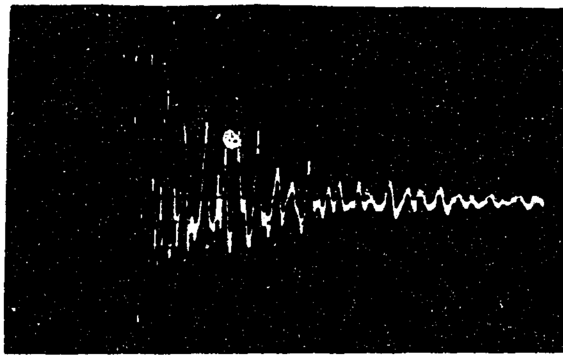
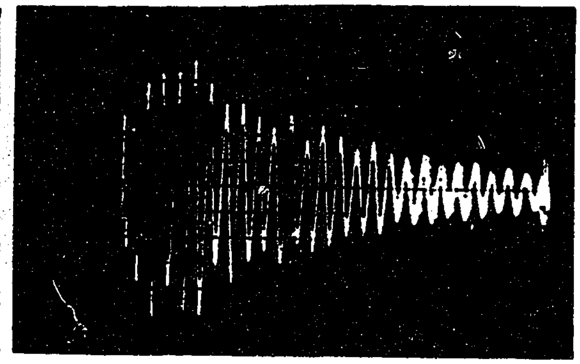


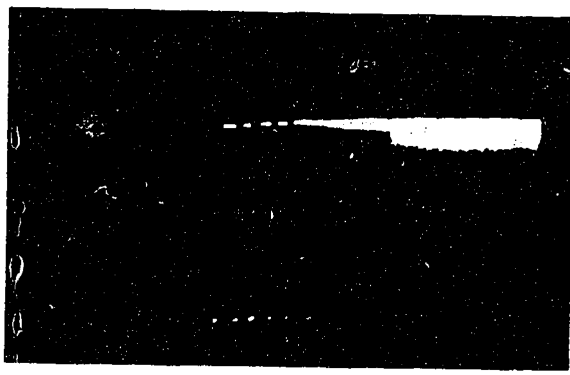
Figure 4.4 Bandpass Filter Freqeuncy Response.



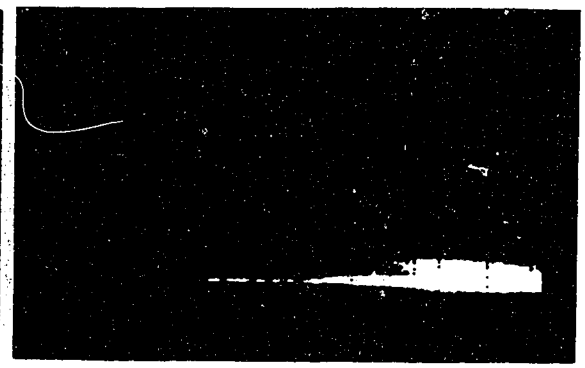
1. Output of Amplifier



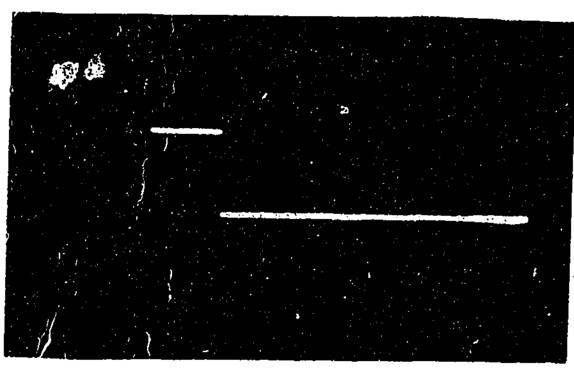
2. Output of Band-Pass Filter



3. Output of Comparator



4. Output of Transistor



5. Output of Monostable Multi-Vibrator

6. Output of Voltage Follower

Plate 4.2 Typical Kernel Signal Waveforms.

Note: 1. - 6. refer to locations on the signal conditioner circuit (Figure 4.3).

Plate 4.2 photo 1 and 2 show the waveforms of bandpass filter input and output signals. Higher and lower frequency components at the input side were attenuated and the output signals became nearly sinusoid and decayed at the frequency of 5000Hz.

4.2.2 Comparator

In addition to frequency discrimination, grain impacts also were identified on the basis of the signal amplitude. Output signals from the bandpass filter were compared with an adjustable reference voltage at two input terminals of the comparator U18. Thus, whenever the amplitudes of decaying impact signals were higher than the reference voltage (obtained from potentiometer R9), the comparator would give a train of positive pulses. Otherwise, the output would remain at the negative saturation voltage of the comparator (see Plate 4.2 photo 3). Adjusting the reference voltage with a potentiometer could increase or decrease the sensitivity of the sensors.

4.2.3 Precision Monostable Multivibrator.

The pulse trains from the comparators could not be used directly to provide grain impact counts. First, the negative part of the signals had to be removed using a high-speed switching transistor (2N3904) and the processed signals are shown in Plate 4.2 photo 4. Schmitt-triggers (SN74HC14) then were used to sharpen the positive pulses and condition the signals to CMOS levels. Finally, the first increasing edge of a pulse train was used to trigger a monostable multivibrator (HCF4538) that generated a single pulse for the duration of the decaying impact signal. This measure could avoid a single impact creating more than one pulse. HCF4538 is a precision CMOS component which could generate an accurate output pulse over a wide range of widths ($10\mu s - \infty$). The pulse width was set by external resistor (R12) and a capacitor (C3).

The monostable outputs (see Plate 4.2 photo 5) were fed through two-stage CMOS Schmitt-triggers and a voltage follower and transmitted as 3-5ms width pulses with amplitudes of +8V (see Plate 4.2 photo 6) along plastic-insulated wires to a single-board computer. Because of high signal-to-noise ratio, the system performed with high noise immunity.

4.2.4 Circuit Board Mounting

The signal conditioner printed circuit boards were attached to the base of each sensor in order to provide better signal conditioning and improve the noise immunity. Plate 4.3 shows three sensors with their signal conditioner

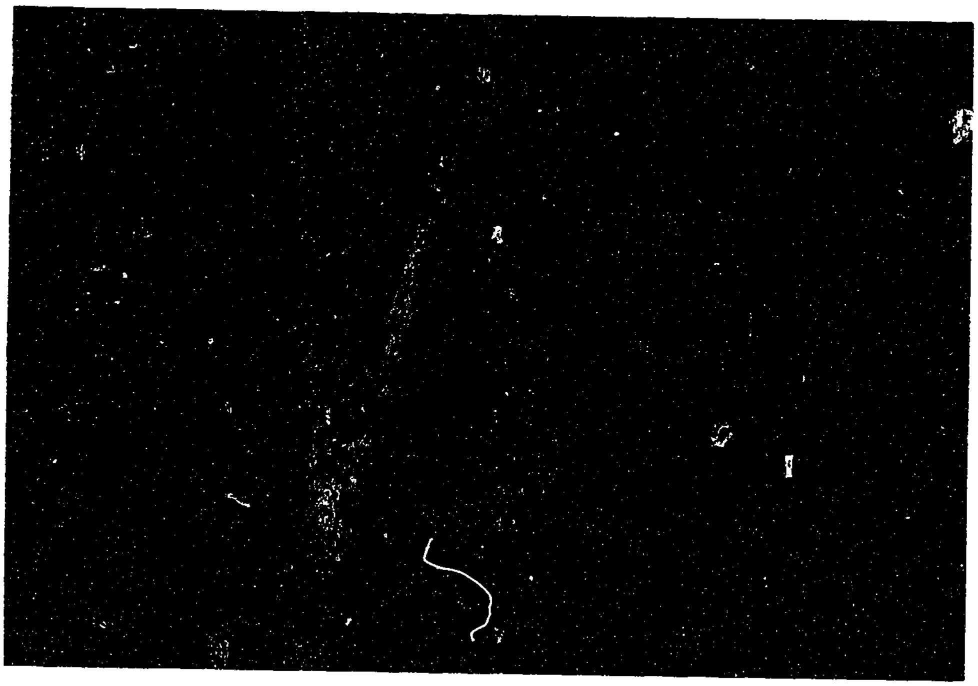


Plate 4.3 Sensors and Signal Conditioners.

printed circuit boards attached. This printed circuit board measured $110 \times 55mm$ and was smaller than the sounding boards. Four small screws fixed the circuit board to a pair of aluminium supports which were fixed to the sensor mounting screws. Plastic insulated washers were used to prevent the screws from contacting the circuit board directly.

A layer of acrylic coating was painted on the surface of these printed circuit boards as well as all the components on the circuit board to protect them while working in harsh threshing environments. These signal conditioners, located underneath a threshing rotor, gave no trouble during threshing tests.

5. MICROPROCESSOR-BASED DATA LOGGER HARDWARE DESIGN

The performance of a combine is affected by many factors. Sometimes the factors vary frequently (the density of the crop, for example). To respond to this situation, the operator has to adjust the ground speed as quickly as possible in order to keep the grain loss to an acceptable level. As the actual grain loss information source, the monitor should respond to the changes of harvesting circumstances quickly and accurately. This indicated that a high performance microprocessor should be selected as the system's central processing unit (CPU).

5.1 Microprocessor Selection

The monitor was a microprocessor-based system which provided multi-channel, high-precision data acquisition and processing; real time regression calculation and an operating system for the coordination of peripheral equipment. The complexity of the system made hardware and software design difficult and complicated. Thus, the system Central Processing Unit (CPU) selection was the most important decision for the monitor hardware design.

An 8-bit microprocessor data acquisition system would be capable of collecting the signals from nine sensors, but the ability of an 8-bit CPU to conduct complicated regression calculation in real time was questionable. As recommended by Leonard and Larson (1988), a 16-bit microprocessor was needed for the hardware design.

Of the various 16-bit microprocessors on the market, two were available for use:

1. Motorola M68000 16-bit microprocessor.
2. Intel 8086/8088 16-bit microprocessor.

The M68000 instruction set is powerful but relatively simple. Its sophisticated interfacing capabilities, and its ability to strongly support multi-tasking (Clements, 1987) made it a possible selection as a system processor. However, because there were no suitable microprocessor development devices immediately available, the M68000 was not selected for this project.

The Intel 8088 on the other hand is used in the IBM-PC computer and all the software could be programmed, debugged, and modified conveniently on an IBM-PC. Therefore the Intel-8088 was selected as the system CPU. The ready availability of development software was also a reason in making this selection. The feasibility of using an IBM-PC to develop an 8088-based instrumentation system had been confirmed previously (Liu et al. 1988). Details of this application are explained in Chapter 6 (Software Design).

5.2 Single Board Computer

The data acquisition and processing system was built around a 16-bit, single board computer. A photograph of the single board computer and printer are shown in Plate 5.1.

A forty-key, soft touch keyboard (DMC 40-key-Membrane-Switch Kit) and a sixteen-digit liquid crystal display (SHARP Dot-Matrix LCD Unit) were connected to the system to monitor data acquisition and to enter commands. A portable thermal printer (Radio Shack TP-10) was employed to print output files which included the original data from sensors, all the regression model parameters, some important intermediate results and the final predicted grain loss values.

The single board computer (SBC) circuit was built by wire wrapping and had to be able to withstand a hostile testing environment. A block diagram of the SBC and a signal conditioner are shown in Figure 5.1.

Usually, power surges, spikes, and static electricity are great "killers" of computers (Williams, 1988). The power surges and spikes could be avoided by using a voltage regulator to convert 12V from a battery to a 5V power supply. The use of a battery was considered appropriate also because a 12V DC power supply is a standard on combines. Static electricity countermeasures were taken to avoid system malfunction or chip damage during assembly and opera-

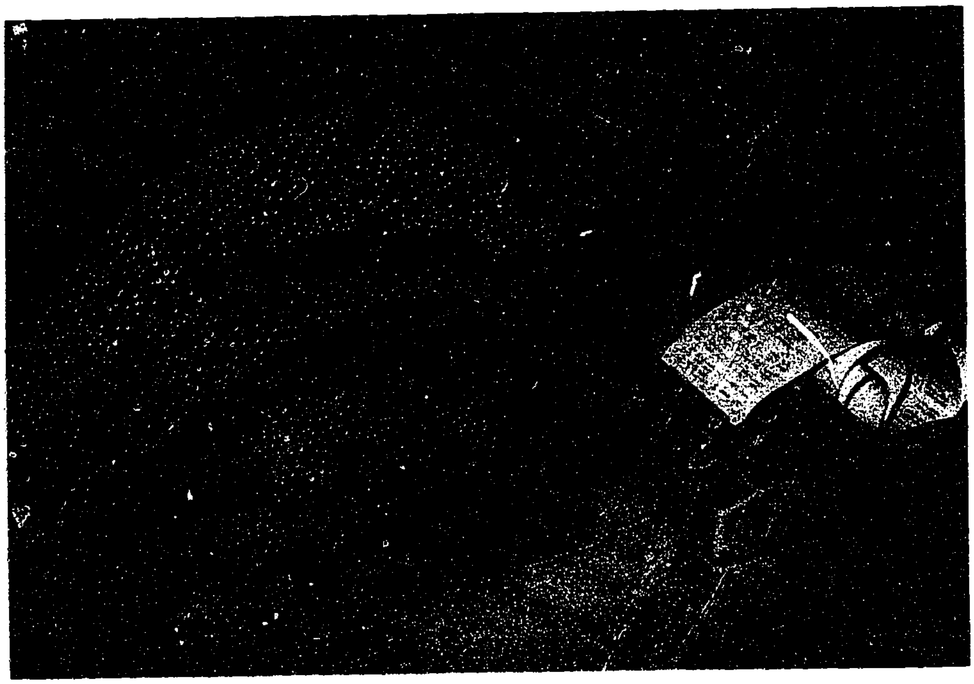


Plate 5.1 A Single Board Computer and a Printe:.

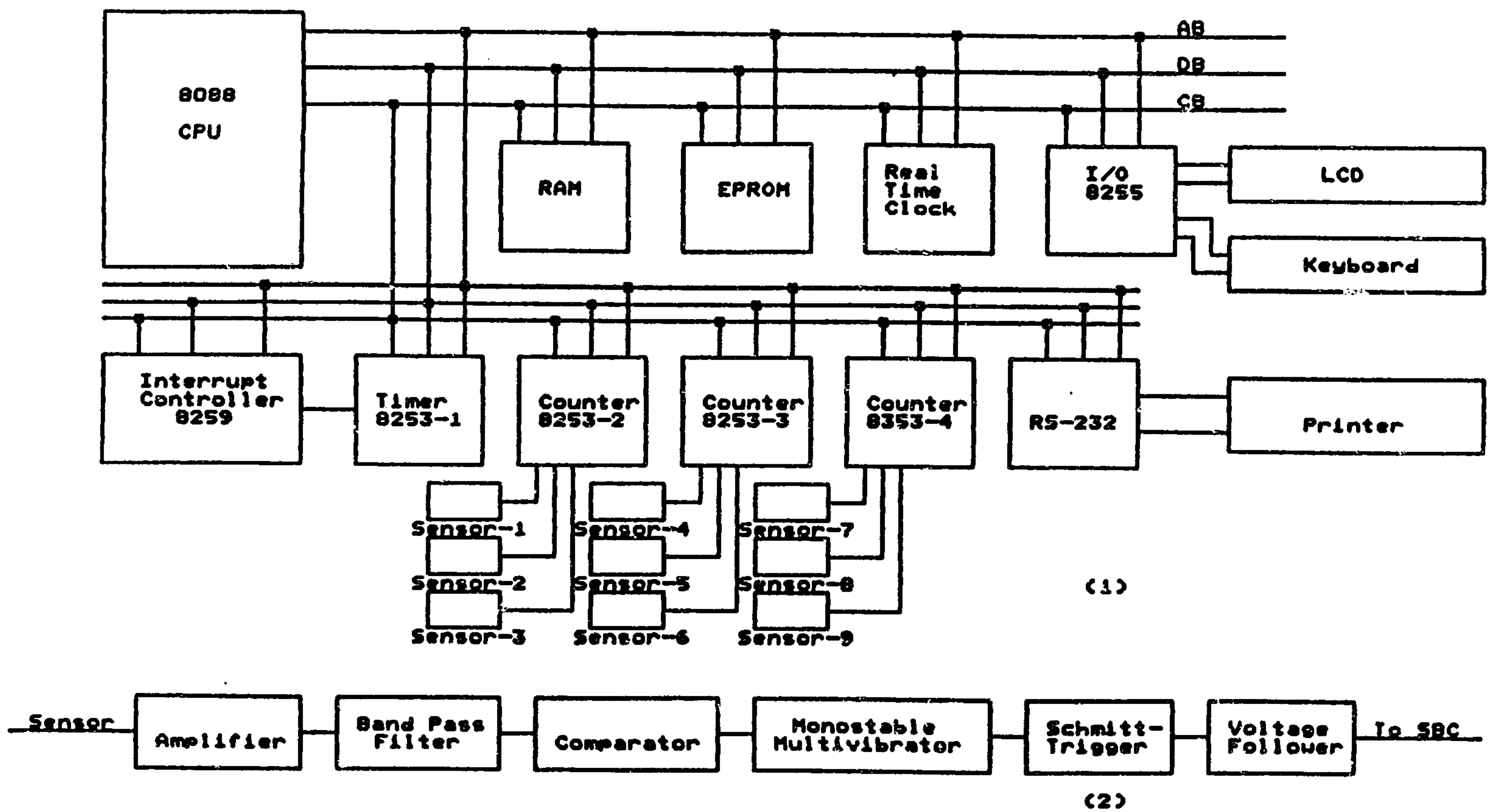


Figure 5.1 A Block Diagram of the Single Board Computer(1)
and a Signal Conditioner(2)

tion.

Threshing tests were carried out during late winter and early spring of 1989. Heaters in the test building were working day and night and the air in the building was dry. The threshing equipment (International Harvester 1460 rotor assembly) was set on dry wood blocks and, thus, was insulated from the ground.

Crop materials were rubbed in the rotor and static charge accumulated in the threshing equipment. For the first of several preliminary threshing runs, the monitor system behaved erratically because the SBC was set in a plexiglass box. Static was conveyed to the SBC via the signal feed wires or by dust and, subsequently, interfered with the normal performance of the monitor system. Later, a sheet metal box was built for the SBC and the +5V power supply negative terminal was also connected to the box which was grounded. All the signal input terminals were isolated from the outside environment by opto-isolators (Appendix II 3. Nine-channel Signal Interface) to improve the monitor system noise immunity. No further problems were encountered due to static electricity.

In order to outline the monitor system structure and its main functions clearly, the instrumentation hardware and its functions are listed in Appendix I. Table 1.

The system memory capacity was 26k byte. One 8k EPROM was reserved for the operating system for the keyboard, LCD and printer control, as well as for the real time data processing routines. Another 8k EPROM was used as storage for look-up tables. The remaining 10k was RAM which was used for data acquisition and processing, storage of intermediate results, and system stack. A detailed memory map is shown in Appendix II. 1. Memory Map.

Parallel (8255) and serial (RS-232) interfaces were also employed to connect the SBC with the keyboard, LCD, printer etc. The interface address map also is shown in Appendix II. 2. I/O Map.

5.3 Timer/Counters and Interrupt Controller

The data acquisition system utilized four Timer/Event Counters (8253). Three of these served as counters to collect the pulse signals from each of nine grain impact sensors, while one worked as a timer to control the sampling rate and as a baud rate generator for serial communications.

The programmable Timer/Event Counter (TC) consists of three identical counting circuits, each of which has clock (CLK) and GATE inputs and an output (OUT). Each can be viewed as containing a Control and Status Register pair, a Counter Register (CR) for receiving the initial count, a

Count Element (CE) which performs the counting but is not directly accessible from the processor, and an Output Latch (OL) for latching the contents of the CE so that they can be read (Liu and Gibson, 1986). The internal structure of a single TC is shown in Figure 5.2.

A Programmable Interrupt Controller (8259) was used to control the data acquisition. The interrupt technique was used so that counting kernel impacts and data processing could be conducted independently.

5.3.1 Sampling Rate Signal

Harvesting is a continuous operation and the grain loss varies continuously. In order to improve the accuracy of predicting actual grain loss, the data sampling time should be measured accurately and the gaps between two samples should be as small as possible to avoid losing any grain impact counts.

Figure 5.2 shows the sampling rate signal generator and the counter for channel #1 (for clarity, the other 8 channels were omitted) and its control circuit.

The sampling frequency generator was composed of two Timer/Counters (TC1 and TC2). TC2 OUT2 was connected to TC1 CLK1 and the system PCLOCK (peripheral clock 2.5MHz) was input into TC2 CLK2. TC2 worked as a square wave generator

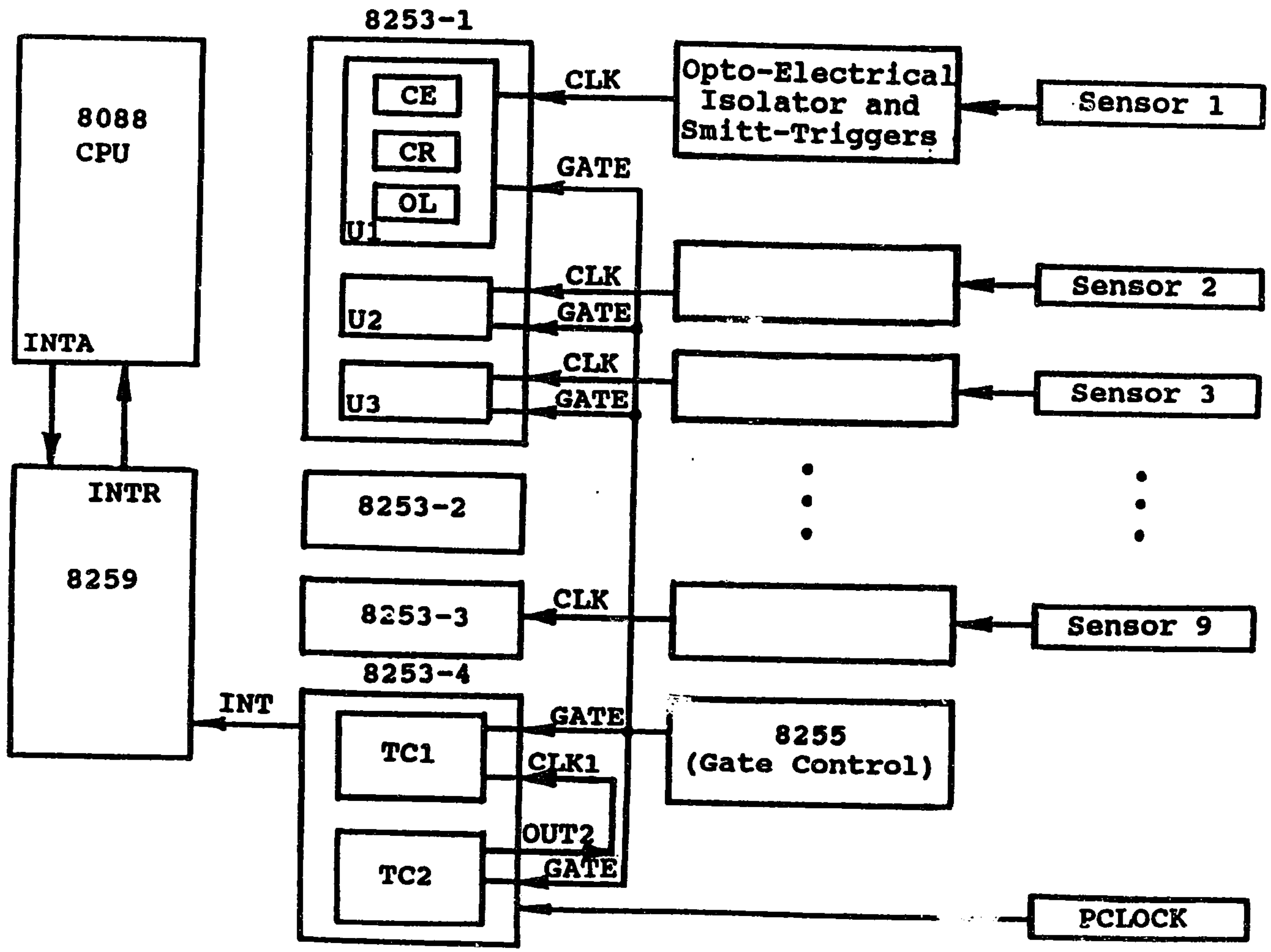


Figure 5.2 Sampling Rate Signal Generator, Counters and their Control Circuits.

(8253 mode 3), and split the 2.5MHz PCLOCK into a lower frequency (25,000Hz) square wave. TC1 provided an interrupt on count termination and split the square waveform further into a 2s sampling rate control signal.

The duration of sampling depended on the counts which were loaded in TC1 and TC2 CRs (Counter Registers) on system initialization. The duration of sampling could be calculated by:

$$Ts = \frac{(count1) \cdot (count2)}{f_p} \qquad 5.1$$

where:

Ts = sampling duration,

f_p = PCLOCK frequency (2.5MHz), and

count1 (100), count2 (50,000) = the initial counts loaded in CR1 and CR2, respectively.

Since the TCs were pulse-edge triggered devices, wide clock cycles increased the timing errors. To obtain an accurate sampling rate, count2 was selected as small as possible while count1 was as big as possible. For this project the sampling rate was 2s and count2 and count1 were 100 and 50,000 respectively. Since PCLOCK was an accurate system clock the only error source was the digital counter which was accurate to ± 1 bit. When these initial counts were loaded in the TCs, the sampling duration time error was:

$$\frac{T_1}{T_s}=\frac{0.04\times 10^{-3}s}{2s}=2\times 10^{-5} \qquad 5.2$$

where:

T_1 = TC1 1 bit counting time,

T_S = sampling duration.

5.3.2 Counting Process

When data acquisition started, the sampling rate generator (TC1, TC2) started to count the PCLOCK until TC1 counting terminated. When this occurred, the TC1 OUT terminal sent a two-second timing pulse to the system interrupt controller (8259) as an interrupt request. If the CPU interrupt was enabled, the interrupt was recognized and a service routine was executed. The interrupt service routine included the following steps:

1. Sending a Gate Control signal via parallel interface 8255 to close all the TCs' GATEs. Counting and timing were stopped temporarily.
2. Reading the nine sensor grain-impact counts stored in the TCs' Output Latch (OL) registers and writing the data into RAM for further real-time data processing.
3. Loading nine initial zero counts to the TCs' CRs and sending a pulse, via 8255, to clear all the TCs' OLs for the next sample.

4. Initializing the sampling rate generator (TC1 and TC2) and opening all the counters' GATEs by sending a Gate Control signal. At this moment, the next sample period started.
5. Conducting the Real Time Data Processing. After the service routine execution was over, the CPU would wait in a loop for the next interrupt.

The interrupt service routine execution time from step 1 through step 4 was the duration of the gap between two samples. This part of the procedure included 156 Intel 8088 assembly language instructions, their execution times are summarized in Appendix I. Table 2.

The 156 instructions to be executed between two samples took 963 system clock cycles. The system clock was 5MHz, giving a total execution time for the 156 instructions of only 0.1926ms. Therefore, compared to the width of a single kernel pulse signal (3-5ms), the gaps between two samples could be ignored. The grain impact signals would only be lost when the signal edges happened to appear at the gap. The possibility of this was only $1x10^{-4}$.

The advantage of this kind of sampling was that the system could monitor the combine operation continuously, and the total actual grain loss could be accumulated accurately sample by sample.

An experiment was carried out to verify monitor counting accuracy. A function generator (HP 3312 A, Hewlett Packard) was connected to a universal counter (HP 5314 A, Hewlett Packard) as well as the grain loss monitor counter input terminals (channel#1 through channel#9). The signal from the function generator was adjusted to resemble those from the sounding board signal conditioners. The initial count for sampling time generator TC1 was 50. This corresponded to a sampling duration of 1 second. This was the same sampling period of the universal counter. The counting results of the nine channels were displayed on the LCD in turn. Examination showed that the readings on all channels of the grain loss monitor and on the universal counter were always exactly the same over an input signal frequency range from 5Hz to 65536Hz.

The upper frequency limit of 65536Hz was determined by the counter (8253) 16-bit register capacity. Programming techniques could have raised this upper frequency limit but, for this project, it was not considered necessary.

6. SOFTWARE DESIGN

The object of this project was to design a real-time data-acquisition and processing system. This implied that a large amount of calculation should be conducted within one sampling duration. Because assembly language was very close to machine code, it could execute at high speed and, also, could be loaded onto a target system with a small memory capacity. Therefore, assembly language was selected as the programming language.

Because of the shortage of programming experience for real time data processing, the following general guidelines were developed for program development:

1. The total data-processing program execution time should be no longer than 1 second.
2. Subject to satisfying prerequisite calculation accuracy, the program should be as short as possible.
3. For further time saving, look up table techniques should be adopted.
4. A modular structure was used, so that the program would be constructed from a number of relatively small and simple modules.
5. The real time data processing program would be debugged on an IBM-PC. Some important intermediate and final

results would need to be compared to those data calculated by a BASIC program which possessed the same functions.

6. When the program was loaded onto the target system, sets of sensor impact data should be loaded to test the program executing on the monitor hardware.
7. The real-time data-processing program execution time was to be measured on an IBM-PC in the debug environment.

6.1 Mathematical Method and Calculation Accuracy Analysis

As mentioned previously, Wang et al. (1984) measured point separation with piezoelectric grain sensors mounted underneath one rotor of a rotary combine with two parallel rotors. These authors reported separation of wheat to decay exponentially with axial distance from the rotor front end, but only if the center region of the concave grate arc was considered. The mathematical representation of grain separation was reported as:

$$Z = Z_0 e^{-Bl} \qquad 6.1$$

where:

Z = grain separation rate (impact counts),

Z_0 = analytical initial separation rate (impact counts),

e = the base of natural logarithms,

B = attenuation coefficient (m^{-1}), and

l = axial distance from the front of the rotor (m).

As mentioned in 1.2.2.2, Bjork (1988) investigated separation around the grate of an axial-flow combine. Equation 1.2, derived from Bjork (1988), could be written in the form of expression 6.2 to describe the separation distribution around the grate arc:

$$Z = A_j + B_j X + C_j X^2 \qquad 6.2$$

where:

Z = point separation (impact counts),

A_j, B_j, C_j = coefficients determining shape of curves,

X = coordinate around concave arcs or grate arc (m),

j = the row number of sensor matrix.

The software design for real time data processing was based on the above equations (6.1 and 6.2) for grain separation distribution in the axial and lateral directions respectively.

6.1.1 Mathematical Models

In this project nine sensors (S1-S9) were located beneath the rotor grate in three rows with three sensors in each row (see Figure 6.1). Data from these sensors were used to obtain regression equations to the separation curves

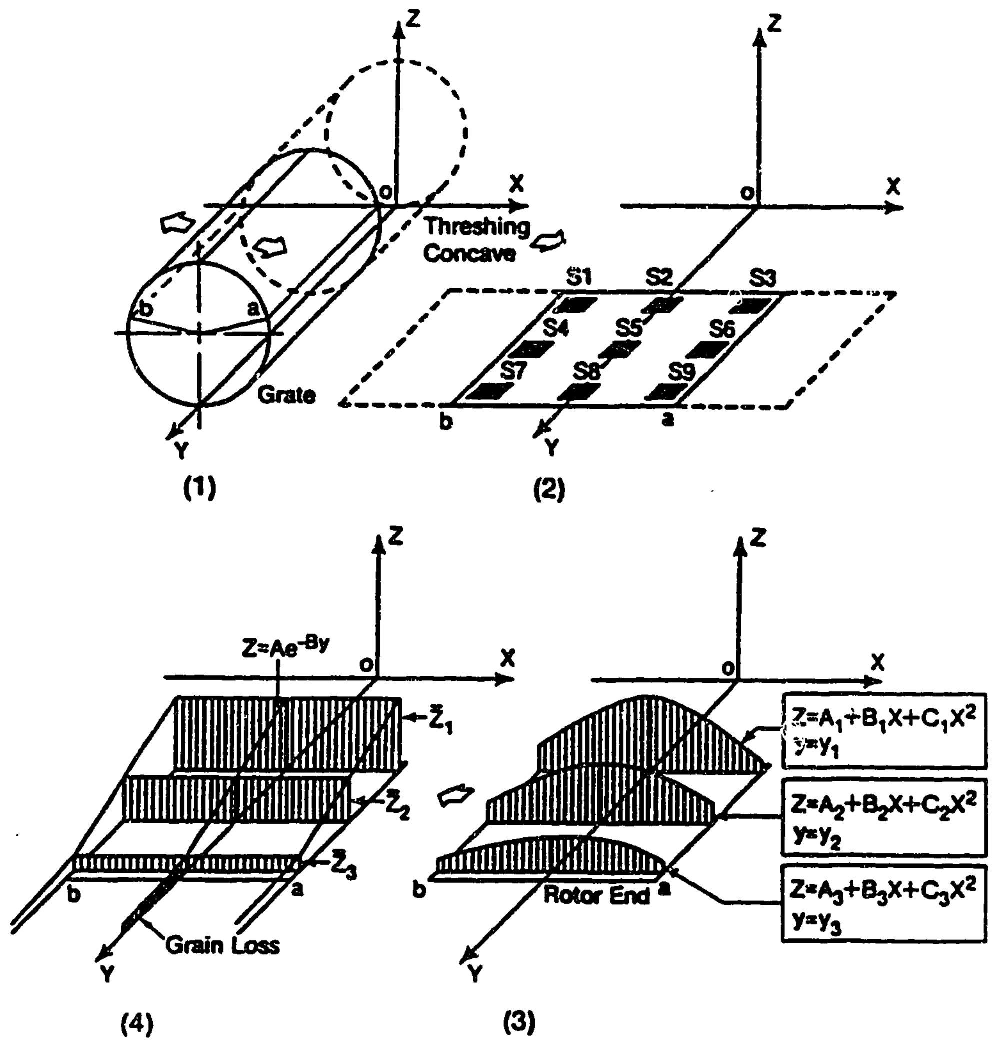


Figure 6.1 The Separation Distribution Underneath the Rotor Grate Section, Sensor Locations and the Mathematical Models.

along and around the grate. These equations were in the form of equations 6.1 and 6.2, respectively.

Mean separation values were determined at each axial distance and these were used to determine the longitudinal separation equation. The latter was integrated from the end of the rotor to infinity to determine the actual grain loss. This procedure is shown schematically in Figure 6.1.

An alternative to the integral mean of equation 6.2 is the arithmetic mean. Though this is simpler, it would not provide as good an estimate of the true mean as the integral mean because of the sensor locations. Therefore the integral mean was calculated for its closeness to reality.

By using the least squares method, equation 6.2 could be expressed as a matrix equation (6.8) to calculate the parameters A_j, B_j, and C_j (j=1, 2, 3 indicates the sensor row). For clarity, only A_1, B_1, and C_1 will be mentioned in the following explanation of the derivation of equation 6.8.

Equation 6.2 can be written as a second-order linear regression model (Draper and Smith, 1981):

$$Z_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i \qquad 6.3$$

where:

Z_i=the impact count on the i_{th} sensor at axial position 1 (i=1, 2, 3 indicates the sensor number),

$\beta_0, \beta_1, \beta_2$= parameters determining shape of the curve,

X_i=location coordinate of the i_{th} sensor around the grate,

ϵ_i=the curve deviation from the true value.

The sum of the squares of deviations from the curve was:

$$S = \sum_{i=1}^{3} \epsilon_i^2 = \sum_{i=1}^{3} (Z_i - \beta_0 - \beta_1 x_i - \beta_2 x_i^2)^2 \qquad 6.4$$

The estimates A_1, B_1, and C_1 were chosen so that the values substituted for $\beta_0, \beta_1, \beta_2$ in equation 6.3 produced the least possible value of S. A_1, B_1, and C_1 could be determined by differentiating equation 6.4 with respect to $\beta_0, \beta_1, \beta_2$ and setting the results equal to zero.

$$\frac{\partial S}{\partial \beta_0} = -2 \sum_{i=1}^{3} (Z_i - \beta_0 - \beta_1 x_i - \beta_2 x_i^2) = 0$$

$$\frac{\partial S}{\partial \beta_1} = -2 \sum_{i=1}^{3} x_i (Z_i - \beta_0 - \beta_1 x_i - \beta_2 x_i^2) = 0 \qquad 6.5$$

$$\frac{\partial S}{\partial \beta_2} = -2 \sum_{i=1}^{3} x_i^2 (Z_i - \beta_0 - \beta_1 x_i - \beta_2 x_i^2) = 0$$

so that the estimates A_1, B_1, and C_1 were given by

$$\sum_{i=1}^{3}(Z_i - A_1 - B_1 x_i - C_1 x_i^2) = 0$$

$$\sum_{i=1}^{3} x_i(Z_i - A_1 - B_1 x_i - C_1 x_i^2) = 0 \qquad 6.6$$

$$\sum_{i=1}^{3} x_i^2(Z_i - A_1 - B_1 x_i - C_1 x_i^2) = 0$$

Rearranging equation 6.6 leads to:

$$3A_1 \quad + B_1 \sum_{i=1}^{3} x_i \quad + C_1 \sum_{i=1}^{3} x_i^2 = \sum_{i=1}^{3} Z_i$$

$$A_1 \sum_{i=1}^{3} x_i + B_1 \sum_{i=1}^{3} x_i^2 + C_1 \sum_{i=1}^{3} x_i^3 = \sum_{i=1}^{3} x_i Z_i \qquad 6.7$$

$$A_1 \sum_{i=1}^{3} x_i^2 + B_1 \sum_{i=1}^{3} x_i^3 + C_1 \sum_{i=1}^{3} x_i^4 = \sum_{i=1}^{3} x_i^2 Z_i$$

which can be written in matrix form as equation 6.8.

$$\begin{pmatrix} 3 & \sum x_i & \sum x_i^2 \\ \sum x_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{pmatrix} \begin{pmatrix} A_1 \\ B_1 \\ C_1 \end{pmatrix} = \begin{pmatrix} \sum Z_i \\ \sum x_i Z_i \\ \sum x_i^2 Z_i \end{pmatrix} \qquad 6.8$$

Equation 6.8 could be simplified as equation 6.9.

$$(X) \begin{pmatrix} A_1 \\ B_1 \\ C_1 \end{pmatrix} = \begin{pmatrix} \sum Z_i \\ \sum x_i Z_i \\ \sum x_i^2 Z_i \end{pmatrix} \qquad 6.9$$

It could be proved that matrix (**X**) was nonsingular ($|X| \neq 0$), so that the Cramer's rule (Williams, 1978) could be used to determine the unique solution (A_1, B_1 and C_1). Since all the elements in matrix (**X**) were constants (sensor fixed locations (or location squared, cubed) in a raw) $|\mathbf{X}|$ could be calculated by hand as a constant. Equation 6.10 was used to determine parameter A_1 :

$$A_1 = \frac{1}{|X|} \begin{vmatrix} \sum Z_i & \sum x_i & \sum x_i^2 \\ \sum x_i Z_i & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 Z_i & \sum x_i^3 & \sum x_i^4 \end{vmatrix} \qquad 6.10$$

Parameters B_1 and C_1 could be obtained in the same way.

The actual procedure followed by the assembly language program is summarized below.

1. Accumulate and store grain impact counts in memory over a set time interval (2s).
2. Read the data for the sensors S1, S2, and S3 from memory.
3. Conduct the best fit calculation using the least squares method by solving the matrix equation. This procedure provided the coefficients for the separation function

   $$Z_1 = A_1 + B_1 x + C_1 x^2 \qquad 6.11$$

   at $y = y_1$

4. Calculate the mean value of Z_1 using:

$$\bar{Z}_1 = \frac{1}{a-b}\int_b^a (A_1 + B_1 x + C_1 x^2)dx \qquad 6.12$$

where:

(a-b) = the length of apertured part of the grate in the grate arc direction (m).

5. Repeat for the other axial locations $y = y_2$ and $y = y_3$. The mean separation values ($\bar{Z}_1, \bar{Z}_2$ and $\bar{Z}_3$) then could be used to obtain the longitudinal equation:

$$\bar{Z} = Ae^{-By} \qquad 6.13$$

where:

$\bar{Z}$ = mean separation distribution in the axial direction.

Equation 6.13 can be linearized in the form:

$$\mathrm{Ln}\,\bar{Z} = \mathrm{Ln}\,A - By \qquad 6.14$$

6. A look up table procedure was used to obtain the logarithms of $\bar{Z}_1, \bar{Z}_2$ and $\bar{Z}_3$.

7. To calculate parameter B, the regression calculation using least squares was carried out by solving the equations:

$$-B = \frac{\sum_{j=1}^{3}\left(y_j - \bar{y}\right)\left(\text{Ln}\,\bar{Z}_j - \overline{\text{Ln}\,\bar{Z}}\right)}{\sum_{j=1}^{3}\left(y_j - \bar{y}\right)^2} \qquad 6.15$$

and

$$\text{Ln}\,A = \overline{\text{Ln}\,\bar{Z}} - B\bar{y} \qquad 6.16$$

where:

$$\bar{y} = \frac{1}{3}\sum_{j=1}^{3} y_j$$

$$\overline{\text{Ln}\,\bar{Z}} = \frac{1}{n}\sum_{j=1}^{3} \text{Ln}\,\bar{Z}_j$$

and

Y_j = the location of the sensor rows.

8. The parameter A was calculated using the Taylor series expansion:

$$A = e^{T} \approx 1 + T + \frac{T^2}{2!} + \ldots + \frac{T^8}{8!} \qquad T = \text{Ln}\,A \qquad 6.17$$

9. Having determined the coefficients A and B of equation 6.13, the predictd grain loss was determined using the Taylor series and multiplication:

$$GL = K\int_{y_0}^{\infty} Ae^{-By}dy = \frac{KA}{B}e^{-By_0} \qquad 6.18$$

$$K = K_1 K_2 C$$

where:

GL = grain loss (impacts),

y_0 = the coordinate of the rotor end (0.974m),

$K_1 = \frac{1}{0.07 \times 0.11 m^2} = 129.87 m^{-2}$, the reciprocal of the effective sampling area of each sensor,

K_2 = a calibration coefficient obtained in threshing calibration,

C = the length of the rotor apertures in the grate section arc direction (1.29m), and

$$e^{-By_0} \approx 1 - By_0 + \frac{(By_0)^2}{2!} - \frac{(By_0)^3}{3!} + \ldots + \frac{(By_0)^8}{8!} \qquad 6.19$$

The assembly language program for this real time data processing included more than three thousand instructions (Appendix IV).

6.1.2 Data Processing Precision Analysis

Assembly language programming was a time consuming and delicate job, but under the condition of real time data acquisition and processing, there were no higher level computer languages that could offer the same high execution speed and ability to access computer hardware directly. In order to reduce the execution time and improve calculation precision, some important techniques were introduced in the software design. These included the calculation of e^T and

LnA.

Only eight terms in the e^T Taylor series were used. This reduced execution time, but the truncation error may increase. When $|T| > 1$, more terms were needed for truncation error reduction. To cope with the conflict of these two aspects, the exponential was divided into integer and decimal fraction parts,

$$e^T = e^a e^b \qquad T = a + b \tag{6.20}$$

The value

$$e^a \qquad (|a| > 1)$$

was found in a look-up table stored in the system EPROM and

$$e^b \qquad (|b| < 1)$$

was obtained by Taylor series. The two parts then were multiplied together. The truncation error is given by (Wang et al. 1984).

$$R_n(b) = \frac{b^{n+1}}{(n+1)!} e^{Qb} \qquad 0 < Q < 1 \tag{6.21}$$

Now, because

$$|b| < 1 \qquad 0 < Q < 1$$

then

$$e^{Qb} < e < 3$$

and, because, $b^{n+1} < 1$, it follows that

$$|R_n(b)| < \frac{3}{(n+1)!} \tag{6.22}$$

When n = 8, the truncation error

$$|R_n(b)| < \frac{3}{9!} \approx 8.27 \times 10^{-6} \tag{6.23}$$

For the integer part (a) of the exponential calculation, a was sent to an index register and the first address of the look-up table was sent to the base register. The sum of the base and index registers (both were inside the CPU) provided the effective address where the result was located. The flow charts of look-up tables for both $\mathrm{Ln}\,\bar{Z}$ and e^T are provided in Appendix III 4.

The next step was to call a multi-precision multiplica- on subroutine to obtain the result e^{a+b} .

The $\mathrm{Ln}\,\bar{Z}$ Taylor series was more complicated than that of e^T. A permanent natural logarithm table was stored in the system EPROM. A look-up table procedure was written to provide the logarithm for an operator range from 1 to 99×10^4. The integer number was changed into floating point representation following which the look-up table procedure was run followed by multiply and add procedures. For example:

$$\bar{Z} = 926 = 9.26 \times 10^2$$

$$\text{Ln}\,\bar{Z} = \text{Ln}\,9.26 + 2\text{Ln}\,10$$

Ln 9.26 and *Ln10* could be found in the natural logarithm table, then, by calling multiplication and addition subroutines, Ln $\bar{Z}$ could be obtained. The calculation became easy and fast.

The data in the natural logarithm look up table had four significant digits. Usually the original data from the sensors and the integral mean values were no more than three digits. Thus, no additional error would be accumulated when the look-up table technique was used.

6.2 Software Structure

Almost all programs are structured according to a hierarchy, with the top of the hierarchy being a controlling program module, called the main program, that directs the execution of the modules that lie just below it in the hierarchy. In turn, these principal submodules control the use of their submodules and so on (Liu and Gibson, 1986).

The combine grain loss prediction and monitoring program included 75 modules and their execution was under the direction of two main programs. One was the main program (system initialization) and the other was the interrupt service program. The flowcharts of this two programs are shown in Appendix III. 1 and 2.

The Real Time Data Processing (RTDP) submodule was the biggest module in the hierarchy. It was called in the execution of the interrupt service program when the data from the sensors were ready to be processed. The RTDP submodule subsequently would call its three submodules: arc best fit for parameters A_j, B_j, C_j; axial regression for parameters A and B; and actual grain loss (GL) prediction. In turn, these modules controlled the use of their submodules. The lowest modules were multi-precision arithmetic routines such as addition, subtraction, multiplication, division, and the submodules of look-up tables, data conversion and transmission etc. The hierarchical diagram for the RTDP submodule is shown in Appendix III 3.

The design sequence of the hierarchy structure was from the top to the bottom, but programming and debugging should be conducted from the bottom to the top. Thus, when programming at higher levels of the hierarchy, the programmer only concentrated on the current level because the lower levels had been well-debugged. This strategy made the programming easier and faster.

6.3 Real Time Data Processing Program Executing Time

The real time data acquisition and processing were conducted in parallel. The real time data processing for one group of samples had to be finished before the next group of samples was ready to be processed. If the execution time was longer than the sampling duration, then this real time data processing program was useless. Determining the real time data processing program execution time before loading the program onto the target monitor system was, therefore, very important.

A real time clock was available in the IBM-PC and, also, the IBM-PC system ROM-BIOS (Read Only Memory Basic Input Output System) had an interrupt service routine (interrupt #26) to read the clock at any time. The system clock "ticks" by generating an interrupt #8 at specific intervals. On each clock tick, the ROM-BIOS interrupt #8 service routine increases the clock count by 1. The clock ticks at a rate that is almost exactly 1,193,180/64KHz, or, roughly, 18.2 times a second. The count is kept as a 4-byte integer at low-memory location hex 46C (Norton, 1985).

A call-subroutine instruction was inserted at the beginning and the end of the RTDP submodule to read the real time clock built in the IBM-PC. By running the RTDP submodule, the current time (which was updated 18.2 times a second) would be read by the clock-read subroutine and then stored in RAM. The RTDP submodule executing time then could be obtained by subtracting two stored times.

The results showed that the clock only updated once or twice which meant that the RTDP execution time was between 54.9 ms and 110 ms. Thus, the CPU would spend more than 89% of its time waiting for the next group of samples to be ready while executing a series of loop instructions.

7. SENSOR LOCATIONS AND INSTALLATION

To obtain separation curves, the sensors were required to be mounted underneath the separation grate section of the rotor and located to measure the separation distribution and, consequently, predict the grain loss. The sensor location scheme was based on supplying sufficient data for the real time regression program to generate three separation distribution curves in the lateral direction and one curve in the axial direction. Figure 6.1 shows the sensor locations and representative separation distribution curves underneath he rotor grate section in both lateral and longitudinal directions.

In arriving at the arrangement shown in Figure 5.1, two aspects were considered. One was the number of sensors required and the other was the sensor locations.

7.1 Sensor Numbers

In this project nine sensors (S1-S9) were located beneath the rotor grate in three rows with three sensors in each row (see Figure 6.1).

For the curve regression, the more sensors used, the higher would be the precision achieved. However, for measurement system installation, maintenance and cost, the fewer sensors used the better.

There were three parameters (A_j, B_j, C_j) in the second order polynomial model of separation distribution in the lateral direction (equation 6.2), and two parameters (A and B) in the exponential model of axial direction separation distribution (equation 6.13). It is always possible to exactly fit n data points with a polynomial of degree n-1 or less (Blank 1980). Thus, a minimum of three sensors were required in the lateral direction to solve the equations and to obtain the parameters A_j, B_j, and C_j. Three mean values, derived from each lateral row of sensors, were used to obtain the exponentially decaying curve in the axial direction (i.e., obtain parameters A and B). The use of three values would result in better curve fits than the use of the minimum two mean values to solve the regression model with two parameters. In turn, this would lead to higher accuracy in the prediction of actual grain loss by the integration of the separation distribution curve from the rotor end to infinity.

7.2 Sensor Locations

Sensor distribution along the full length of the rotor concave and grate sections could supply more information for percentage loss prediction and yield map drawing (Bjork, 1988). However, problems such as high cost, low reliability and difficulty of maintenance might be encountered. The peak separation rate under the threshing concave could lead to impacts at a high enough frequency to saturate sensors and reduce measurement accuracy. Since the distribution underneath the rotor grate segment in the axial direction was expected to be an exponentially-decaying curve, only the grate section of the rotor was considered in the sensor installation.

Figure 7.1 shows a cross section of the rotor grate segment of the International Harvester 1460. The top part of the rotor shell (104° *to* 256°) was closed while the bottom part had apertures.

Both Wang (1984) and Bjork (1988) observed a peak separation rate near the center of the grate. Wang (1984) believed that the reason for this peak was that the summation of the forces acting on the grain, straw and chaff perpendicular to the concave surface would change as the material moved in a lateral direction. At the central part of the grate the summation of forces was greater than at any other point. Following this reasoning, one sensor was installed at the central point in the lateral direction to detect the peak separation rate.

Although separation processing could not occur along

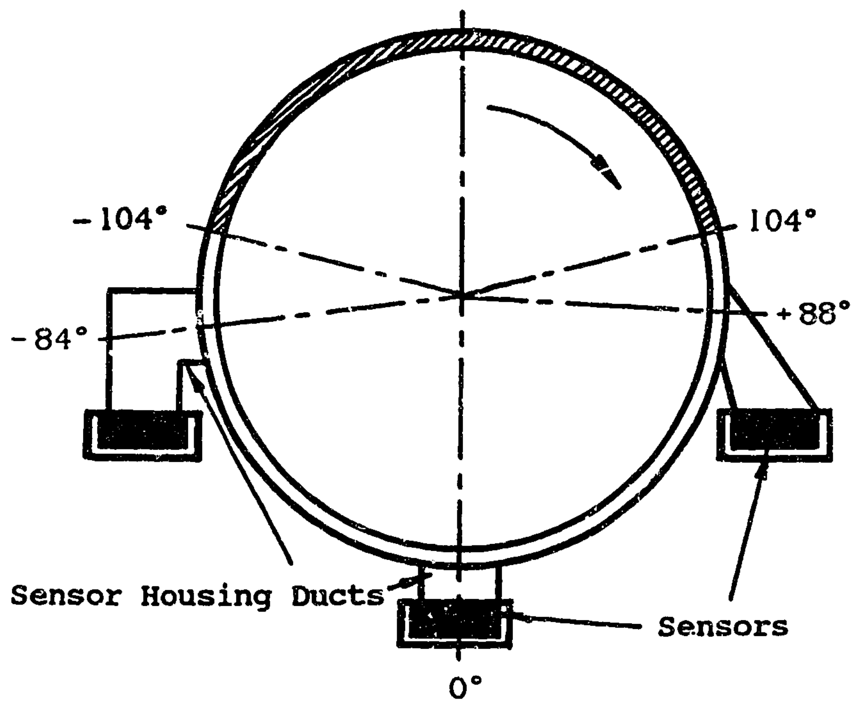


Figure 7.1 Cross Section of the Rotor Grate Segment.

the top of the rotor, the grain could be separated from the straw and chaff matt and gather near the top inside surface of the rotor. This phenomenon would cause the separation value to be non-zero at the border between the closed and apertured parts (Bjork, 1988). Thus, two locations (+88° and -84°), which were close to the border of the closed and apertured regions were selected for sensor installation to measure the separation rate near the border area.

7.3 Sensor Installation

All nine sensors were housed in sheet metal ducts which were attached with screws to the rotor grate at the predetermined positions shown in Figure 7.1 and Figure 7.2. The duct row locations were 0.227, 0.461 and 0.827m from the front end of rotary grate segment. The ducts were based on those used by Bjork (1988) for his rotary combine threshing characteristics study.

Minor modification was needed to add a part at the bottom of every duct to install each sensor and its signal conditioner. Wang (1984) and Larson (1987) studied the optimum orientation of the sensor sounding boards to grain flow. Both authors determined that the sensor orientation affected its sensitivity and accuracy. The best angle between the sounding board and grain flow was determined to

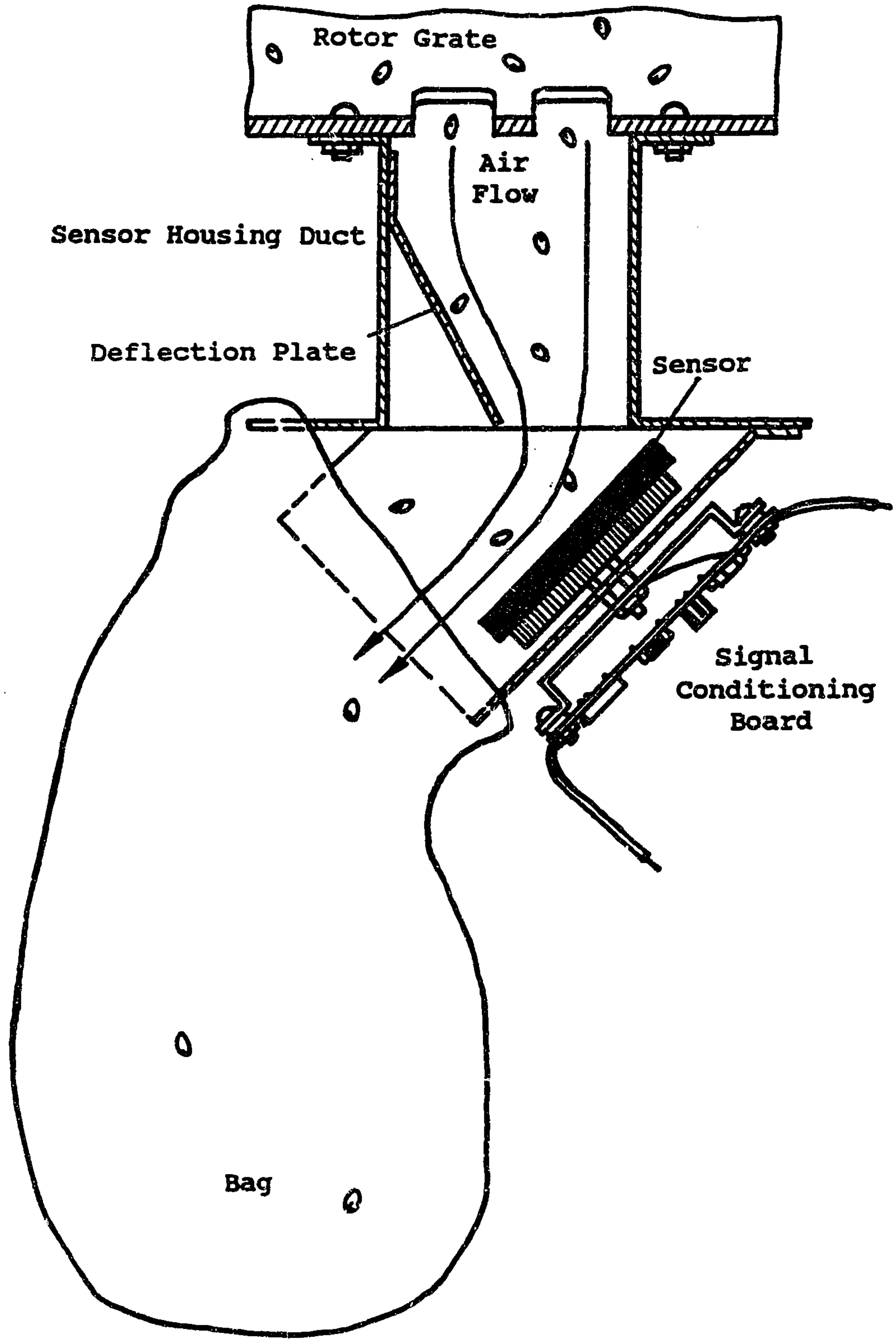


Figure 7.2 Sensor Housing Schematic Layout.

be 40° *to* 45°. Figure 7.2 shows a schematic layout of the duct, sensor and signal conditioner installation.

The rotor, which was turning at 800 or 1000 rpm for the experimental tests, caused the crop materials between the rotor and the grate to move in nearly spiral path. Ideally, the kernels would leave the rotor in planes perpendicular to the rotor axis. Therefore, the sensor surfaces should make an angle of 45° with these planes. The sensor orientation and row location is shown in Plate 7.1.

In order to obtain the same sampling area, the inlet area of the ducts should be identical. Thus, every inlet covered four grate holes (see Figure 7.3). Calculation of the equivalent area of the inlet was based on the assumption that the measured separation rate was the mean value of all points of the sampling area (see Figure 7.4).

The measured separation rate was the total kernels collected in $70 \times 110 mm^2$ sampling area within one sampling period (see equation 7.1). This mean separation rate was taken to represent the central point of sampling area which was also the sensor location.

$$S_p = \frac{N}{0.07 \times 0.11 \times 2 m^2 s} = K_1 N \qquad 7.1$$

where:

S_p = Separation rate N/m^2s,

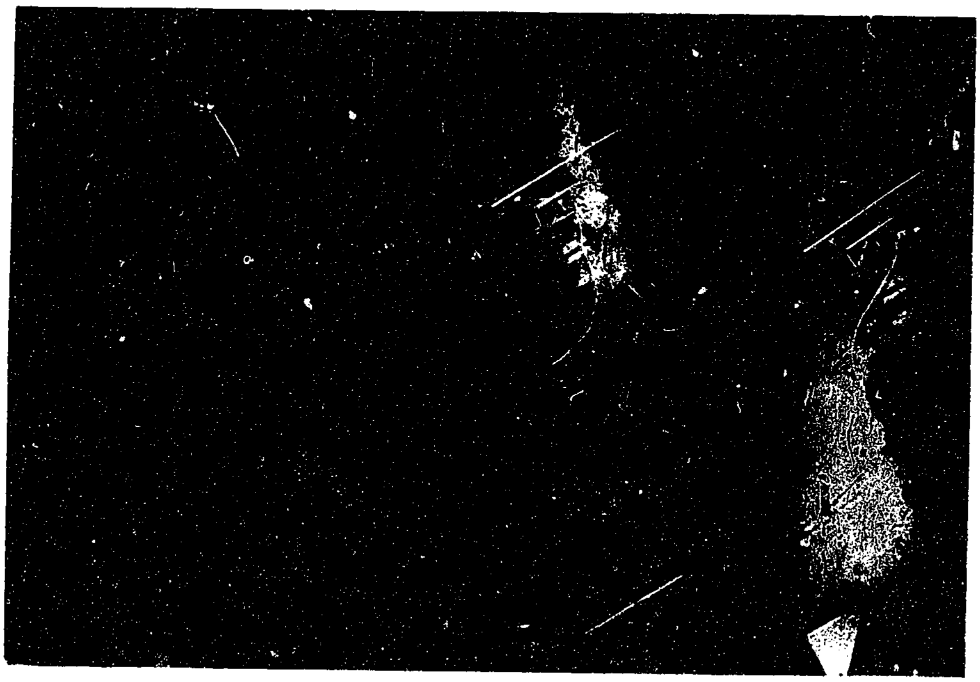


Plate 7.1 Sensor Orientation and Row Location.

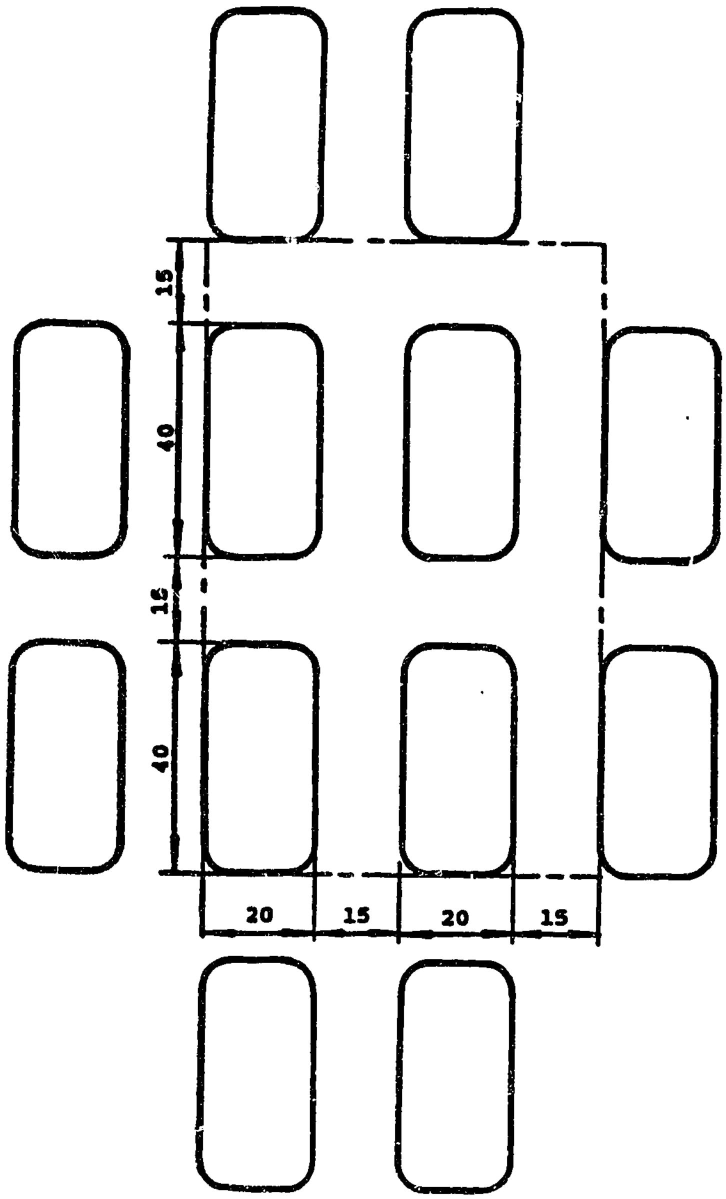


Figure 7.3 The Sampling Area at the Grate Section
(dimensions in mm).

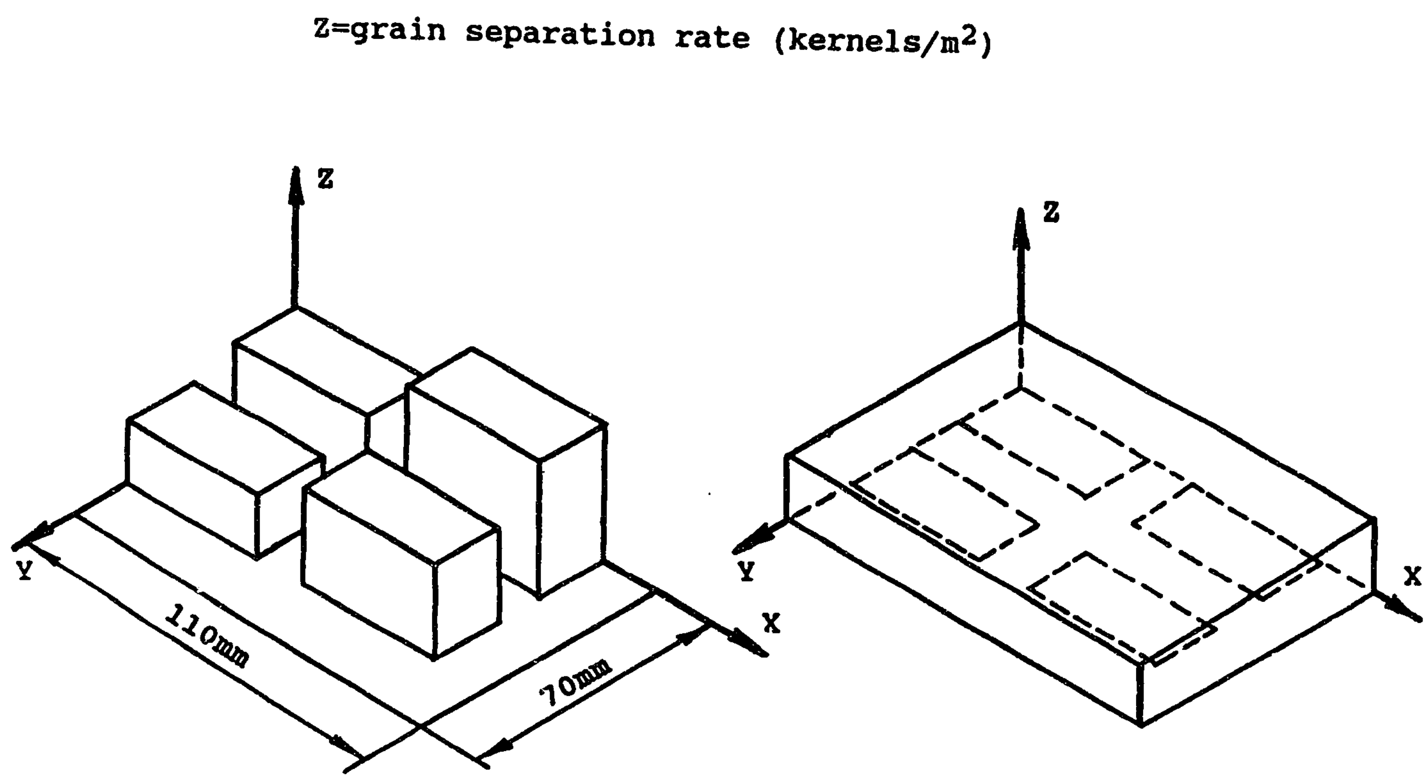


Figure 7.4 Sampling Area Separation Distribution and Assumed Mean Separation Value.

N= the number of kernels.

In each case the inlet area was $70x110mm^2$ and the outlet area was $100x40mm^2$. The effective sounding board sensing surface was the projected area of duct outlet on the sounding board surface i.e.

$$\frac{100\times 40}{\cos 45°} = 5657mm^2 \quad 7.2$$

This meant that 58.9% of the sounding board surface area ($160\times 60mm^2$) was exposed to the discharged Grain and MOG flow. However, because the length of the sensor surface (160mm) was longer than the length of the duct outlet and, since it was very important that all the seeds coming through the inlet of the duct be detected by the sensor surface, a deflection plate was located inside the duct to control kernel direction (see Figure 7.2).

Using sheet metal ducts to sample separation and to provide sensor housings not only directed the seed flow toward the sounding board but, also, allowed determination of the exact sensor locations and sampling areas.

In addition, other advantages of this system included:

1. Kernels were leaving the rotor and going into the ducts at different speeds and directions. Restrained by the duct walls and the deflection plate, high speed kernels would lose some energy and kernel speeds would tend to be

Uniform impacts would provide uniform signals and enhance signal conditioning.

2. The air flow blowing through the duct would change its direction near the sounding board (see Figure 7.2). Relatively light materials, especially chaff, would be separated from the Grain and MOG mixture mat. Some chaff might no longer contact the sounding board surface and this would improve the sensitivity of the sensor.

No experiments were carried out to verify the above theories, but the results of the threshing experiments confirmed them in a subjective way.

To keep the air and the material flowing smoothly (see Figure 7.2), the outlet of the ducts should not be smaller than the actual apertured area of the grate sampled. In this design, the outlet area of the ducts was $4000mm^2$, and the sampled apertured area was $3200mm^2$.

The Grain and MOG mixture going through the outlet with the equivalent area would do so more easily than going through four small holes on the rotor grate segment. No unexpected choking or blocking of materials happened inside the ducts during a total of 76 threshing runs. The interference of the duct-sensor installation to the threshing performance, therefore, could be ignored.

8. SIMULATION AND THRESHING EXPERIMENT

Equipment for transportation, feeding, threshing, cleaning and weighing was needed for crop threshing experiments. Three to four people also were needed to operate the equipment. Also, the effects of the harsh environmental conditions on the electronic test system had to be considered. Therefore, the experiments were divided into two main parts: simulation experiments and threshing experiments. Thus, unexpected failures and unnecessary labour and time stage could be avoided.

Simulation Experiment

The hardware and software design of the test system had to be tested before actual threshing experiments in order to confirm the feasibility of the test method and reliability of the system. A threshing simulation experiment was carried out by using a modified grain kernel delivery system, to deliver controlled grain flows to the sensor matrix. The grain flows simulated grain separation distribution underneath the rotor grate section of a combine. This laboratory simulation also allowed the sensors to be calibrated.

8.1.1 Magnetic Feeder Modification

A magnetic vibratory feeder (Syntron BF2A) was utilized to deliver the grain kernels.

Because the grain loss monitor was a multi-sensor measurement system. The single outlet magnetic feeder was not suitable for delivering grain kernels for simultaneous calibration of nine sensors. Furthermore, the reliability of the grain loss monitor system needed to be confirmed using all sensors before actual threshing tests were carried out. A grain delivery system was built (see Plate 8.1) for grain distribution simulation and sensor calibration.

This system could be used to:

1. Preset the grain feed rates to nine sensors,
2. Simulate the separation distribution underneath the rotor at nine specific points where the sensors would be located,
3. Collect the grain which hit individual sensors for calibration, and
4. Allow the sensitivity of different points of any sensor to be measured easily.

A piece of 15mm acrylic sheet with 9 holes (d=18mm) was attached to the outlet of the magnetic vibratory feeder. Nine adjustable slide gates were made which could be used to control the grain delivery rate through each hole. Grain

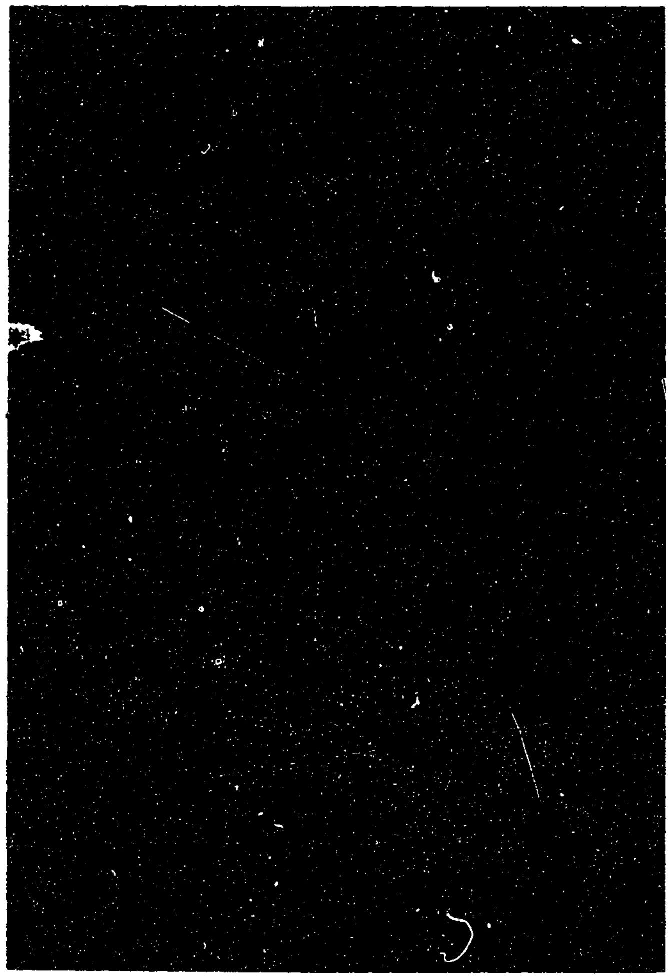


Plate 8.1 Magnetic Feeder Assembly for Grain Distribution Simulation and Sensor calibration.

would drop into nine small hoppers which were located beneath the feeder outlets. Kernels would subsequently be delivered to the nine sensors via nine, semitransparent, soft plastic tubes.

Making an angle of 45° to the horizontal, all the nine sensors, along with their signal conditioners, were set in plastic boxes. After hitting the sounding boards, kernels would fall into the plastic boxes and could be collected for counting.

An acrylic frame with 27 holes was located above the sensors and grain collection boxes to hold kernel delivery tubes. The outlets of kernel delivery tubes were approximately perpendicular to the ground and the kernel stream was kept at 45° to the sounding board surface. On the frame, there were three holes above each sensor so that the kernel delivery tubes could be located to evaluate the sensitivity at the right, the central and the left parts of the sensor surfaces.

The kernel collection boxes were set on a wheeled table so that the upper or lower sides of the sounding boards could also be tested if the table was shifted in the appropriate direction.

8.1.2 Sensor Calibration Using Simulation Equipment

The accuracy of the predicted value of grain loss depends on the accuracy of the sensors. The sensors could be calibrated using three potentiometers (R5, R9 and R12) on each signal conditioner (see Figure 3.3). Proper adjustment of these potentiometers could improve the sensitivity and selectivity of the sensors. However, if all the potentiometers were adjusted at the same time, it would be very difficult to select the point at which the sensor signals were conditioned optimally.

As mentioned in section 3.1, the impact signal frequency was within a narrow band and the time for the signal amplitude to decay to a certain value was independent of the height from which the kernels were dropped. Therefore, two potentiometers were preset. One (R5) was for adjustment of bandpass filter central frequency and the other (R12) was for adjustment of the monostable multivibrator high level output duration. For barley, preset adjustment provided the same central frequency (5000Hz) and the same monostable status duration (3ms) for all the 9 sensor signal conditioners.

Only one potentiometer (R9) needed to be adjusted for the comparator reference voltage during calibration. This made the calibration procedure much easier than adjusting all the potentiometers.

The sensor calibration sequence was as follows:

1. Program an EPROM. The function of this program was to open all nine counter's gates for 10 seconds and count the grain impact pulses. The results were printed out.
2. Barley kernels were put in the hopper of the magnetic feeder and the slide gates were set at the appropriate positions.
3. All the sensors were placed in their grain collection boxes and the grain delivery tube outlet positions were verified to be above the appropriate points of the sounding boards to be tested.
4. The signal conditioner and the SBC were powered and the sounding boards were tapped manually and the signals from conditioners were examined on an oscilloscope (this step was needed only at the beginning of the test to check correct function of the circuitry).
5. When "PRESS F2 TO TEST" was displayed on the SBC's LCD, function key F2 was pressed and, at the same time, a stop watch was started and the magnetic feeder was turned on.
6. At 8-9 seconds from the start time, the magnetic feeder was stopped and all the kernels in the grain delivery tubes would descend on the sensor sounding boards before the time limit (10s).

7. The kernels in the collection boxes were weighed. The numbers of kernels were calculated using the following formula and compared with the numbers in the printed file.

$$N = \frac{W}{W_{tk}} \qquad 8.1$$

where:

N=the number of kernels,

W=the weight of kernels (g) in each collection box, and

W_{tk}=the weight (g) of one thousand kernels of grain used.

8. The reference voltages of the signal conditioners were adjusted by means of the potentiometers (R9) to decrease the count errors.

9. The above steps were repeated and the count errors were calculated. By using Students' 't' test (Steele and Torrie, 1980) the errors based on 95% confidence interval sould be within $\pm 10\%$.

The count error was defined by:

$$error = \frac{N_a - N_m}{N_a} \times 100\% \qquad 8.2$$

where:

N_a = number of kernels (actual),

N_t = number of kernels (tested).

The null and alternative hypothesis for this calibration were:

$$H_0: \bar{E} = 0$$

$$H_1: \bar{E} \neq 0$$

where:

$\bar{E}$=the mean of the errors (%).

The statistic of the test is

$$t = \frac{\bar{E} - 0}{s/\sqrt{n}} \qquad 8.3$$

where:

n=44 sample size,

$\bar{E} = -4.486$ sample mean,

s=19.77 standard deviation.

The 't' statistic described by equation 8.3 was calculated to be -1.505 with 43 degrees of freedom. The null hypothesis was accepted at the level of $\alpha = 0.05$. The 95% confidence interval places the mean of the population (calibration error) between -6.023% and +6.023%.

Calibration results (Appendix I Table 3.1) show that adjusting the reference voltage could change the sensitivities of the sensors. The test error sign could be changed from negative to positive (or vice versa) when the reference voltage of the comparators were increased or decreased (see Appendix I Table 3.1.1 Sensor 2).

8.1.3 Separation Distribution Simulation

Grain separation distribution curves underneath the rotor grate section could be represented by second order polynomials in the arc directions and decaying exponential in the axial direction. By using a modified grain delivery system, simulation tests were carried out to create sensor counts which were close to previously measured separation rates at the proposed sensor locations (Bjork, 1988). To do this, the slide gates of the magnetic feeder were adjusted so that the middle sensors of each row were hit by higher grain flows than the other two sensors in the same row. Similarly, the mean delivery rates in each row were adjusted to simulate the decaying separation distribution in the axial direction.

The grain delivery system was started and the data acquisition system was run for ten seconds. The sensor impact counts and calculated "grain loss" results then were printed out. The symbols of the regression parameters A_j, B_j, C_j and A, B should satisfy the distribution functions as mentioned above.

One sample result from such a run is shown below:

SAMPLE ------------- 04

S1 = 0285 S2 = 0495

S3 = 0204 S4 = 0090

S5 = 0195 S6 = 0072

S7 = 0031 S8 = 0079

S9 = 0060

A1 = +0495.0768 B1 = -0033.5769

C1 = -0337.5228 $\bar{Z}_1$ = 0356.0000

A2 = +0195.0257 B2 = -0004.3028

C2 = -0153.9039 $\bar{Z}_2$ = 0132.0000

A3 = +0079.0155 B3 = +001.1118

C3 = -0034.7235 $\bar{Z}_3$ = 0053.0000

Ln $\bar{Z}_1$= 0005.8721 Ln $\bar{Z}_2$= 0004.4427

Ln $\bar{Z}_3$= 0003.9703

A= +0533.1630 B= +0003.0043

GL= K*0010.3149

The above simulation test example shows the collection of sensor counts (S1 to S9) and the results of the regression calculations. The sign of parameter C_i determined the shapes of the distribution curves in the arc direction. Thus, if $C_i > 0$, the shape would be concave and if $C_i < 0$, the shape would be convex. In this sample all the signs of C_i were neg , so that these three curves were convex in shape.

The RTDP program took three mean values $\bar{Z}_1$, $\bar{Z}_2$ and $\bar{Z}_3$, which decayed in the axial direction, and found their logarithms in a look-up table. A least-square regression was conducted in the axial direction to calculate the parameters A and B. The sign of parameter B determined the shape of the exponential curve. If parameter $B > 0$, e^{-By} would be a decaying curve in the axial (Y) direction and this was confirmed in all the simulation test results.

In this example the predicted actual grain loss was 1,728. That is 1,728 kernels were "lost" with straw within a 2-second sampling duration.

8.2 Threshing Experiment

The purpose of threshing tests was to confirm feasibility and reliability of the hardware and software design in an actual combine environment. In addition to verifying the stability of the system in a harsh threshing environment, the tests were carried out to calibrate the sensors and to analyze the accuracy of the predicted actual grain loss values.

8.2.1 Threshing Test Apparatus

The threshing tests were conducted in the Harvesting Laboratory at the University of Alberta's Ellerslie Research Station during the late winter and early spring of 1989. A schematic diagram of the test apparatus is shown in Figure 8.1.

The International 1460 Harvester rotor assembly was used for the threshing tests. The rotor assembly was housed in a plywood box to contain threshed materials and dust. Two removeable walls on each side of the housing box provided access to the rotor assembly for maintenance and sensor calibration. Sensor housing ducts, with the sensors and signal conditioner circuit boards attached, were installed on the rotor grate section as shown in Figures 7.1, 7.2 and Plate 7.1.

A pair of 15.2m crop conveyers was set in front of the rotor to deliver the crop to the feed auger of the combine. The feed speed of the conveyer was controlled by a gear box and the crop feed rate could be controlled further by the amount of crop loaded on the conveyers. The conveyers were driven by an electric motor.

A CASE-International 7130 128KW tractor was located

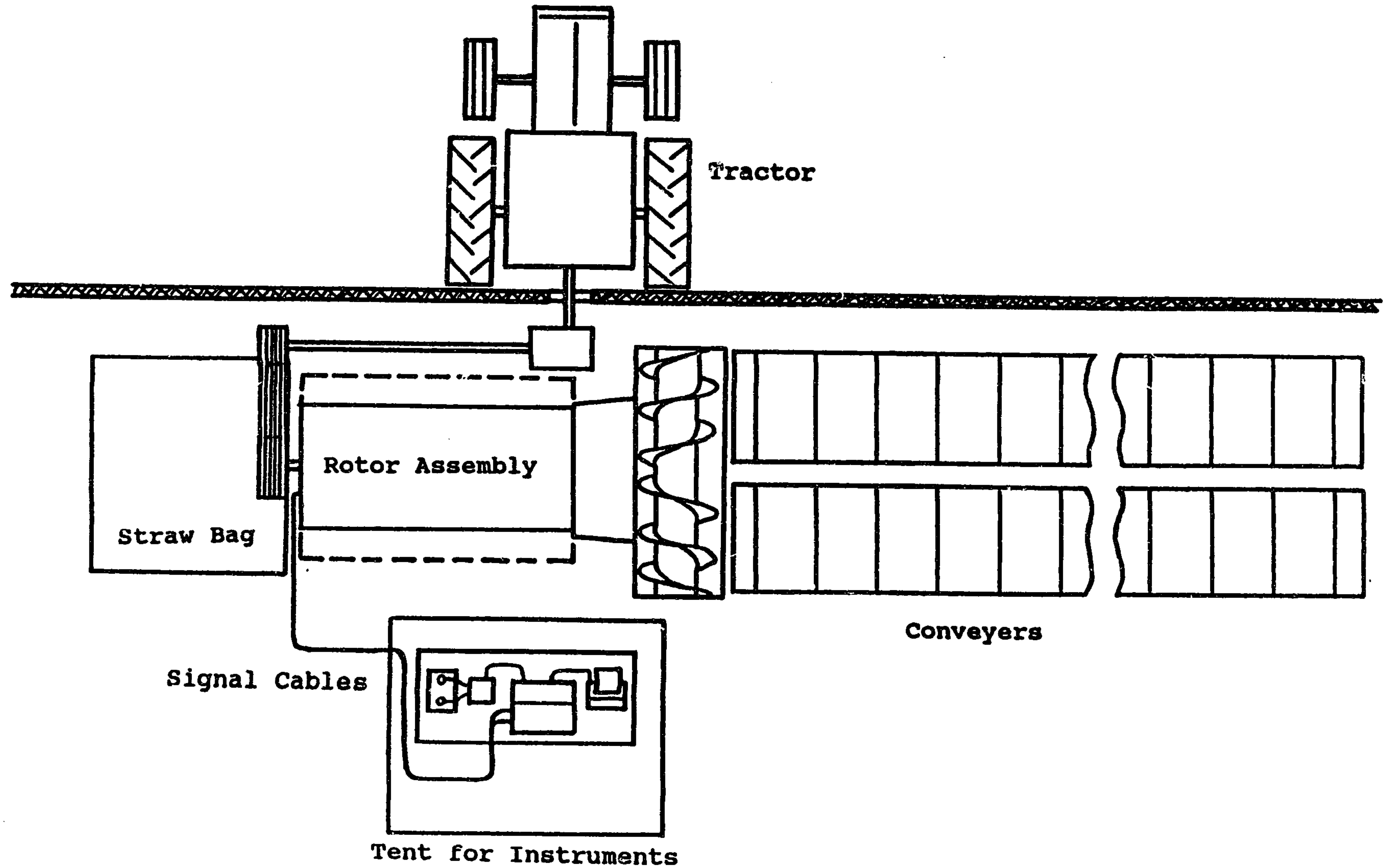


Figure 8.1 Threshing Test Equipment

outside the building with its power takeoff shaft going through a hole in the wall of the building to the rotor assembly transmission system to supply threshing power.

A control panel was mounted on the right side of the rotor assembly near the auger. This panel could control three electrical motors. One DC motor was used to change the rotor transmission ratio (rotor speed) and the other two AC motors were used to drive the conveyers and the elevator which lifted the crop from the auger table and fed it into the rotor.

The test equipment, which consisted of a single board computer, portable printer, 12V battery and power supply regulator, was set in a $2 \times 2.5 \times 2$ m plastic tent to protect the equipment from dust. To cope with interference from static electricity, as mentioned in section 4.2, a piece of antistatic sheet, which was well grounded, was laid underneath the SBC on a table. A piece of steel ($10 \times 4 \times 3$cm) with a piece of wire connected to the metal case of the SBC was also set on the sheet to discharge any possible static on the SBC.

A $1.5 \times 1.5 \times 1.5$m cloth bag was hung on a metal frame at the straw discharge outlet of the rotor to collect threshed straw, chaff and lost grain. A straw cleaning machine, designed and built by the Prairie Agricultural Machinery

Institute (PAMI), was used to clean the straw and collect the lost kernels.

Threshed grain, chaff and a small amount of straw were collected by 18 pails located underneath the rotor. This material was cleaned and used to determine G/MOG ratios (see Appendix I Table 7).

A small set of aspirated sieves (SEEDBURO 618W) was used to clean samples of separated material for sensor calibration.

Mechanical scales were used to weigh the crop material on to the conveyers and the straw, chaff and grains that were collected during threshing.

8.2.2 Description of the Material for Threshing

Preharvested whole crop (barley), which was piled outside the laboratory building, was the threshing test material. The selection was based on the availability of this crop from other experiments. However, significant loss rates had been observed in barley with a similar rotary combine (section 1.2.2.3) and barley was considered more difficult to detect with impact sensors than, say, wheat. It was, therefore, a good test crop.

The crop bundles were moved into the building and set with the heads upright a couple of weeks before the threshing tests started, thus the moisture content tended to be uniform. Later, the straw samples were weighed before and after drying in a microwave oven to measure the moisture contents. The results (see Appendix I. Table 4.)showed that highest and lowest straw moisture contents were 9.7% and 16.8% with a mean value of 14.2% w.b. Grain samples also were tested by using a CENCO Moisture Balance (Appendix I Table 5) and their moisture contents were between 11.0% and 13.0% with a mean value of 11.9%w.b. Since the heads of the crop were exposed to warm dry air inside the building for two weeks, grain moisture contents were lower and more uniform than that of the straw.

The mass of one thousand grain kernels was obtained from four samples and the mean was 36g (see Appendix I Table 6).

8.2.3 System Noise Immunity Test and Sensor Calibration

Mechanical vibration, audio-frequency noise and electromagnetic noise would be generated when the rotor and elevator were operating. These could influence system operation and sensor sensitivity, and the influence of the above factors needed to be established before conducting the final threshing tests.

8.2.3.1 System Noise Immunity Test

The function of an acoustic sensor, which is very sensitive to vibration and audio-frequency noise, is similar to the function of a microphone. Although the central frequency of the band pass filters in signal conditioners was several times higher than rotor fundamental frequency, harmonic frequencies of the vibration and noise possibly could still be selected by the signal conditioners and eventually form false pulses. The nine eight-meter-long signal feed wires and one common signal ground wire were insulated with plastic without shielding. Therefore, electrical noise generated by equipment in the building was easily picked up by these wires. A noise signal (50mv p-p) was detected near the SBC interface due to the long signal feed wires (see Plate 8.2). Although the opto-electric isolators and Schmitt-triggers at the interface of the SBC had high noise immunity, this still needed to be tested in the harsh threshing environment.

The system immunity test was important and simple to conduct:

The rotor was run at the speeds which would be used in threshing tests. No crops were fed into the rotor (empty run). Data acquisition equipment was started and the elevator and conveyer switches were turned on and off contineuously. After data acquisition stopped, the sensor data

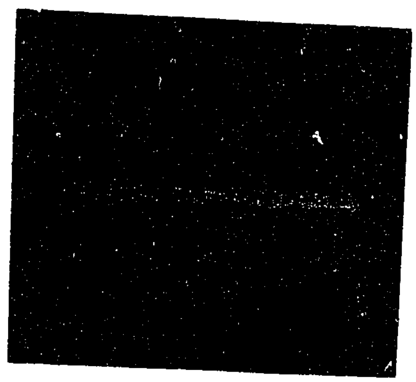


Plate 8.2 Noise at the Long Signal Feed Wires

in the printed file were checked. If the system performed with high noise immunity, all the sensor data would be zero.

After the first test run, only one non-zero count was found. The reason was that one sensor housing mounting screw was too long and its end was just touching the sounding board surface. When this screw was replaced by a short one, no false data were found in subsequent noise immunity tests.

8.2.3.2 Sensor Calibration Using Threshing Equipment

As stated in section 8.1.2, the material used for sensor calibration with the modified grain delivery feeder, was barley seeds. Calibration accuracy was ±10% for a 95% confidence interval. However, in actual threshing conditions, a grain, chaff and straw mixture would contact the sounding boards and the sensitivity of the sensors could be reduced. Lower reference voltages would, therefore, need to be selected to maintain reasonable sensitivity. Thus, further calibration of the sensors was carried out after installation on the combine rotor.

Calibration was carried out using grain-collection bags hung on the outlet of the sensor housing ducts. Two triangular holes were left in both sides of the outlet (see Figure 6.2) to provide a passage for air flow and to minimize the bag interference.

The crop feed rate was 7.2T/h and rotor speeds were 1000 and 800 rpm for sensor calibration. After threshing, material in the bags was cleaned and the samples were counted by hand.

The calibration results are shown in Appendix I Table 3.2.1 to Table 3.2.3.

Threshing tests for sensor calibration were repeated six times. The counted numbers of kernels were compared with the predicted numbers and sensitivity was adjusted by changing the reference voltages accordingly. The best reference voltage, which is printed in bold letters, was selected for the formal threshing tests.

A straight line (tested impact counts vs actual impact counts) was fitted and the slope of this straight line was obtained. Ideally, this slope should always be equal to one, but in real situation it only could be statistically close to one. Steeper slopes meant that sensors were too sensitive or vice versa.

A calibration coefficient (K_2 in equation 6.18) was then derived from this slope (reciprocal). It could be verified that this calibration coefficient could improve the grain loss prediction accuracy and reduce the error due to imperfect sensor calibration. If all the sensor counts were multiplied by this coefficient (K_2), the predicted grain losses would be affected as if the coefficient (K_2) was in equation 6.18. The reason was that the sensitivity of sensors could only affect the axial regression model parameter A (equation 6.16) but not parameter B (equation 6.14), so that the sensor sensitivity error could be adjusted linearly.

This calibration coefficient could be entered into the test system before the threshing test and the results showed that this coefficient was very important to improve the accuracy of the predicted grain losses.

8.2.4 Experimental Design and Threshing Test

The threshing time could be divided into three stages: rotor filling time (about 1-2s), stable time (depended on the crop feed time) and rotor emptying time (2-4s). To raise the test accuracy and reduce the influence of the transient stages (1 and 3), the length of the stable stage was required to be much longer than that of the transient stages.

Initial threshing tests (Liu and Leonard, 1989) using a short crop feeding time of 6s gave a low regression correlation between predicted and actual losses (0.67). This was due to the limited threshing duration which, in turn, depended on the length of the crop conveyers and the crop feed time. A longer threshing duration was needed to eliminate interference of the transient stage and to improve the accuracy of predicted losses.

A long crop feed time (12s) was adopted for the final threshing tests. The total data acquisition time for each run was 20s, which was 8s longer than the crop feed time. Therefore, no unexpected grain impact signals would be lost due to the transient time of stages 1 and 3.

Two rotor speeds (800 and 1000rpm) and two feed rates (6T/h and 12T/h) were selected. There were four combinations (12T/h-800rpm, 12T/h-1000rpm, 6T/h-800rpm and 6T/h-1000rpm). Each combination included three replicates giving a number of total threshing tests of twelve. The sequence of these 12 runs (shown in Table 8.1) was random.

Weighed crop material was evenly distributed on both conveyers with the heads toward the combine auger. This crop orientation and distribution was close to real situations when a combine is harvesting in the field.

When all the preparations were complete, the elevator and the conveyers were started and the crops were fed into the rotor. Function key F2 on the keyboard of the SBC was pressed to start the real time data acquisition and processing when the crop reached the auger.

Straw and lost grain were collected by straw bag (section 8.2.1) for determination of actual lost grain. The crop feed time was recorded by a stop watch. The regression parameters and predicted losses were printed out on the portable printer after each test run.

To evaluate the test system, a meaningful hypothesis was formulated. The null hypothesis of the 't' test for this experiment was that the mean of the ratio of actual and predicted grain losses which were obtained in all test runs was equal to one. An alternative hypothesis would be that the mean of above ratio was not equal to one.

The null and the alternative hypothesis were:

H_0: $\mu = 1$

H_1: $\mu \neq 1$

To test the null hypothesis H_0: $\mu = 1$, assuming that the population was normally distributed and, the test criterion is given by equation 8.3 (Steele and Torrie, 1980)

$$t = \frac{M - \mu_0}{s/\sqrt{n}} \qquad 8.3$$

where:

M = mean of ratios sampled from the population,

$\mu_0 = 1$, specified population mean,

s = standard deviation of sample, and

n = sample size.

The best fit curves in the arc direction and the regression curves in axial direction and their corresponding scattered data were plotted to evaluate the curve fitting calculations. The best fit straight line also was plotted to correlate the predicted and the actual grain losses.

9. RESULTS AND DISCUSSION

Because of the non-homogeneous physical characteristics of the crop material and complicated movement of grain and MOG in the rotor, crop threshing and grain separating were random processes. Therefore, data acquisition, real time data processing and the analysis of test results were based on statistical theory.

9.1 Sensor Calibration Analysis

Data are printed in normal and bold letters (in Appendix I Table 3.1.1 to Table 3.2.3 (six Tables in total)). The bold data are the data collected under the condition of final signal conditioner reference voltages.

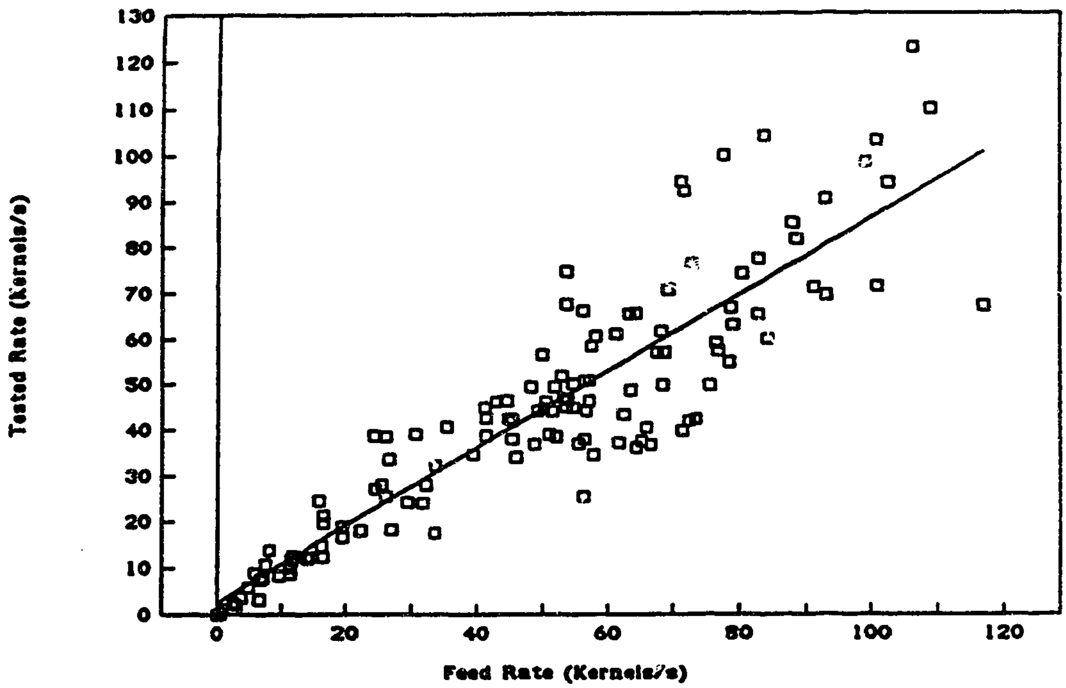
To test the effect of calibration, sensed grain counts were regressed against actual counts using straight-line, least squares regression. Overall and selected data were used to conduct the straight-line regressions and the statistics are shown in Appendix I Tables 3.4.

Straight lines were fitted to the simulation calibration result (test vs actual). The overall data (Table 3.1.1 to 3.1.3) regression had 129 observations with R^2=0.81 and slope=0.84. When the bold data in the same tables are taken, the regression has 48 observations with

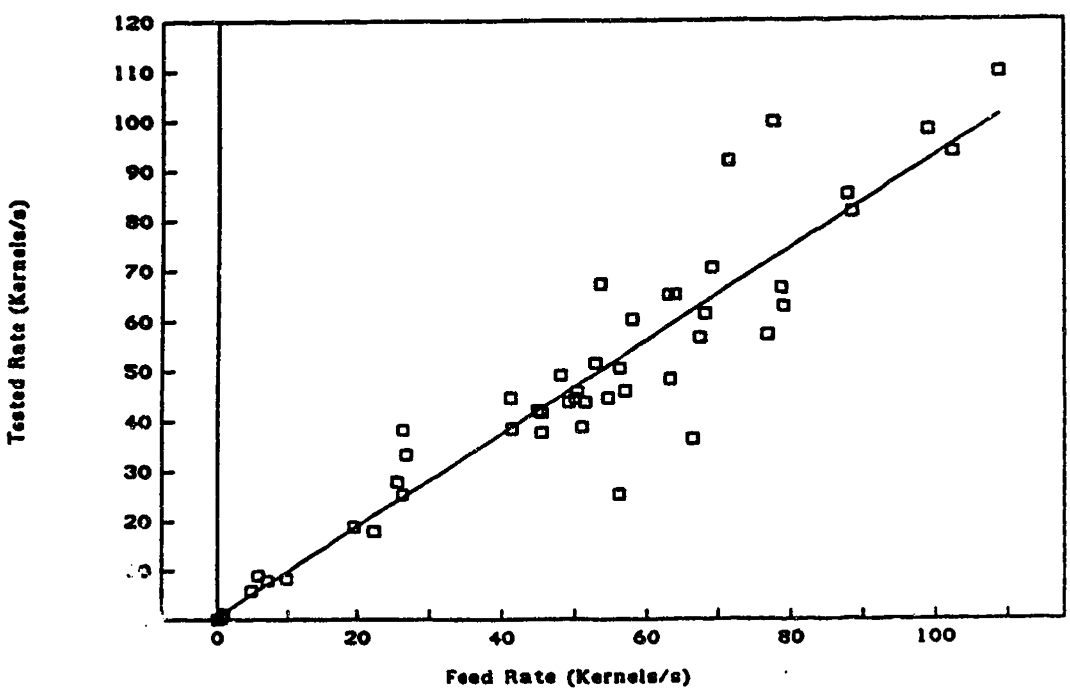
R^2=0.87 and slope=0.92. These results illustrate that adjusting the sensor sensitivity improved the accuracy of the sensors. Figure 9.1 shows the regression lines and the scattered data for sensor simulation calibration.

In a similar manner, straight lines were fitted to the threshing calibration results (tested vs actual) and the results are shown in Figure 9.2. The overall data straight line fit had 53 observations with R^2=0.64 and slope=0.72. These values were not as good as the value obtained in simulation calibration. The fact that MOG was included in the mixture could be the main reason of this low correlation. However, if data (bold type in table 3.2.1 - 3.2.3) which were obtained with final reference voltages were used for the same straight line fit, the regression had 31 observations with R^2=0.83 and slope=1.20. These were much better than the parameters obtained from the overall data.

As mentioned in 7.2.3.2, a calibration coefficient which was the reciprocal of above best fit straight line slope was derived. On the basis of these tests, the calibration coefficient $K_2 = 0.8326$. This could be entered in to the SBC for measurement of losses under similar conditions during the threshing tests.

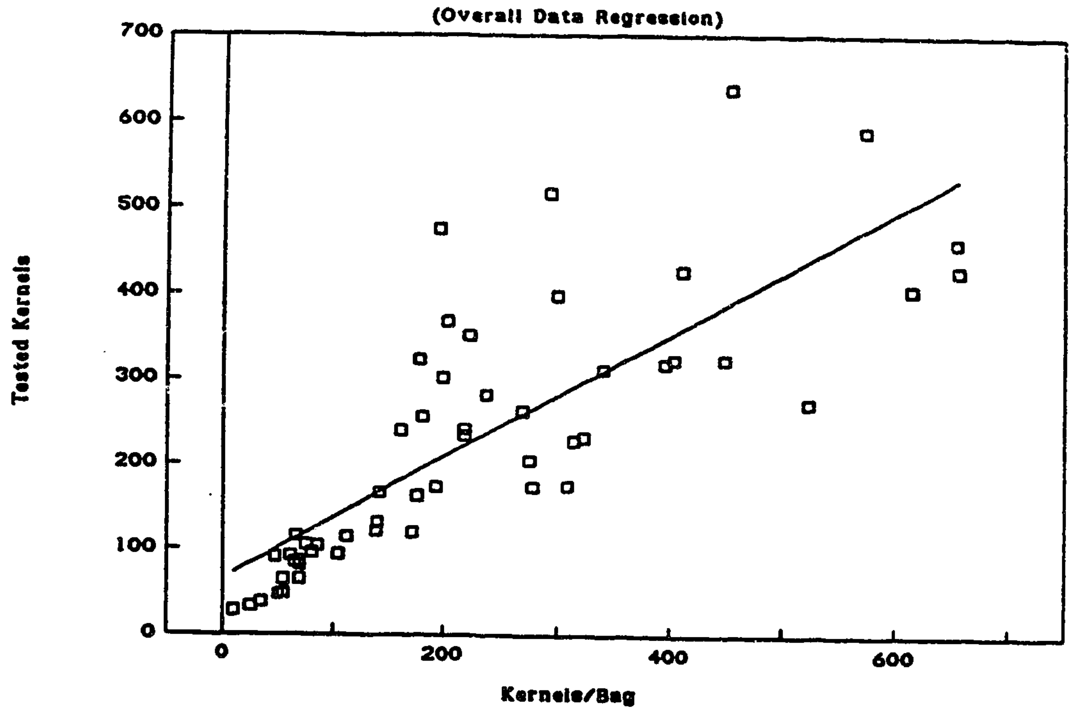


(A) Overall Data

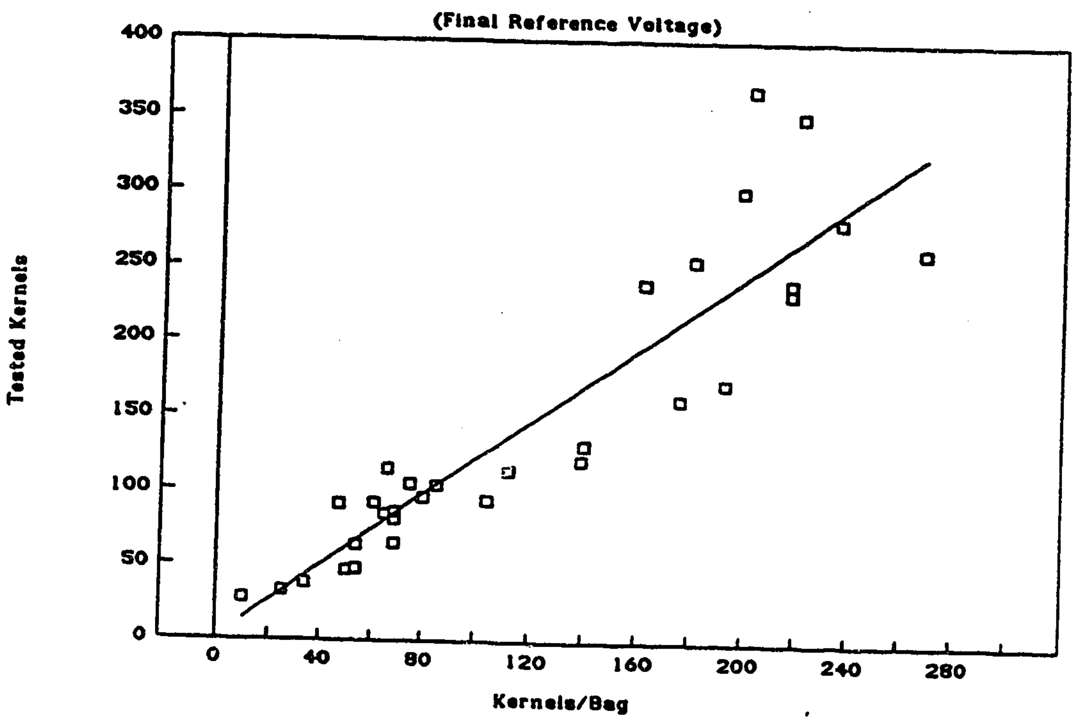


(B) Final Adjustment

Figure 9.1 Sensor Calibration Using Simulation System



(A) Overall Data



(B) Final Reference Voltage

Figure 9.2 Sensor Calibration Using Threshing Equipment

9.2 Threshing Data Analysis

Twelve runs were carried out for the threshing tests. The results are shown in Table 9.1. The data with star symbols indicate that these data were collected in the rotor filling or emptying transient processes. For each run, three such data points were rejected as bad data (usually these values were low) and the remaining samples were considered to be valid. Predicted losses of every samples were summed for each run and the sum was multiplied by the coefficient K ($K = K_1 K_2 C$). These are shown, together with actual losses in Table 9.2. In this Table, the actual and predicted losses are shown as numbers of kernels.

The hypothesis test was based on the same concept used in section 8.2.4. The 't' statistic (see equation 8.3) was 0.4797 with M=1.033, μ=1.0, n=12 and s=0.2383 with 11 degrees of freedom. The null hypothesis was accepted with α=0.05. The 95% confidence interval was between 0.8486 and 1.1514, that was, ±15% for a rotor speed of 800-1000rpm and a feed rate of 6-12t/h.

The best fit straight line of actual and predicted losses is shown in Figure 9.3 and the result shows high correlation and accuracy (R^2=0.92 and slope=1.02). Both predicted and actual losses illustrated that, at high rotor speeds and low feed rates, losses were low and, at low rotor speeds and high feed rates, these losses were high.

Table 9.1 Predicted Grain Loss of Ten Samples.

Sample	Run No. 1 12-1000-1	Run No. 2 6-1000-1	Run No. 3 6- 800-1	Run No. 4 12- 800-1
1	4.4255	1.4534	0.8487	2.2503
2	12.8991	1.3389	1.9342	23.5301
3	9.8913	2.0709	2.4337	21.7999
4	3.8709	2.7547	3.8404	24.7331
5	5.3398	2.2878	4.3337	9.5769
6	4.6752	2.5686	2.4384	17.2664
7	3.0365	2.2659	3.4656	20.5647
8	* 0.1929	* 0.6726	* 0.8874	* 0.9930
9	* 0.1093	* 2.5461	* 0.1409	* 1.5818
10	* 0.1063	* 0.3294	* 0.1227	* 0.1597
Sample	**Run No. 5 12- 800-2**	**Run No. 6 12-1000-2**	**Run No. 7 6- 800-2**	**Run No. 8 6-1000-2**
1	0.8532	2.7170	1.5297	0.4389
2	16.0863	14.9360	1.1075	2.8866
3	11.1840	8.5428	0.7126	4.1035
4	13.9482	6.7226	3.1266	1.6150
5	19.7022	6.9132	1.8555	0.9211
6	24.7018	3.8041	3.5630	2.9672
7	34.8644	4.6301	1.6707	2.7552
8	* 6.1248	* 0.7735	* 2.8434	* 0.7228
9	* 3.3669	* 0.2745	* 1.2387	* 0.1807
10	* 0.1978	* 0.1978	* 0.7829	* 0.2045
Sample	**Run No. 9 12- 800-3**	**Run No.10 6- 800-3**	**Run No.11 12-1000-3**	**Run No.12 6-1000-3**
1	* 0.2670	* 0.0638	1.2470	* 0.4229
2	4.7383	1.8189	7.0505	1.6595
3	5.4946	4.8724	5.0564	1.5890
4	9.3912	1.4556	8.3994	2.5204
5	9.9039	3.4729	4.6980	2.0147
6	10.8050	1.7355	7.1469	2.5865
7	5.4320	1.8975	4.4011	1.3928
8	5.3734	1.8323	* 0.7932	3.0828
9	* 1.0262	* 0.3384	* 1.1015	* 2.3134
10	* 0.1312	* 0.1437	* 0.2806	* 0.1213

* Predicted losses in transient, they were omitted.

Table 9.2 Predicted and Actual Grain Loss of 12 Runs

Run No.	Feed Rate	Rotor Speed	Rep	Actual Loss (g)	Actual Loss (kernels)	Predict Loss (kernels)	M Actual/Predict
2	6	1000	1	59	1639	2056	0.7972
8	6	1000	2	82	2277	2226	1.023
12	6	1000	3	52	1444	2017	0.6972
3	6	800	1	120	3333	2691	1.239
7	6	800	2	74	2055	2075	0.9908
10	6	800	3	71	1972	2383	0.8275
1	12	1000	1	318	8833	6156	1.435
6	12	1000	2	255	7083	6732	1.052
11	12	1000	3	212	5888	5300	1.111
4	12	800	1	511	14194	16699	0.8500
5	12	800	2	575	15972	16923	0.9438
9	12	800	3	368	10222	7132	1.433

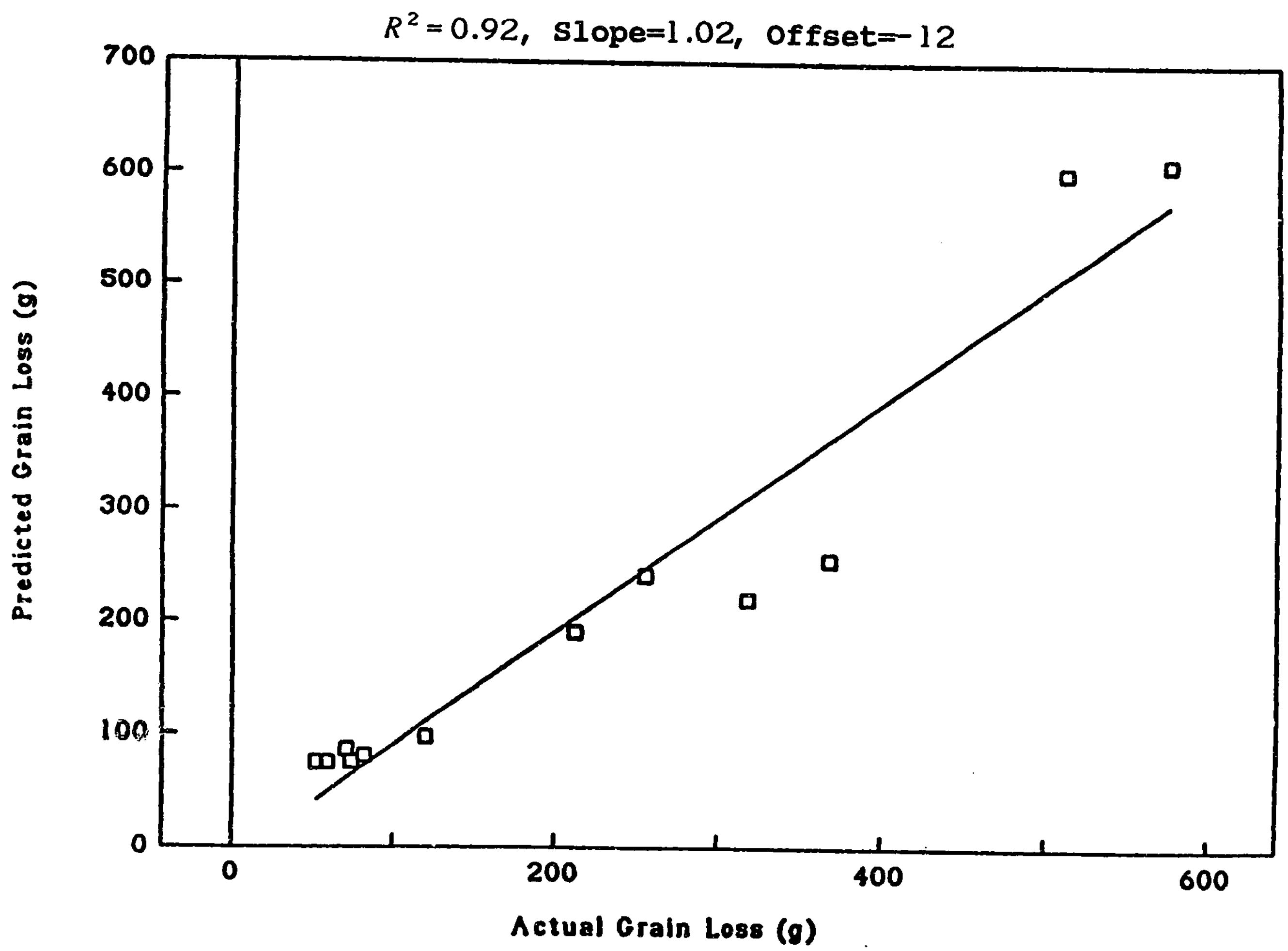


Figure 9.3 The Best Fit Straight Line

A test run was randomly selected from test runs (Run No. 4) and the real-time-calculated parameters of distribution curves were used to draw curves for both arc and axial directions. These curves are shown in Appendix V 1 and 2. The curves show that separation distribution in the axial direction was exponentially decaying, while most separation distribution in the arc direction were convex shaped curves. The three horizontal straight lines were the mean values of the second order polynomial curves in the arc direction. All these real-time-fitted curves confirmed the feasibility of the software design.

10. SUGGESTIONS FOR FURTHER WORK

The accuracy of predicted losses was affected by many factors. The effects of transient processes during threshing runs was one of the significant factors that could be reduced in some ways. The most effective method would be to increase the feed time. To do this, the best way would be to increase the length of the conveyer but the size of the building restricts this option. Appropriate increases in the thickness of the crop on the conveyer coupled with slower conveyer speeds would help.

The sensors should be calibrated before threshing in the field. The calibration coefficient could be obtained after harvesting for a short distance in the field as a calibration run. The ratio of predicted losses to actual losses could be the calibration coefficient. The multi-sensor test system was manipulated as a one-input (actual losses) and one-output (predicted losses) instrument and was calibrated by entering the calibration coefficient as a single number. This was the only way to simplify the calibration and the monitor system could be used in field harvesting.

A straw bag and cleaning equipment were needed for calibration. However, it would also be possible to clean threshed straw by feeding discharged straw into the combine again to clean out the lost grain.

Because of the limited time available in indoor threshing runs, the RTDP was executed after each of the ten samples that had been collected in each run. If this data processing method was used for monitoring the grain loss in field harvesting, the non-uniform, multiphase materials separated in the rotor might cause the separation rate to vary and, thus, the monitor reading would vary too. To provide a stable, readable display, digital averaging could be used to store several samples in memory before using them to determine the distribution curve using least squares regression. The sample data would be scattered on both sides of the curve which would best represent the separation distribution.

Field threshing tests are necessary to verify the performance of the grain loss monitor system in real harvesting conditions.

Since this grain loss monitor was a 16-bit microprocessor-based, real-time data acquisition and processing instrument, system CPU and other interfaces have a lot of potential for further development. The execution time of data acquisition and the RTDP model occupied less than 11% of CPU time. Thus, 89% of CPU data processing ability could be used for further useful information processing. For example: to monitor the engine performance, plot yield maps or even communicate with a host computer to access a harvest expert database system via microprocessor-controlled communication equipment.

11. CONCLUSIONS

Sounding board signal conditioners could determine grain and non-grain impacts and performed with high noise immunity. Grain impact counting accuracy could statistically be within ±10% for a 95% confidence interval.

The assembly language, real-time data processing was able to represent grain separation distribution both around and along the separation rotor. The real time calculation took between 55 and 110ms.

Monitoring actual grain loss from an axial-flow combine in real time by means of a microprocessor-based instrument system has been demonstrated to be feasible. The linear regression of actual and predicted losses showed a high correlation (R^2=0.922), and a slope very close to one (1.018).

The comparison of predicted to actual losses showed a statistically significant correlation at the level of $\alpha = 0.05$ with the data from all the test runs.

The plotted curves, both in the arc direction and in the axial direction, showed that the regression model selection was reasonable. The real time data processing calculation software design provided good estimates of actual grain losses.

Prolonging the sampling and feed times could achieve higher correlations between predicted and actual losses.

Further tests using threshing runs with longer feed times and field tests are needed to evaluate both system hardware and software.

12. REFERENCES

ASAE. 1987. Standard S343.1. Agricultural Engineers Yearbook. Am. Soc. Agric. Eng., St. Joseph, Michigan.

Baker, J. 1988. Personal Communication, Baker Electronic Enterprises. Edmonton, Alberta.

Bjork, A. 1988. Computer modelling of grain separation and grain separation losses for a rotary combine. Paper No. PNR 88-102. Am. Soc. Agric. Eng., St. Joseph, MI.

Blank, L. 1980. Statistical Procedures for Engineering, Management, and Science. McGraw-Hill Book Company. New York.

Clements, A. 1987. Microprocessor System Design 68000 Hardware, Software and Interfacing. PWS Publishers, Boston, Massachusetts.

Cox, S.W.R. Microelectronics in Agriculture and Horticulture Electronics and Computers in Farming. Granada Publishing Limited - Technical Book Division. Frogmore, St. Albans. U.K.

Draper, N.R. and Smith, H. 1981. Applied Regression Analysis. John Wiley & Sons Inc. New York.

Huisman, W. 1983. Optimum Cereal Combine Harvester Operation by means of Automatic Machine and Threshing Speed Control. Doctoral Thesis. Agricultural University, Wageningen, The Netherlands.

Johnson, D.E. 1980. A Handbook of Active Filters. Prentice-Hall, Englewood Cliffs, N.J.

Larson, R. 1987. MicroProcessor-Based System For Measurement of Straw Walker Grain Loss. Master's Thesis. University of Alberta, Edmonton, Alberta.

Larson, R. and Leonard, J. 1988. Digital and Analog Processing of Signals from Grain Impact Sensors. Computing and Electronics in Agriculture.

Liu, C., Leonard, J. and Feddes, J. 1988. Monitoring Flight Activities from A Beehive. Paper No. 88-401 CSAE 1988. Calgary, Canada.

Liu, C. and Leonard, J. 1989. Microprocessor Based Real Time Grain Loss Prediction and Monitoring. Paper No. 160. Proceedings, Beijing 89 ISAE, Beijing, P.R. China.

Liu, Y.C. and Gibson, G.A. 1986. Microprocessor Systems: the 8086/8088 Family Architecture, Programing and Design. Prentice-Hall Inc. Engliwood Cliffs, N.J.

Lunty, J.M. 1986. Measurement of Combine Straw Walker Grain Loss. Master's Thesis. University of Alberta, Edmonton, Alberta.

Norton, P. 1985. Programmer's Guide to the IBM PC. Microsoft Press. Redmond, Washengton.

PAMI 1978. Evaluation Report No. E3078A. International 1460 Self-Propelled Combine. Prairie Agricultural Machinery Institute, Humboldt, Saskachwan.

Wrubleski, P. 1978. PAMI Combine Report Interpretation. Prairie Agricultural Machinery Institute, Humboldt, Saskatchewan.

Reed, W.B. 1978. A review of Monitoring Devices for Combines. Proceedings of the International Grain and Forage Harvesting Conference. Iowa State University, Ames, Iowa. September, 1977. Publication No. 1-78, Am. Soc. Agric. Eng. St. Joseph, MI.

Steele, R.G.D. and Torrie, J.H. 1980. Principles and Procedures of Statistics. A Biometric Approach. McGraw-Hill Book Company. New York.

The Green Book, 1983. Grain and Seed Crop Handling Equipment. Directory Publication Ltd. London, England.

Wang, G. 1984. A Study of Separation Characteristics of a Rotary Combine. Master's Thesis. University of Saskatchewan, Saskatoon, Saskatchewan.

Wang, C., Yan, D. and Liu, D. 1984. The Principle and Application of Microcomputer. Science and Technology Publishing House. Hubei, P. R. China.

Williams, G. 1978. Computational Linear Algebra with Models Allyn and Bacon, Inc. Boston, USA.

Williams, G.B. 1988. Keep Your PC Healthy. Byte IBM Special Edition. Fall 1988. A McGraw-Hill Publication.

13. APPENDIX

Appendix I.

Table 1. Basic Structure and Main Functions of the Intelligent Instrumentation System

Name of components	**Main Functions (Hardware)**
CPU (8088)	Central Processing Unit
EPROM	Monitor Programs for control and data processing, look-up table, etc. are stored in EPROM.
RAM	Interrupt array, stack, input and output data, intermediate and result of calculation data are stored in RAM.
I/O Parallel	Connected to keyboard, display unit, DIP switches or sensors etc.
I/O Serial (RS-232)	Up (or down) load data or some other information to (or from) the host computer or peripheral equipment which has RS-232 interface.
I/O A/D	Data acquisition interface connected to sensors and signal conditioner outputs.
Real Time Clock	Supplies the time base for graphics or for current time display.
Programmable Interrupt Controller	Accept interrupt requirements from programmable interfaces. Interrupt priority management.
Timer and Event Counter	Timer: To mark intervals of time for both the processor and external devices (e.g. baud rate generator). Counter: To count external events and make the counters available to the processor.
Peripheral Equipment	Display Unit: Display keyboard command, input coefficients, data etc. Keyboard: Entry of commands, coefficients, real time clock set, etc. Printer: Hard copy output. Sensors: Any transducers which could convert electrical quantities into 0-5V DC signals or TTL pulses could be used.

Table 1. (continued)

	(Software)
Keyboard Monitor	Keyboard scanning and command string (ASCII) process.
Display and Printer Control	Control Program makes it possible for the display unit (LCD) and the portable printer to work in Hand Shake Mode.
Data Acquisition and Data Processing	1. Sensor scanning interval. 2. Convert sensor output into engineering units. 3. Arithmetic transcendental and polynomial functions.

Table 2. Instruction Execution Time in Sampling Gaps (Liu and Gibson, 1986)

Instruction	Operation	Cycles	Instruction Number	Total Cycles
PUSH	push register into stack	11	6	66
POP	pop register off stack	8	6	48
MOV	accumulater to memory	10	9	90
MOV	immediate to register	4	30	120
MOV	register to register	2	38	76
OUT	output via fixed port	10	27	270
IN	input via fixed pcrt	10	18	180
NOT	logic negative	3	9	27
LEA	load effective address	8	1	8
CALL	intrasegment direct	19	2	38
RET	procedure return	8	2	16
NOP	null operation	3	8	24
Total			156	963

Table 3.1.1 Sensor Calibration Using Simulation System

Sensor 1				Sensor 2				Sensor 3			
Vr (V)	Actual Number	Tested Number	Err (%)	Vr (V)	Actual Number	Tested Number	Err (%)	Vr (V)	Actual Number	Tested Number	Err (%)
0.4	829	772	-6.88	0.3	836	1038	24.2	0.5	1009	711	-29.5
0.4	728	761	4.53	0.3	711	939	32.1	0.5	765	587	-23.3
0.4	25	25	0.00	0.3	535	744	39.1	0.5	165	123	-25.5
0.4	37	36	-2.70	0.3	75	107	42.7	0.5	30	20	-33.3
0.4	430	459	6.74	0.3	244	385	57.8	0.5	537	463	-13.8
0.4	245	270	10.2	0.3	81	138	70.3	0.5	394	345	-12.4
0.4	67	75	11.9	0.3	159	245	54.0	0.5	163	147	-9.82
0.4	804	740	-7.96	0.3	1060	1228	15.8	0.5	827	651	-25.3
0.389	504	458	-9.13	0.37	528	515	-2.46	0.5	115	117	1.74
0.389	692	704	4.73	0.37	666	365	-45.2	0.5	611	607	-0.65
0.389	634	484	-23.6	0.37	562	254	-54.8	0.382	317	240	-24.3
0.389	454	417	-8.15	0.37	514	438	-14.8	0.37	6	3	-50.0
0.389	97	83	-14.4	0.37	885	815	-7.90	0.37	580	602	3.79
0.389	563	505	-10.3	0.37	501	445	-11.2	0.37	570	460	-19.3
0.389	787	665	-15.5	0.37	1024	936	-8.59	0.37	879	850	-3.30
0.389	546	446	-18.2	0.37	675	566	-16.1	0.37	641	652	1.72
0.389	492	438	-11.0	0.37	510	388	-23.9	0.37	412	446	8.25
0.389	449	421	-6.23	0.37	454	377	-17.0	0.37	481	491	2.08

Table 3.1.2 Sensor Calibration Using Simulation System

Sensor 4				Sensor 5				Sensor 6			
Vr (V)	Actual Number	Tested Number	Err (%)	Vr (V)	Actual Number	Tested Number	Err (%)	Vr (V)	Actual Number	Tested Number	Err (%)
0.4	115	117	1.74	0.43	306	389	27.1	0.3	785	546	-30.4
0.4	113	87	-23.0	0.43	414	423	4.34	0.3	756	495	-34.5
0.4	686	567	17.3	0.43	113	99	-12.4	0.3	573	581	1.40
0.4	445	460	3.37	0.43	445	460	3.37	0.3	546	498	-8.79
0.4	684	495	-27.6	0.43	498	562	12.9	0.3	570	505	-11.4
0.4	842	597	-29.1	0.43	560	658	17.5	0.3	912	710	-22.1
0.4	459	338	-26.4	0.43	166	196	18.1	0.3	931	693	-25.6
0.4	196	165	-15.8	0.43	119	125	5.04	0.3	518	491	-5.21
0.4	531	467	-12.1	0.43	166	212	27.7	0.3	140	119	-0.15
0.4	144	123	-14.6	0.43	354	404	14.1	0.3	337	320	-5.04

Table 3.1.3 Sensor Calibration Using Simulation System

Sensor 7				Sensor 8				Sensor 9			
Vr (V)	Actual Number	Tested Number	Err (%)	Vr (V)	Actual Number	Tested Number	Err (%)	Vr (V)	Actual Number	Tested Number	Err (%)
0.353	616	369	-40.0	0.34	65	31	-52.3	0.375	487	367	-24.6
0.353	659	402	-40.0	0.34	713	395	-44.6	0.375	535	448	-16.3
0.353	623	431	-30.8	0.34	724	417	-42.4	0.375	520	382	-26.6
0.33	643	356	-8.8	0.32	651	372	-41.0	0.320	322	279	-13.4
0.33	577	344	-11.4	0.32	554	368	-33.6	0.320	515	438	-15.0
0.30	563	377	-22.1	0.32	1170	668	-42.9	0.320	565	440	-22.1
0.30	734	421	-42.6	0.30	768	571	-27.4	0.307	790	628	-20.5
0.30	336	175	-47.9	0.30	223	180	-19.3	0.307	414	385	7.00
0.30	270	182	-32.6	0.30	195	189	-3.08	0.307	262	254	-3.05
0.30	294	241	-18.0	0.30	255	279	9.41	0.307	267	334	25.1
0.30	929	902	-2.90	0.30	990	981	0.909	0.307	262	383	46.2
0.30	1007	1028	2.1	0.30	1088	1096	0.740	0.307	715	919	28.5
0.28	682	613	-10.1	0.30	631	651	3.17	0.37	776	997	28.5
0.28	48	58	20.8	0.30	58	89	53.4	0.40	535	672	25.6
0.28	0	0	0	0.30	71	79	11.3	0.41	9	11	22.2

Table 3.2.1 Sensor Calibration Using Threshing Equipment

Sensor 1				Sensor 2				Sensor 3			
Vr (V)	Actual Number	Tested Number	Err (%)	Vr (V)	Actual Number	Tested Number	Err (%)	Vr (V)	Actual Number	Tested Number	Err (%)
0.35	278	173	-37.7	0.3	613	405	-33.9	0.3	655	427	-34.8
0.29	275	204	-25.8	0.25	195	476	-31.5	0.3	340	311	-8.53
0.29	395	318	-19.5	0.25	523	271	-48.2	0.196	653	460	-29.6
0.2	218	240	10.1	0.198	572	590	31.5	0.196	203	368	81.3
0.2	218	233	6.88	0.198	452	638	41.2	0.160	292	517	77.1
0.2	162	239	47.5	0.198	410	427	4.15	0.20	299	398	33.1

Table 3.2.2 Sensor Calibration Using Threshing Equipment

Sensor 4				Sensor 5				Senscr 6			
Vr (V)	Actual Number	Tested Number	Err (%)	Vr (V)	Actual Number	Tested Number	Err (%)	Vr (V)	Actual Number	Tested Number	Err (%)
0.35	172	119	-30.8	0.38	315	227	-27.9	0.23	448	323	-27.9
0.31	139	121	-12.9	0.30	309	174	-43.7	0.269	178	323	81.5
0.31	176	162	-7.95	0.30	323	231	-28.5	0.269	403	323	19.9
0.31	61	92	50.8	0.215	193	173	-10.4	0.222	269	262	-2.60
0.31	75	105	40.0	0.215	222	351	58.1	0.222	237	281	18.6
0.31	80	96	20.0	0.215	199	301	51.3	0.222	181	255	40.9

Table 3.2.3 Sensor Calibration Using Threshing Equipment

Sensor 7				Sensor 8				Sensor 9			
Vr (V)	Actual Number	Tested Number	Err (%)	Vr (V)	Actual Number	Tested Number	Err (%)	Vr (V)	Actual Number	Tested Number	Err (%)
0.308	54	48	-11.1	0.286	No Data	362		0.450	142	166	16.9
0.308	69	65	-5.80	0.286	112	114	1.79	0.462	85	104	22.4
0.308	69	86	24.6	0.286	140	131	-6.43	0.462	104	94	-9.62
0.308	25	33	32.0	0.286	47	91	93.6	0.462	69	81	17.4
0.308	34	38	11.8	0.286	65	85	30.8	0.462	66	115	74.2
0.308	10	28	1.80	0.286	54	64	18.5	0.462	50	47	-6.00

Table 3.3 Statistics of Actual and Tested Grain Impact Counts

Simulation Overall				After Calibration (Simulation)			
Const	St. Err.	Slope	R^2	Const	St. Err.	Slope	R^2
25.0	0.0364	0.837	0.806	0.648	0.517	0.923	0.874
Threshing Overall				After Calibration (Threshing)			
Const	St. Err.	Slope	R^2	Const	St. Err.	Slope	R^2
65.4	0.0750	0.716	0.641	2.45	0.102	1.20	0.826

Table 4. Straw Moisture Contents

Run No.	Wet Straw (g)	Dry Straw (g)	Moisture Contain (Wet Base %)
1	118	100	15.3
2	124	107	13.7
3	112	97	13.4
4	131	112	14.5
5	145	123	15.2
6	107	89	16.8
7	127	110	13.4
8	124	112	9.7
9	147	127	13.6
10	113	97	14.2
11	134	112	16.4
12	129	111	14.0

Moistuer Containt mean value wet base: 14.2 % .

Toshiba Microwave was used for this test. The working selection was defrozen. Drying time was 20 minutes.

Table 5. Grain Moisture Contents

Run No.	Test Time (Min)	Heating set	Moisture Contain (Wet Base %)
1	60	100	11.0
2	30	120	11.8
3	35	110	11.6
4	30	120	12.6
5	30	120	12.8
6	30	120	11.8
7	30	120	11.1
8	30	120	12.0
9	30	120	11.2
10	30	120	13.0
11	30	120	11.4
12	30	120	11.0

Moisture contain mean value wet base: 11.9 %.

CENCO Moisture Balance was used for the test (Central Scientific Company, CENCO NOS 026680 - 001) .

Table 6. Barley Sample 1000 Kernel Weights

Run No.	Samples	Weight/1000 Kernels (g)
1 - 4	1	35.77
1 - 4	2	34.83
5 - 8	3	38.52
5 - 8	4	33.55

Mean value of 1000 kernel weight is 36 g .

Table 7. The Material Weights and G/MOG

Run No.	Crop (Kg)	No. of Bundles	Straw (Kg)	Chaff (Kg)	MOG (Kg)	Grain (Kg)	G/MOG
1	80.0	29	27.9	7.03	34.9	42.0	1.20
2	39.0	12	13.4	3.86	17.3	19.7	1.14
3	40.0	13	13.8	3.63	17.4	21.1	1.21
4	80.0	22	30.2	6.12	36.3	41.5	1.14
5	80.0	23	31.1	5.67	36.8	41.3	1.12
6	80.0	21	28.8	7.26	36.1	41.6	1.15
7	39.9	11	14.3	4.31	18.6	21.8	1.17
8	42.5	11	13.8	4.99	18.8	23.6	1.26
9	81.3	22	28.8	7.03	35.8	43.3	1.21
10	41.3	7	14.7	4.99	19.7	21.1	1.07
11	80.5	21	27.7	6.80	34.5	44.0	1.28
12	39.0	10	12.2	4.08	16.3	22.5	1.38

Appendix II.

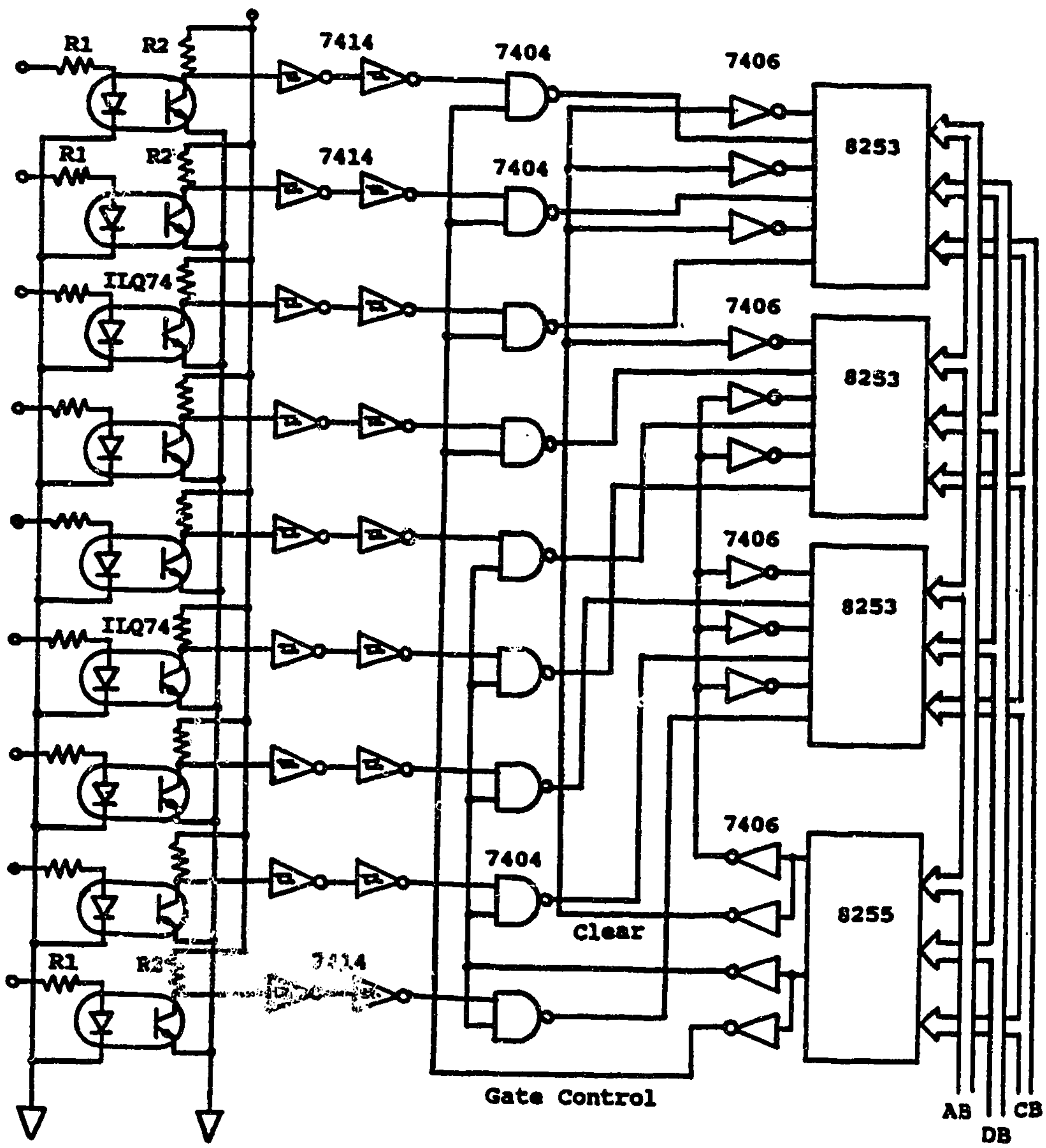
1. Memory Map

IC	Function	Address
SRM2264	INTERRUPTS ARRAY (1024 Bytes)	0000H 0400H
	DATA STORAGE AREA	0401H 1FFFH
MK48T02	1776 Bytes RAM in Real Time Clock	2000H 26F0H
	STACK (256 Bytes)	26F1H 27F0H
	Reserved	27F1H 27F7H
	RTCTRL	27F8H
	SECOND	27F9H
	MINUTE	27FAH
	HOUR	27FBH
	DAY	27FCH
	DATE	27FDH
	MONTH	27FEH
	YEAR	27FFH
	No Physical Memory	2800H BFFFH
2764	Look-up Table (EPROM 2)	C000H DFFFH
2764	Data Acquisition and Processing Modules (EPROM 1)	E000H FFEFH
	Boot-up (Reset)	FFF0H
		FFFFH

2. I/O Map

IC	Function	Address
8255-1	I/O A #0 I/O B #0 I/O C #0 I/O Control #0	00H 01H 02H 03H
8251	*UART-Data UART-Control	10H 11H
8253-1	Timer #0 Timer #1 Timer #2 Timer Mode	20H 21H 22H 23H
8259	Interrupt Controller Initialization Interrupt Controller Operation	30H 31H
8253-2	Counter #1 Counter #2 Counter #3 Counter Mode #1	40H 41H 42H 43H
8255-2	I/O A #1 I/O B #1 I/O C #1 I/O Control #1	50H 51H 52H 53H
8253-3	Counter #4 Counter #5 Counter #6 Counter Mode #2	60H 61H 62H 63H
8253-4	Counter #7 Counter #8 Counter #9 Counter Mode #3	70H 71H 72H 73H
8255-3	I/O A #2 I/O B #2 I/O C #2 I/O Control #2	80H 81H 82H 83H

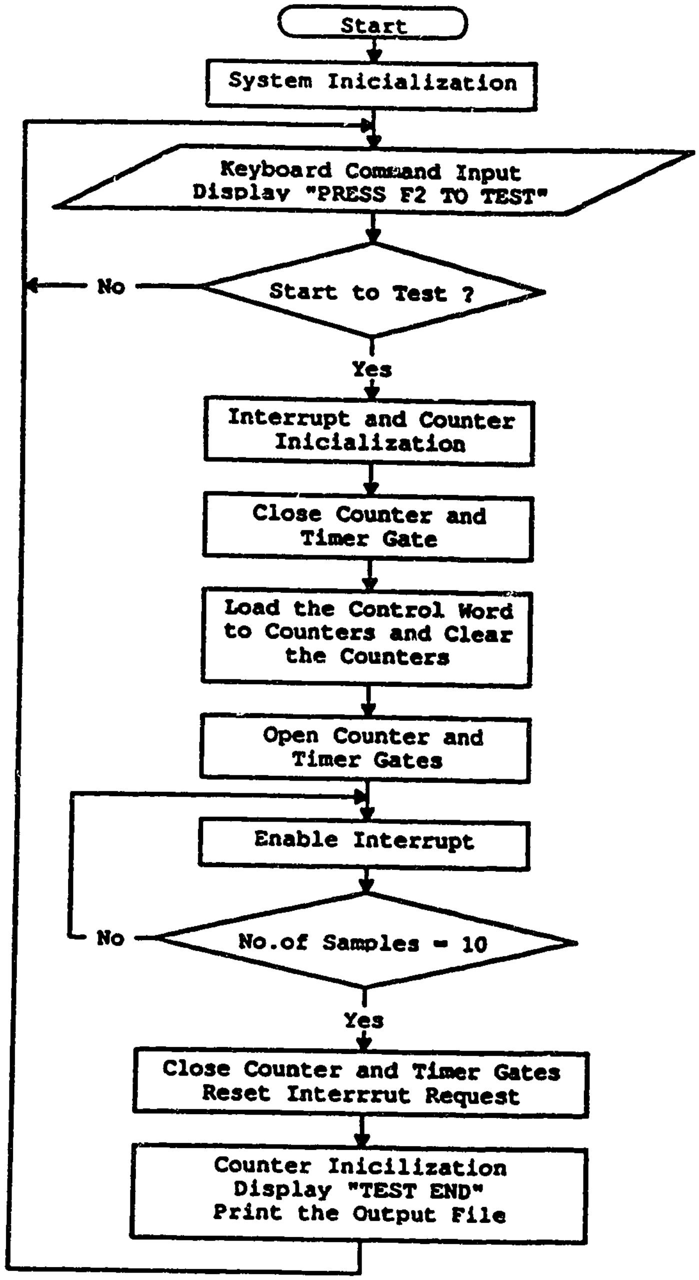
*UART: Universal Asynchronous Receiver and Transmitter



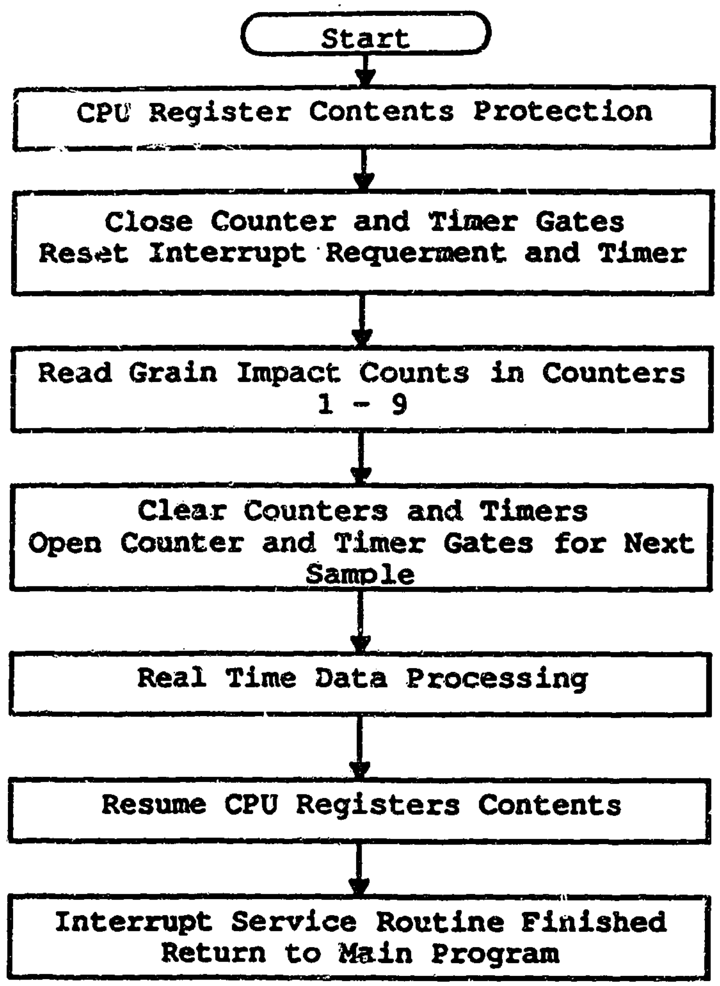
3. Nine Channel Signaı ıterface.

Appendix III.

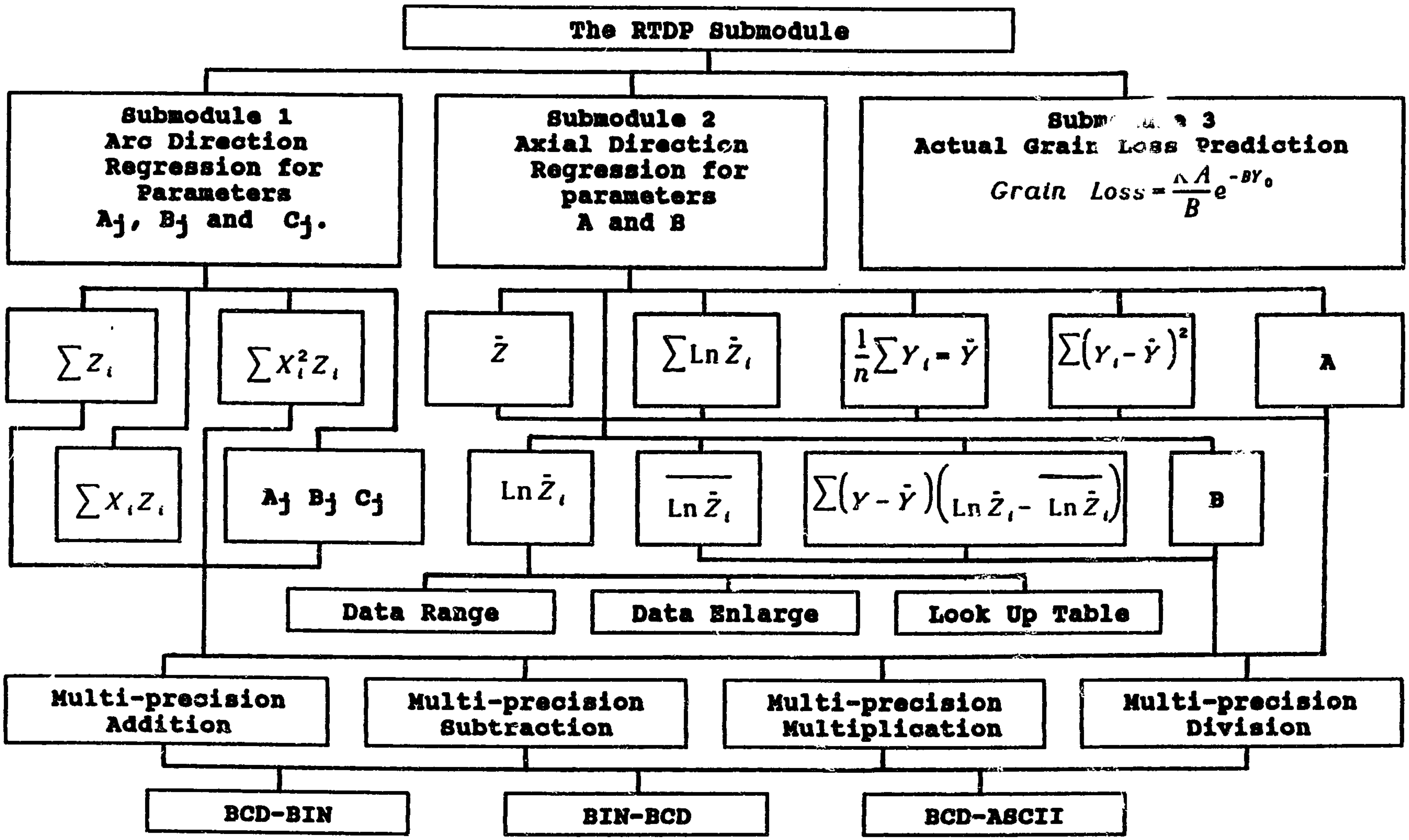
1. Main Program Flowchart
2. Interrupt Service Routine Flowchart
3. The Hierarchical Diagram for the RTDP
4. Look-up Table Flowchart



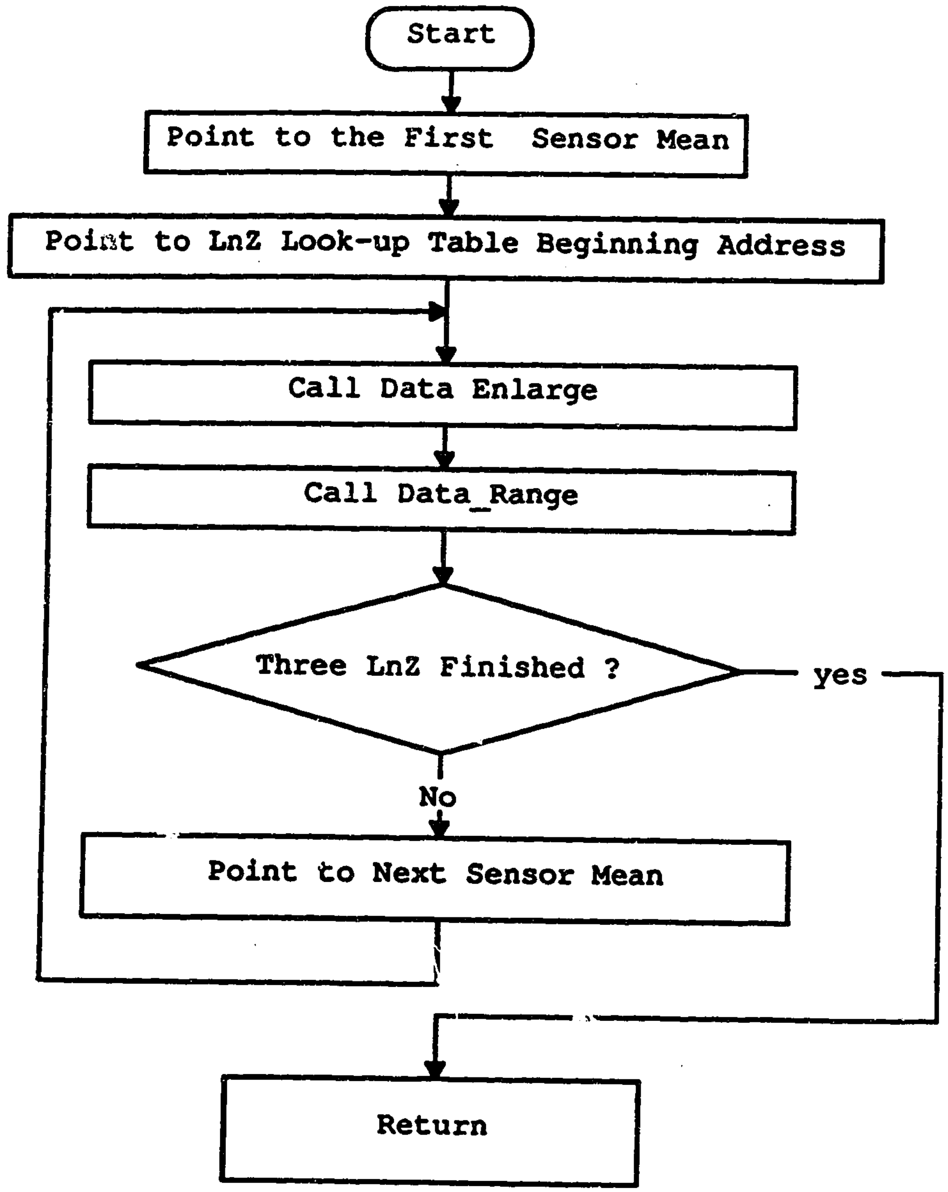
1. Main Program



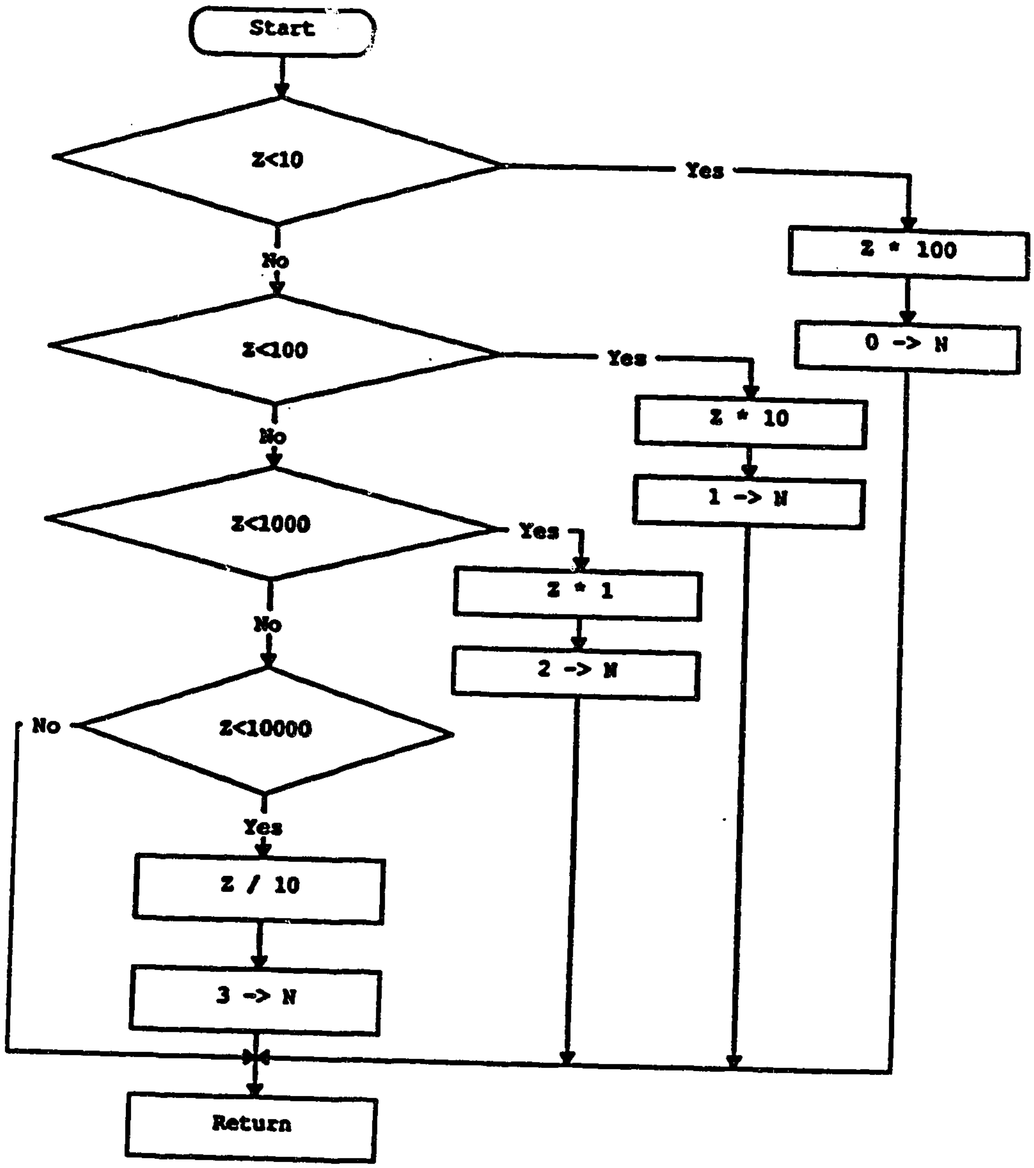
2. Interrupt Sevice Routine



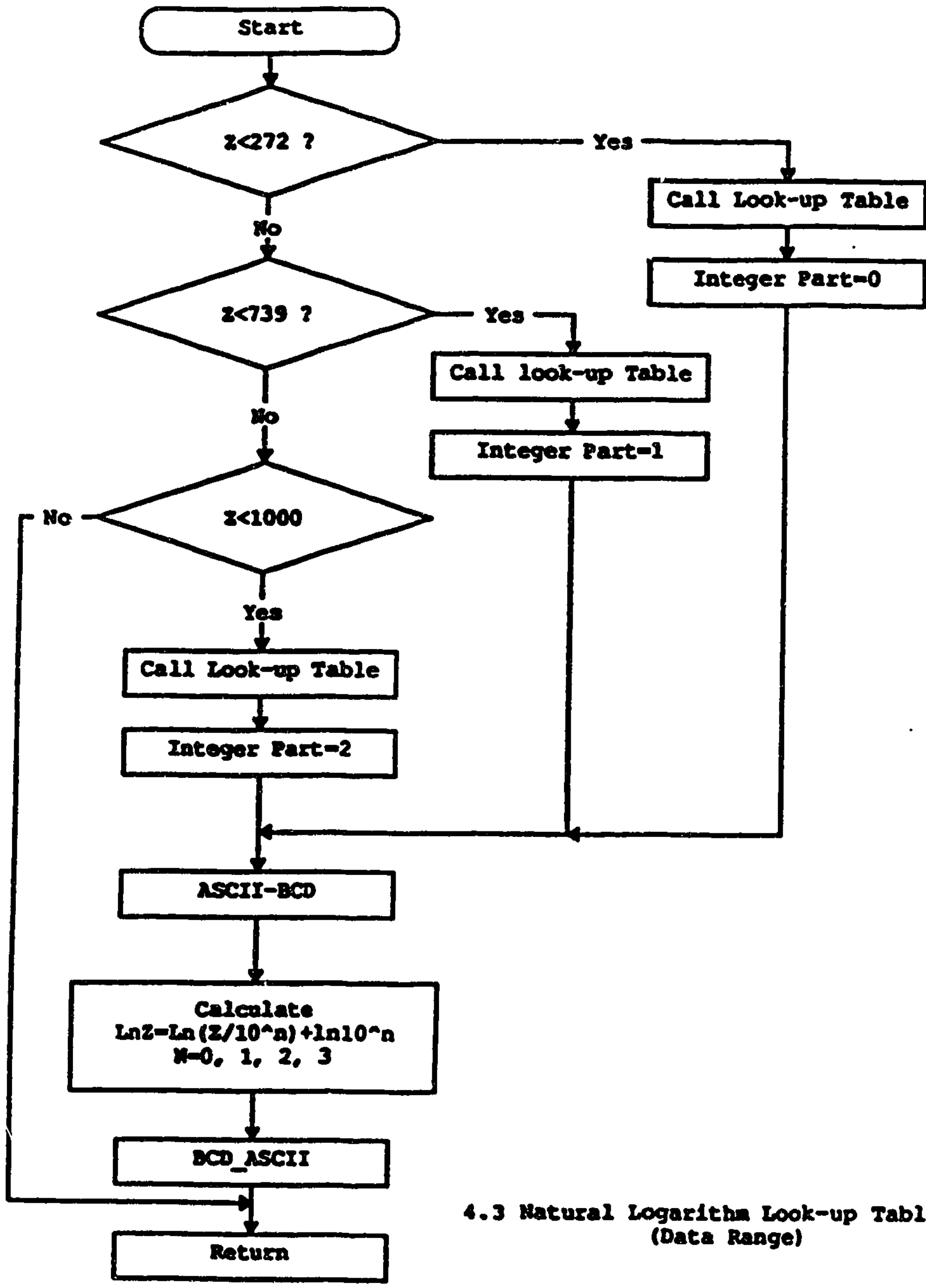
3. The Hierarchical Diagram for the RTDP.



4.1 Natural Logarithm Look-up Table (Main)

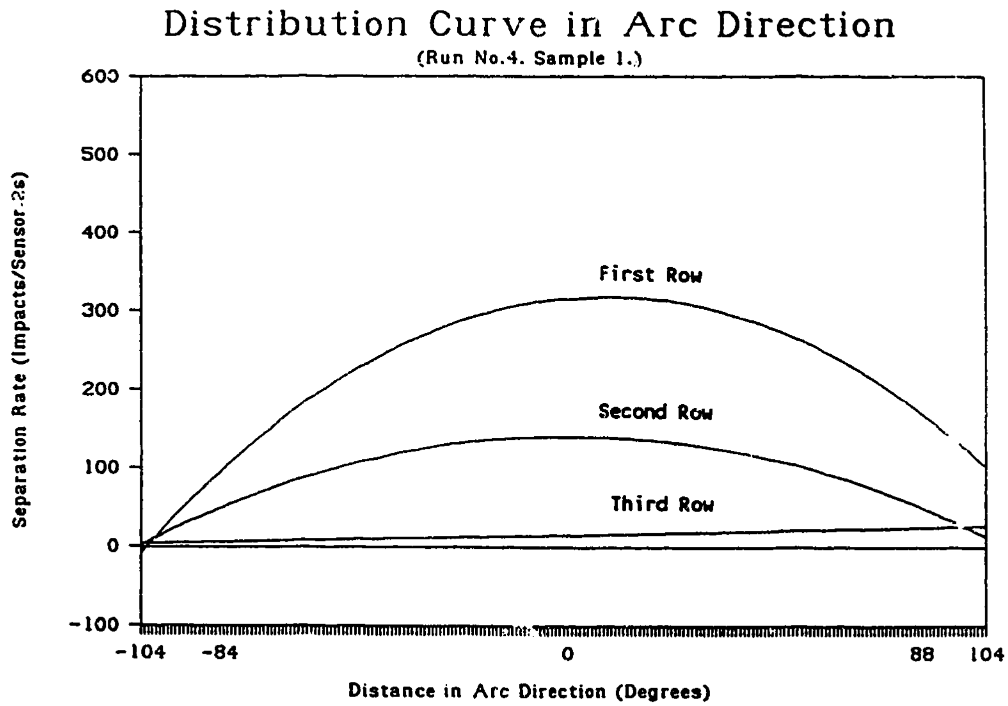


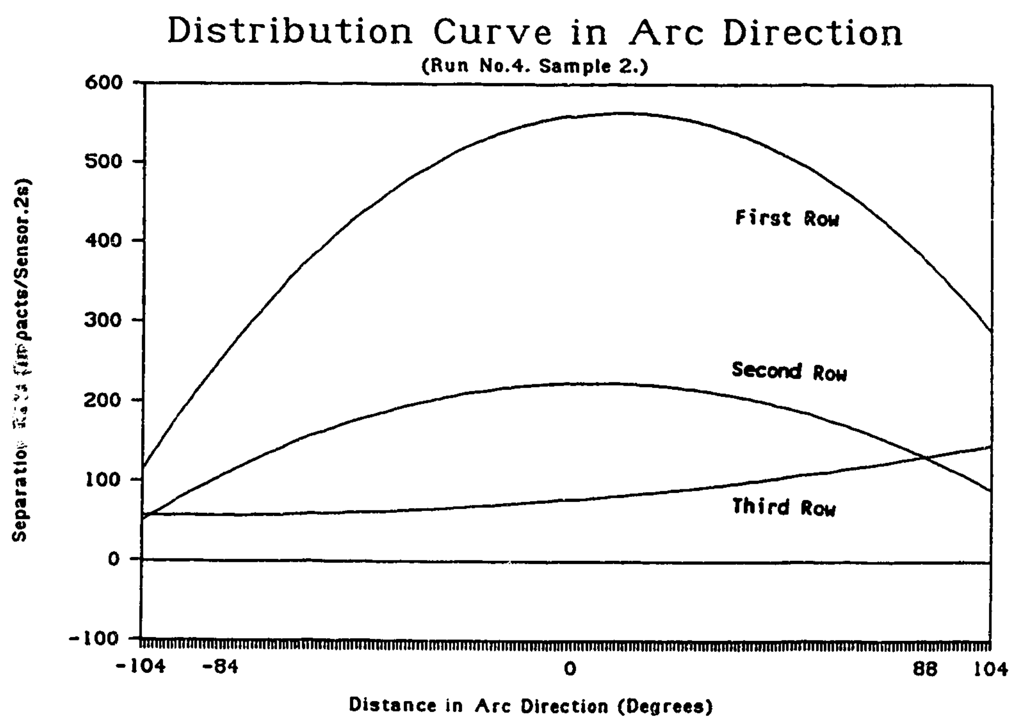
4.2 Natural Logarithm Look-up Table
(Data Enlarge)

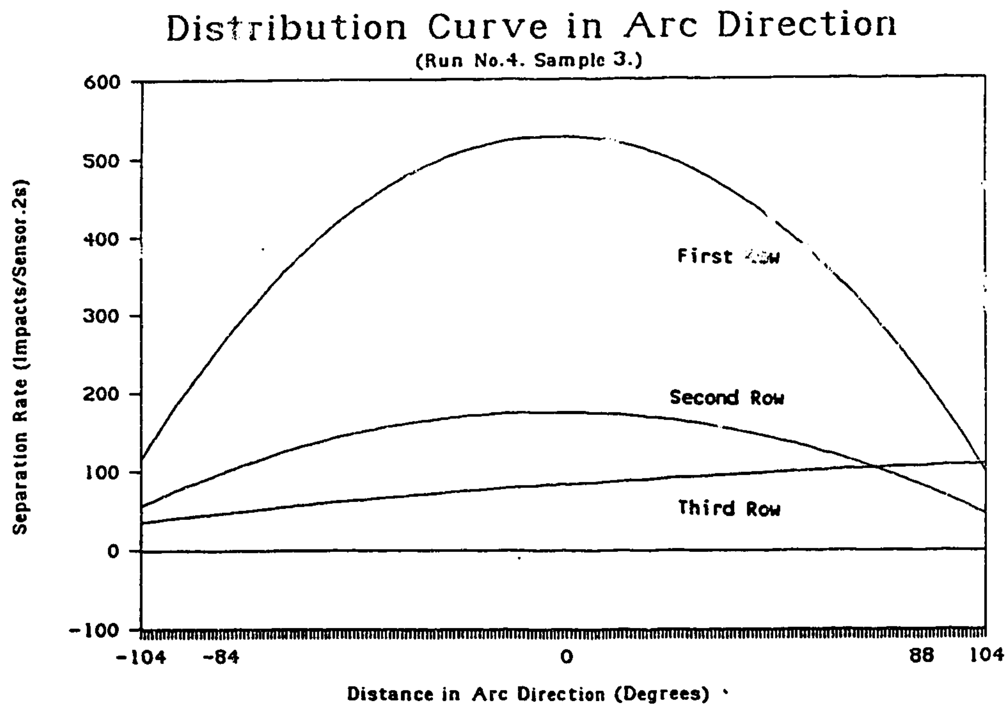


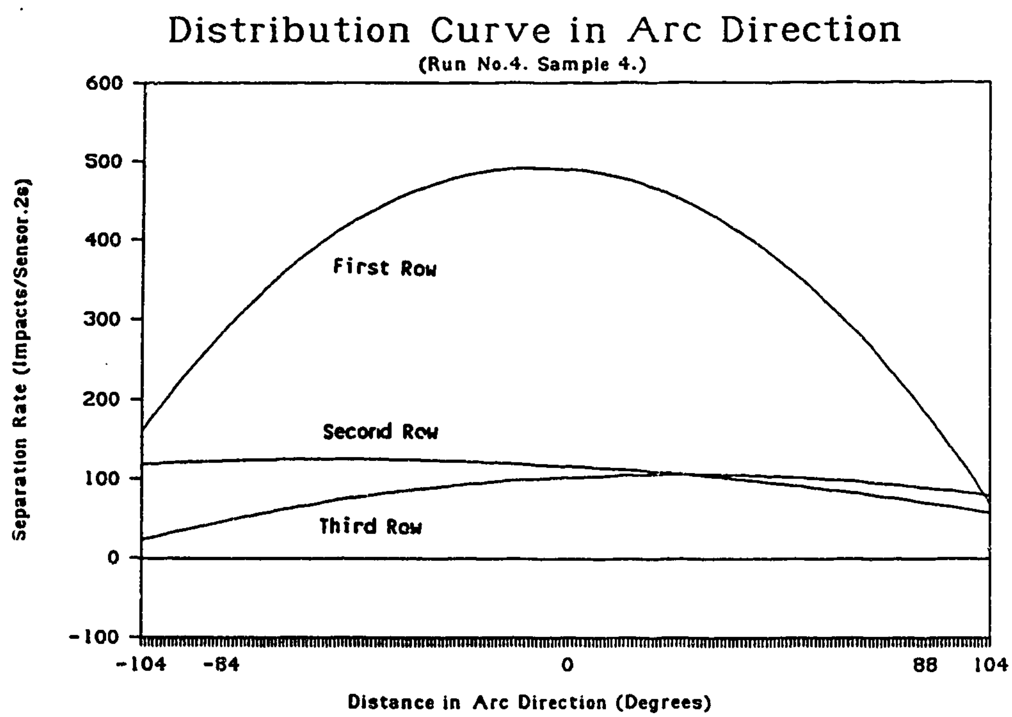
4.3 Natural Logarithm Look-up Table
(Data Range)

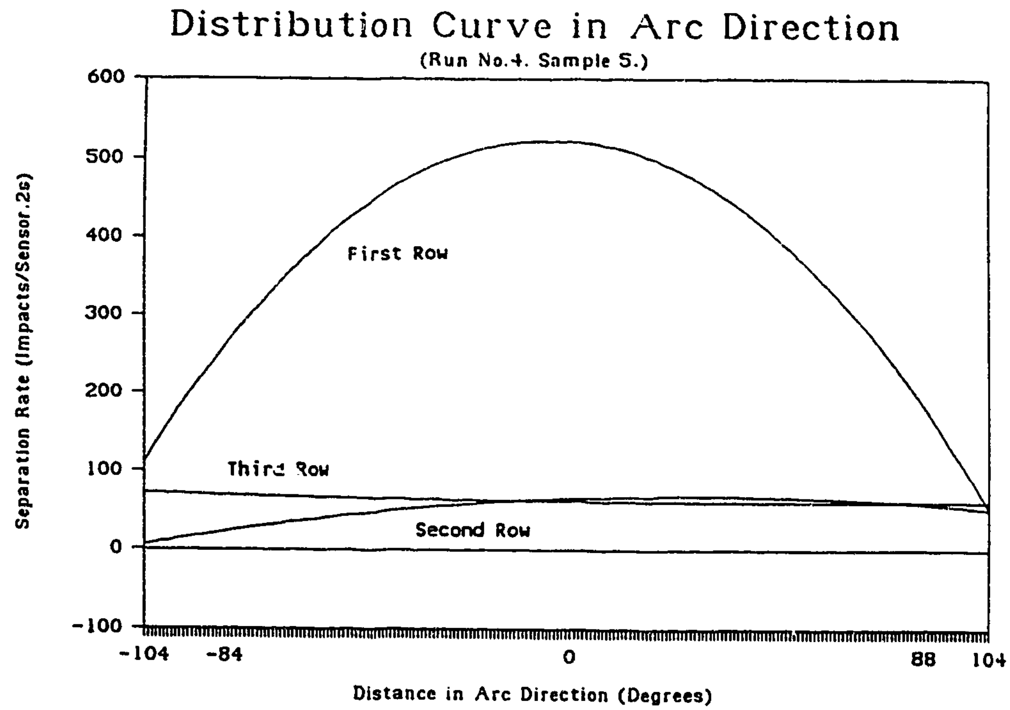
Appendix IV. 1. The Best Fit Curves in Arc Direction

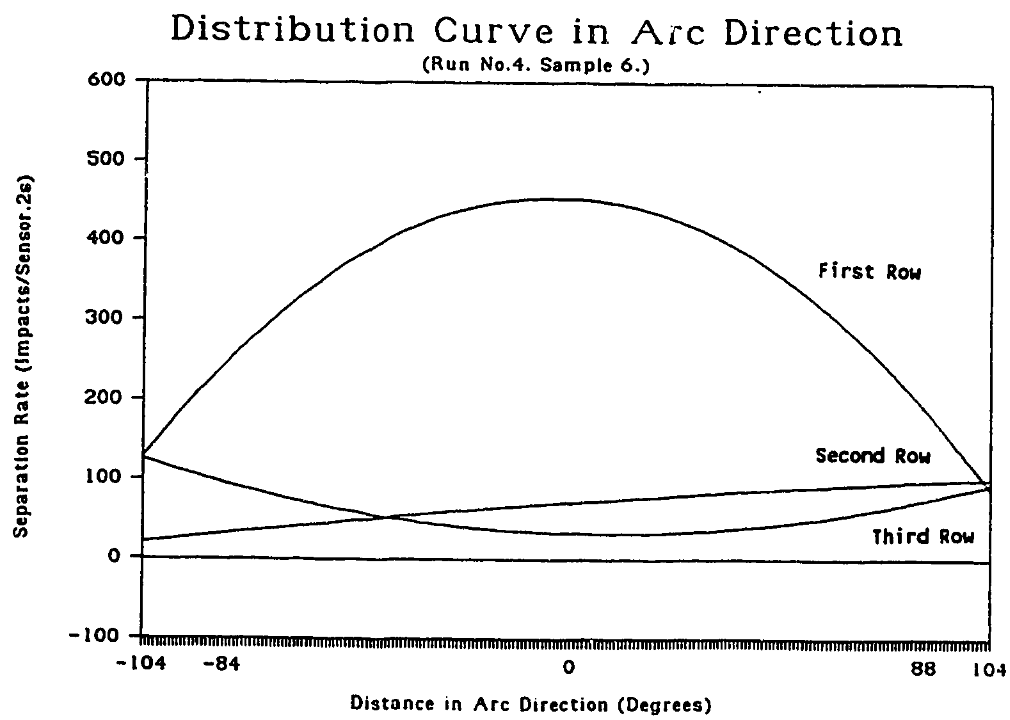


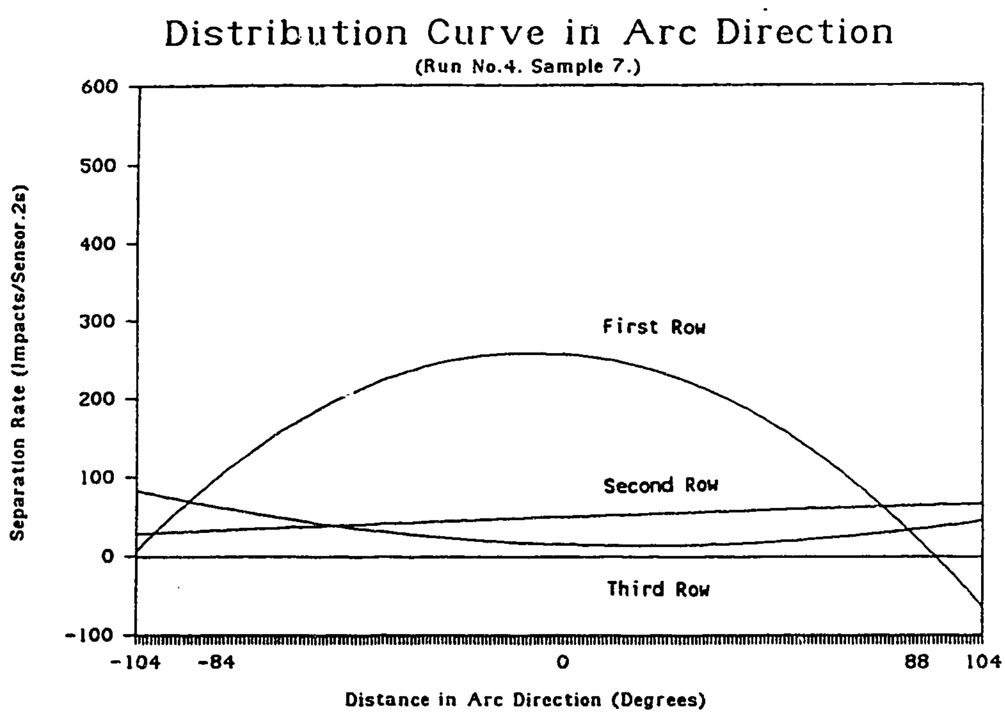



Distribution Curve in Arc Direction
(Run No.4. Sample 3.)
Separation Rate (Impacts/Sensor.2s)
600
500
400
300
200
100
0
-100
First Row
Second Row
Third Row
-104
-84
0
88
104
Distance in Arc Direction (Degrees)

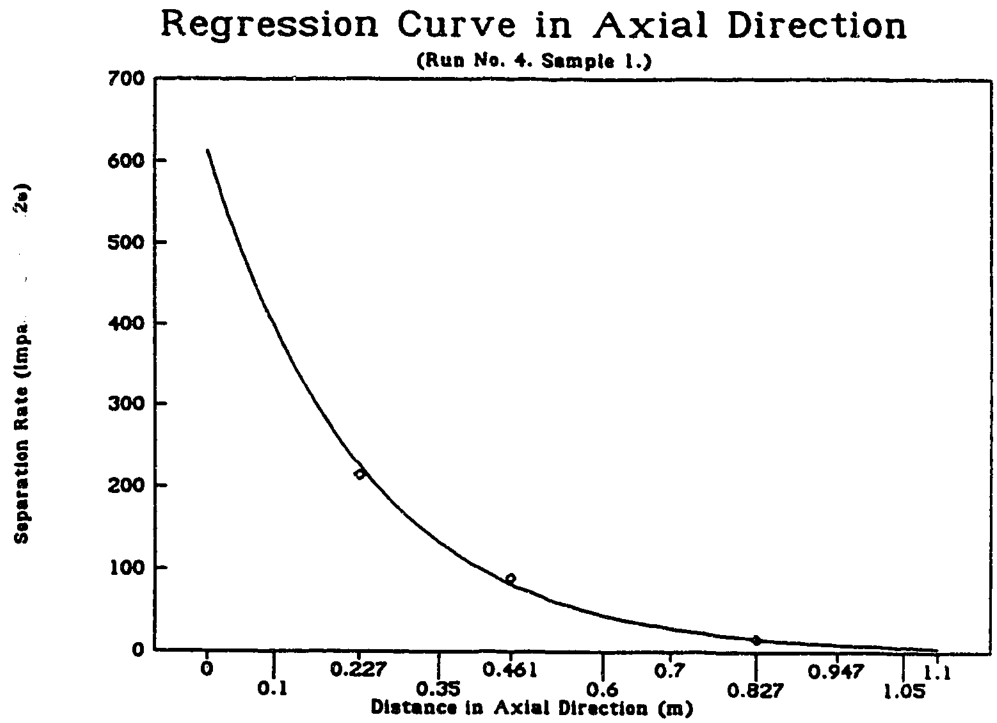

Distribution Curve in Arc Direction
(Run No.4. Sample 4.)
Separation Rate (Impacts/Sensor.2s)
600
500
400
300
200
100
0
-100
First Row
Second Row
Third Row
-104
-84
0
88
104
Distance in Arc Direction (Degrees)

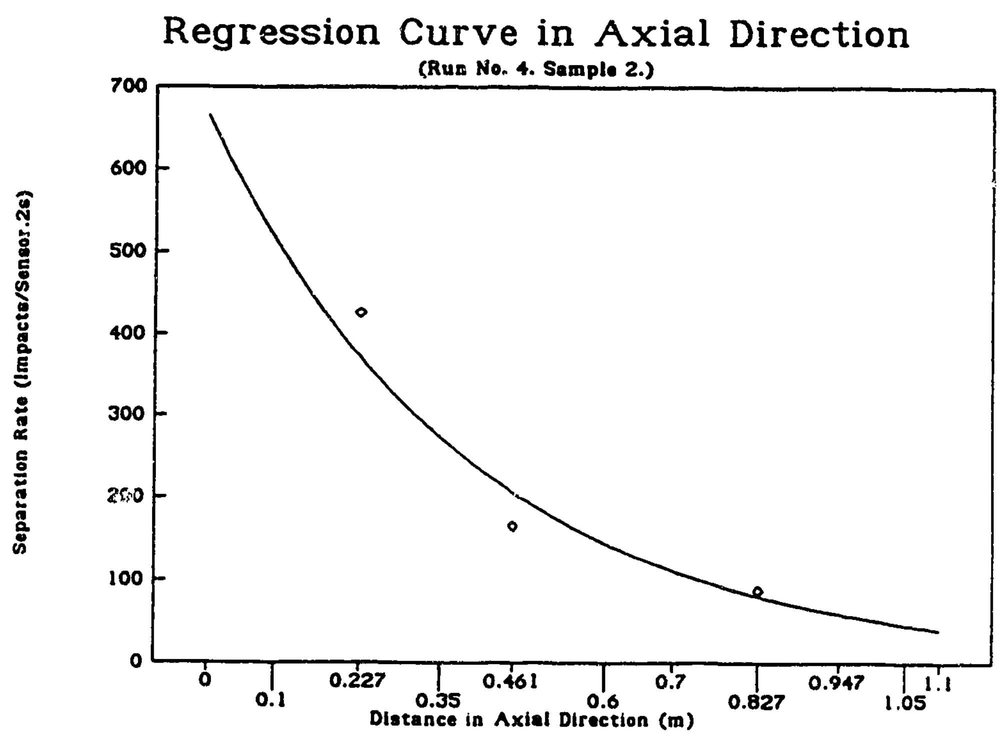

Distribution Curve in Arc Direction
(Run No.4. Sample 5.)
Separation Rate (Impacts/Sensor.2s)
600
500
400
300
200
100
0
-100
First Row
Third Row
Second Row
-104
-84
0
88
104
Distance in Arc Direction (Degrees)


Distribution Curve in Arc Direction
(Run No.4. Sample 6.)
Separation Rate (Impacts/Sensor.2s)
600
500
400
300
200
100
0
-100
First Row
Second Row
Third Row
-104
-84
0
88
104
Distance in Arc Direction (Degrees)


Distribution Curve in Arc Direction
(Run No.4. Sample 7.)
Separation Rate (Impacts/Sensor.2s)
600
500
400
300
200
100
0
-100
First Row
Second Row
Third Row
-104
-84
0
88
104
Distance in Arc Direction (Degrees)

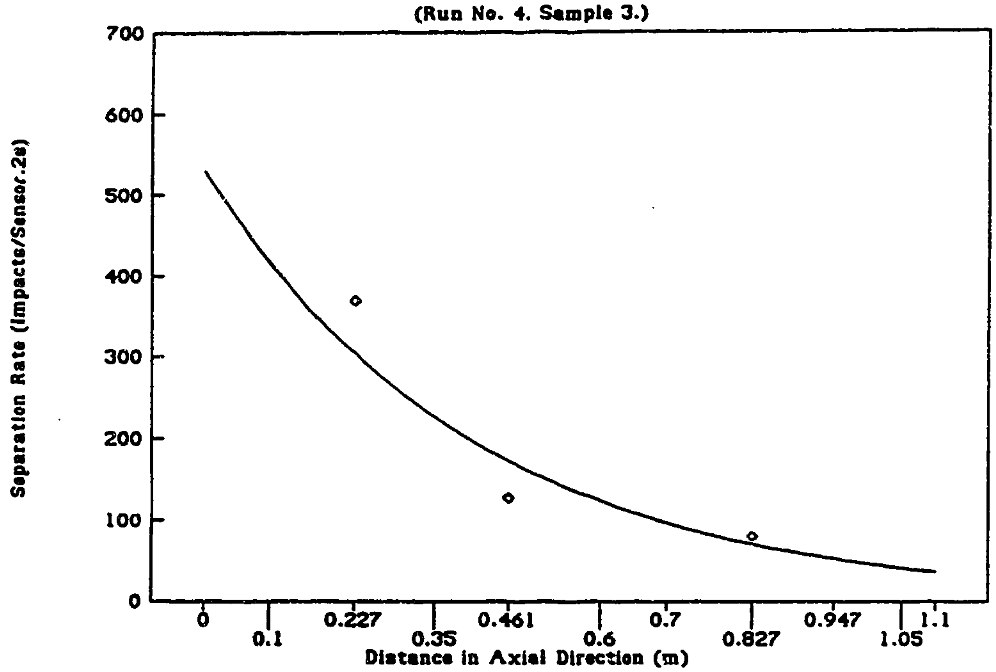
Appendix IV. 2. The Best Fit Curves in Axial Direction





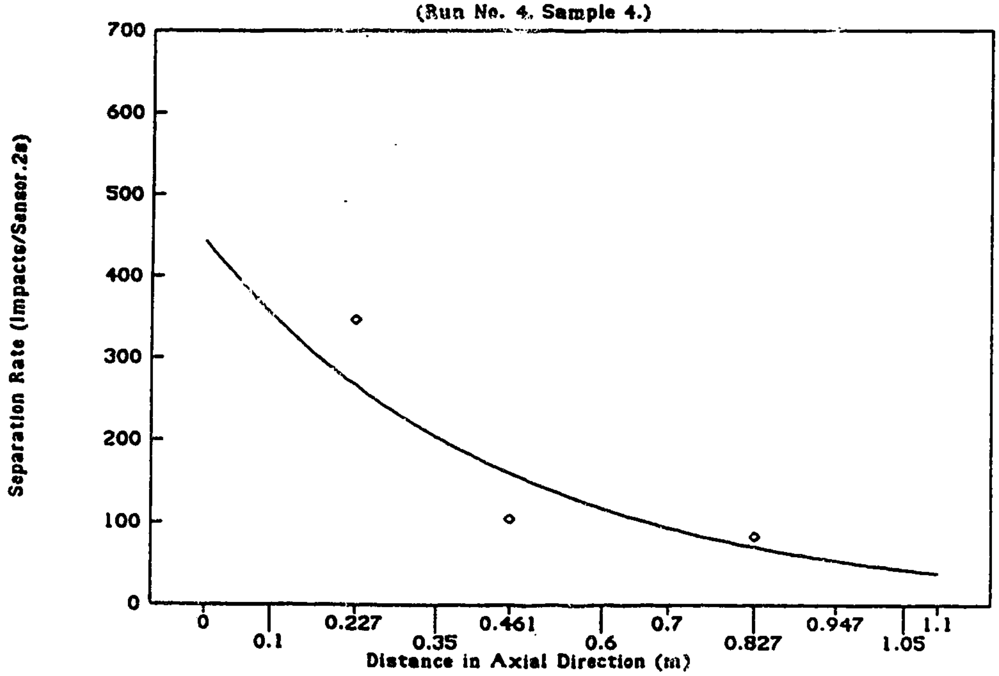
Regression Curve in Axial Direction

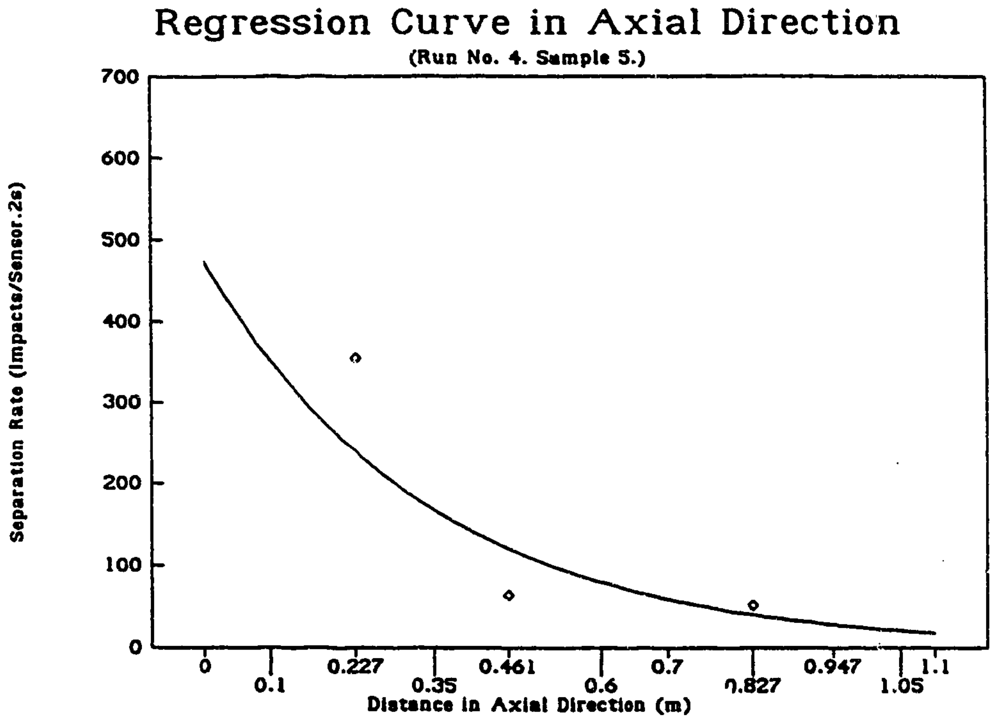
(Run No. 4. Sample 3.)

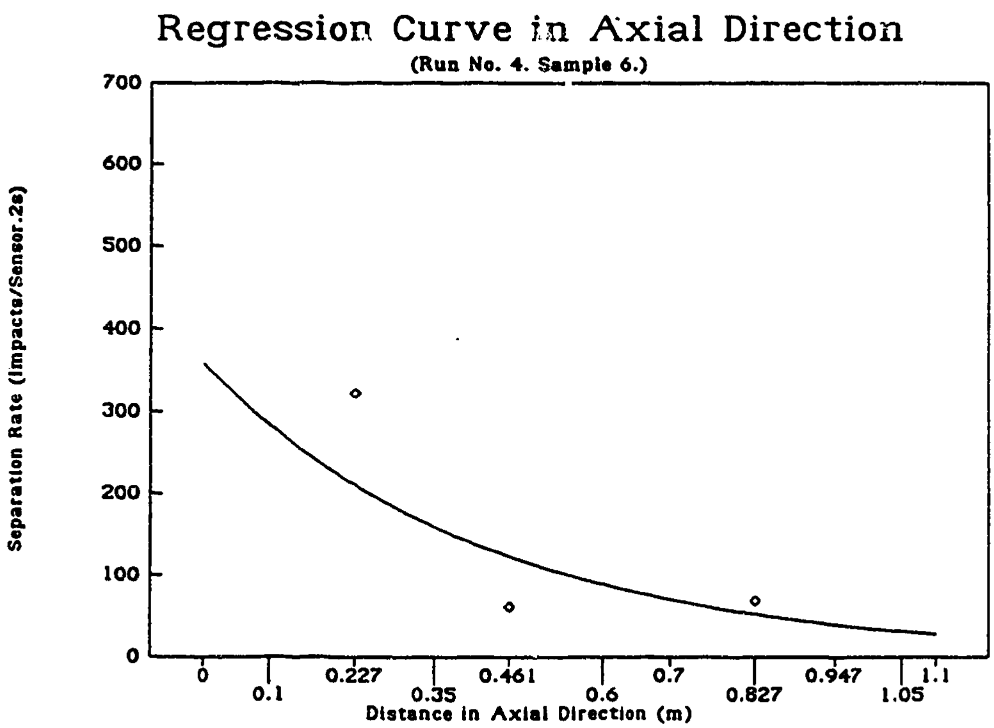


Regression Curve in Axial Direction

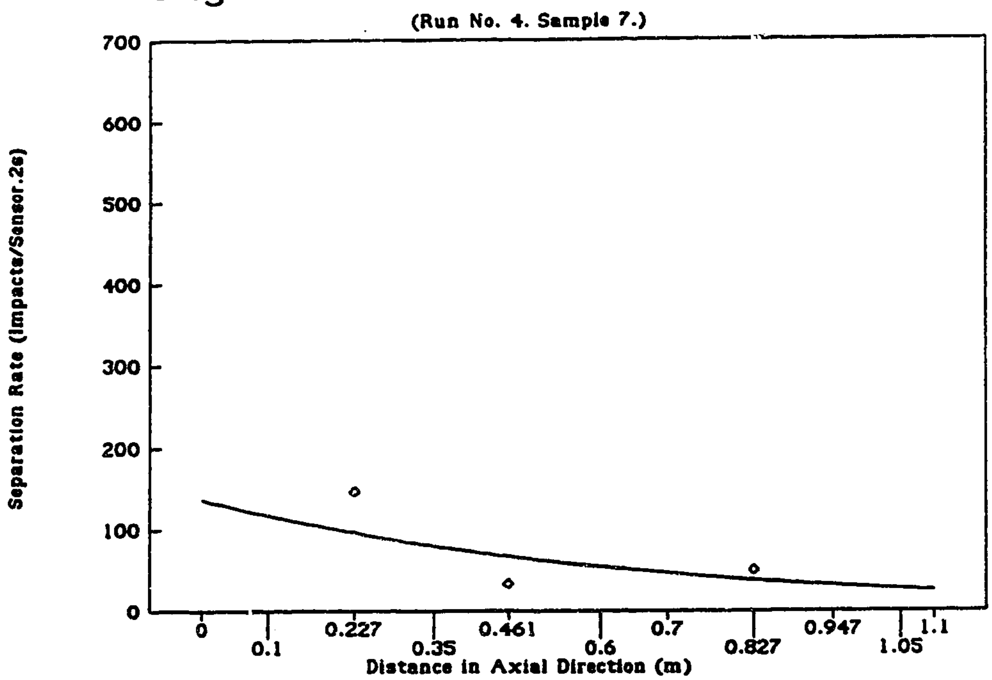
(Run No. 4. Sample 4.)




Regression Curve in Axial Direction
(Run No. 4. Sample 5.)
Separation Rate (Impacts/Sensor.2s)
700
600
500
400
300
200
100
0
0
0.1
0.227
0.35
0.461
0.6
0.7
0.827
0.947
1.05
1.1
Distance in Axial Direction (m)


Regression Curve in Axial Direction
(Run No. 4. Sample 6.)
Separation Rate (Impacts/Sensor.2s)
700
600
500
400
300
200
100
0
0
0.1
0.227
0.35
0.461
0.6
0.7
0.827
0.947
1.05
1.1
Distance in Axial Direction (m)

Regression Curve in Axial Direction



Appendix V.

Combine Grain Loss Monitoring Assembly Language Program List

```
                                   TITLE    COMBINE GRAIN LOSS MONITOR

                             .LIST
                                          PAGE      71,132
                             ;------------------------------------------------------------------;
                             ;     This progam was written by Liu, Chengqi, graduate student at  ;
                             ; the Department of Agricultural Engineering, University of Alberta. ;
                             ;     This project was to develope a data acquisition and real time ;
                             ; data processing system to monitor grain loss from an axial-flow   ;
                             ; combine.                                                          ;
                             ;     The supervisor for this project was Dr. Jeremy Leonard,       ;
                             ; professor at the Department of Agricultural Engineering, University ;
                             ; of Alberta.                                                       ;
                             ;     The following program is composed of three main parts:       ;
                             ;                                                                  ;
                             ;     1. Variable definition.                                      ;
                             ;     2. Real Time Data Processing Program.                        ;
                             ;     3. Mini-operation System.                                    ;
                             ;                                                                  ;
                             ;     The Sampling Duration Time was 2 Seconds.                    ;
                             ;                                                                  ;
                             ;     All right reserved.                                          ;
                             ;------------------------------------------------------------------;
0000                         GRAIN       SEGMENT   PARA PUBLIC 'CODE'
                                         ASSUME    CS:GRAIN, DS:GRAIN, SS:GRAIN, ES:GRAIN

                             ;           EXTRN     CLOCK_READ:FAR

                             ;------------------------------------------------------------------;
                             ;           PART 1: VARIABLE DEFINATIONS                           ;
                             ;------------------------------------------------------------------;
= 0001                                   CM_1      EQU      1         ;1!
= 0002                                   CM_2      EQU      2         ;2!
= 0006                                   CM_3      EQU      6         ;3!
= 0018                                   CM_4      EQU      24        ;4!
= 0078                                   CM_5      EQU      120       ;5!
= 02D0                                   CM_6      EQU      720       ;6!
= 13B0                                   CM_7      EQU      5040      ;7!
= 9D80                                   CM_8      EQU      40320     ;8!
= 0000                                   IOA       EQU    00H         ;I/O 8255
= 0001                                   IOB       EQU    01H
= 0002                                   IOC       EQU    02H
= 0003                                   IOCTR     EQU    03H
= 0010                                   UART_DATA EQU    10H         ;RS 232 8259
= 0011                                   UART_CTRL EQU    11H
= 0020                                   TIMER_0   EQU    20H         ;TIMER 8253 BAUD RATE GENERATER
= 0021                                   TIMER_1   EQU    21H         ;SQURE WAVE GENERATER
= 0022                                   TIMER_2   EQU    22H         ;SECON SIGNAL GENERATER
= 0023                                   TIMER_M   EQU    23H         ;MODE WORD REGISTER

= 0030                                   COM_WRD_1 EQU    30H         ;PROGRAMMABLE INTERRUPT
                                                                      ;CONTROLER 8259
= 0031                                   COM_WRD_234 EQU  31H         ;COMMAND WORD 1 & COMMAND
                                                                      ;WORDS 2, 3, 4
= 0040                                   COUNTER_1 EQU    40H         ;SENSOR 1
= 0041                                   COUNTER_2 EQU    41H         ;SENSOR 2
= 0042                                   COUNTER_3 EQU    42H         ;SENSOR 3
= 0043                                   MODE_1    EQU    43H
= 0050                                   IOA_2     EQU    50H         ;A/D, A/D CONTROL, DIP SWITCH
= 0051                                   IOB_2     EQU    51H         ;PB2 BEEP, PB3 S GATA, PB7
= 0052                                   IOC_2     EQU    52H         ;COUNTER GATEC
= 0053                                   IOCTR_2   EQU    53H
= 0060                                   COUNTER_4 EQU    60H         ;SENSOR 4
= 0061                                   COUNTER_5 EQU    61H         ;SENSOR 5
= 0062                                   COUNTER_6 EQU    62H         ;SENSOR 6
= 0063                                   MODE_2    EQU    63H
= 0070                                   COUNTER_7 EQU    70H         ;SENSOR 7
```

```
Microsoft (R) Macro Assembler  Version 4.00            9/6/90 21:17:24

COMBINE GRAIN LOSS MONITOR                              Page     1-2

 = 0071                            COUNTER_8 EQU    71H        ;SENSOR 8
 = 0072                            COUNTER_9 EQU    72H        ;SENSOR 9
 = 0073                            MODE_3    EQU    73H
 = 0080                            CWOOO     EQU    80H        ;SIX DIFFERENT COMBINATIONS
 = 0089                            CWOOI     EQU    89H        ;FOR I/O PORT 8255.
 = 0090                            CWIOO     EQU    90H        ;I - INPUT, O - OUTPUT.
 = 0099                            CWIOI     EQU    99H
 = 0092                            CWIIO     EQU    92H
 = 009B                            CWIII     EQU    9BH
 = 0000                            STACK_BS  EQU  0000H        ;STACK POINT
 = 0000                            POINT     EQU  0000H
 = 27FF                            YEAR      EQU  27FFH        ;THESE VARIABLES ARE THE
 = 27FE                            MONTH     EQU  27FEH        ;ADDRESSES OF MEMORY UNITS IN
 = 27FD                            DATE      EQU  27FDH        ;REAL TIME CLOCK.  THEY CAN
 = 27FC                            DAY       EQU  27FCH        ;BE ACCESSED AS REGULAR RAM.
 = 27FB                            HR        EQU  27FBH
 = 27FA                            MIN       EQU  27FAH
 = 27F9                            SEC       EQU  27F9H
 = 27F8                            RTCTRL    EQU  27F8H        ;REAL TIME CONTRL WORD ADDRESS.

 0400                              ORG       0400H
 0400  1000[           OUT_PUT     DB        1000H DUP(0)      ;UP LOAD DATA AREA IN RAM.
           00
                  ]


 2000                              ORG       2000H
 2000  0000            RUN_No      DW        0          ;THE NUMBER OF RUNS.
 2002  0000            SAMPLE_No   DW        0          ;THE NUMBER OF SAMPLES.
 2004  0000            SAMPLE_No_1 DW        0          ;THE NUMBER OF SAMPLES FOR PRINT SIZE
                                                        ;CALCULATE.
 2006  0000            OUT_MEM     DW        0          ;KEEP THE CURRENT ADDRESS OF OUT-FILE.
 2008  00              FLAG1       DB        0          ;DISPLAY LENGTH COUNTER LCD FULL IT=10H
 2009  00              FLAG2       DB        0          ;AFTER CALL WORKING SET TO 1.
 200A  00              FLAG3       DB        0          ;DISPLAY REAL TIME CLOCK.
 200B  00              FLAG5       DB        0          ;IF SET, INTERRUPT HAS BEEN SERVICED.
 200C  00              FLAG6       DB        0          ;SET YEAR OR HOUR.
 200D  00              FLAG7       DB        0          ;1D33H.
 200E  ????            POINT_S     DW        ?
 2010  ????            POINT_S1    DW        ?          ;PROTECT THE POSITION WHERE THE NEXT
                                                        ;LETTER DIS
 2012  ??              KEY_DIS     DB        ?          ;FOR PRESSED KEY DISPLAY
 2013  0014[           STRING      DB        20 DUP(?)  ;FOR COMMAND STORAGE
           ??
                  ]

 2027  ??              COUTER_A    DB        ?          ;THIS PART WAS THE VARIABLES FOR
 2028  ??              COUTER_B    DB        ?          ;KEYBOARD MONITOR PROGRAM.
 2029  ??              ALM         DB        ?
 202A  ??              BLM         DB        ?
 202B  ????            PRESS_T     DW        ?          ;TIMES THE KEYS WERE PRESSED
 202D  ????            KEY_NUMBER  DW        ?          ;WHICH KEY WAS PRESSED
 202F  0010[           YEAR_RAM    DB        16 DUP(?)    ;YEAR -- DAY  IN RAM WOULD BE IN LCD
           ??
                  ]

 203F  0010[           HOUR_RAM    DB        16 DUP(?)    ;HOUR -- SECOND IN RAM FOR LCD
           ??
                  ]

 204F  ??              I_T_V       DB        ?            ;INTERVAL TIME VARIABLE
 2050  ??              T_RAM1      DB        ?            ;
 2051  ??              T_RAM2      DB        ?            ;
 2052  0010[           P_DATA      DW 16 DUP(?)           ;1D34H--1D43H
           ????
                  ]
```

```
 2072  0010[                 DATA_DISPLAY DB 16 DUP(0)       ;DATA TO BE DISPLAYED STORAGE
               00
                        ]

 2082  00                    LEFT         DB      0          ;RESERVED FOR DISPLAY CHANNELS
                                                             ;NUMBER 1,3,5,7,9
 2083  00                    RIGHT        DB      0          ;RESERVED FOR DISPLAY CHANNELS
                                                             ;NUMBER 2,4,6,8,0
                             ;------------------------------------------------------------;
                             ;            NEXT PART OF MEMORY NEEDS TO BE CLEARED          ;
                             ;            AFTER EACH SAMPLE DATA PROCESSING.               ;
                             ;------------------------------------------------------------;
 2084  0002[                 RANGE        DW      2 DUP(0)   ;SENSOR RANGE DETERMINATION WORD
               0000
                        ]

 2088  0000                  RANGE_1      DW      0          ;AFTER ENLARGEMENT DATA STORE HERE
 208A  0000                  SEN_RM       DW      0          ;SENSORS ADDRESS MEMORY UNIT
 208C  0000                  SEN_L_RM     DW      0          ;SENSORS LOGRITHM ADDRESS MEMORY UNIT
 208E  0000                  N            DW      0          ;N IN EQU. Ln10^N
 2090  0000                  N_DIVIDE     DW      0
 2092  0000                  Xi_M         DW      0          ;Xi ADDRESS MEMORY UNIT
 2094  0000                  LnYi_M       DW      0          ;LnYi ADDRESS MEMORY UNIT
 2096  00                    SIGN_X       DB      0          ;THE SIGN OF SIG(Xi-Xmean)
 2097  00                    SIGN_Y       DB      0          ;THE SIGN OF SIG(Yi-Ymean)
 2098  00                    SIGN_SIGMA   DB      0          ;THE SIGN OF D
 2099  0000                  Xi_Xm_M      DW      0          ;Xi-Xm ADDR. FOR CALCULATE (Xi-Xm)^2
 209B  00                    ABOVE_16     DB      0          ;FLAG = 1 WHEN DIVISOR IS >=65536
 209C  0004[                 EXP_N        DW      4 DUP(0)   ;THE RESULT OF EXP N BIN
               0000
                        ]

 20A4  0000                  Z_MEAN_RM    DW      0          ;STORE Z_MEAN_1 ADDRESS
 20A6  0000                  OUT_PUT_RM   DW      0          ;STORE OUT_PUT ADDRESS
 20A8  0000                  DI_MEM       DW      0          ;DISTINATION ADDRESS MEMORY
 20AA  0004[                 S1_TEMP      DW      4 DUP(0)   ;THE FINAL RESULT TEMPERARY STORSGE
               0000
                        ]

 20B2  0008[                 TABLE_M      DW      8 DUP(0)   ;TEMPERORY LOOK_UP TABLE STORAGE UNIT
               0000
                        ]

 20C2  0008[                 PAR_PRODUCT  DW      8 DUP(0)   ;PARTIAL PRODUCT OF 8 DIGIT MULTIPLY
               0000
                        ]

 20D2  0008[                 INTERMIDIATE DW      8 DUP(0)   ;2 BYTES FOR ADDING PARTIAL PRODUCT
               0000
                        ]

 20E2  0008[                 PRODUCT      DW      8 DUP(0)   ;PRODUCT OF EIGHT DIGIT MULTIPLY
               0000
                        ]

 20F2  0008[                 FIVE_PLUS    DW      8 DUP(0)   ;4 OMMIT 5 PLUS FOR MULTIPLICATION
               0000
                        ]

 2102  0008[                 SIGMA_LnYi_M DW      8 DUP(0)   ;16 BITS FOR SIGMA Yi TEMPERARELY
               0000
                        ]

 2112  0008[                 MEAN_LnYi    DW      8 DUP(0)   ;16 BITS FOR MEAN VALUE OF LnYi
               0000
                        ]
```

```
2122  0008[                     X1           DW        8 DUP(0)     ;16 BITS FOR X1
           0000
                      ]

2132  0008[                     X2           DW        8 DUP(0)     ;16 BITS FOR X2
           0000
                      ]

2142  0008[                     X3           DW        8 DUP(0)     ;16 BITS FOR X3
           0000
                      ]

2152  0008[                     X0           DW        8 DUP(0)     ;16 BITS FOR X0
           0000
                      ]

2162  0008[                     SIGMA_Xi_M   DW        8 DUP(0)     ;16 BITS FOR SIGMA Xi TEMPERARELY
           0000
                      ]

2172  0008[                     MEAN_Xi      DW        8 DUP(0)     ;MEAN VALUE OF Xi
           0000
                      ]

2182  0008[                     COMPLETE     DW        8 DUP(0)     ;0000000010000000 ARE STORED
           0000
                      ]

2192  0008[                     INTERMIDIATE_SUB DW    8 DUP(0)     ;16 BITS FOR THE RESULT OF SUB Xi-Xm
           0000
                      ]

                                                                    ;OR Yi-Ym
21A2  0008[                     SIG_Xi_Xm_Yi_Ym  DW    8 DUP(0)     ;RESULT OF D's NOMINATOR
           0000
                      ]

21B2  0008[                     Xi_Xm        DW        8 DUP(0)     ;Xi - Xm
           0000
                      ]

21C2  0008[                     Yi_Ym        DW        8 DUP(0)     ;Yi - Ym
           0000
                      ]

21D2  0008[                     X1_Xm        DW        8 DUP(0)
           0000
                      ]

21E2  0008[                     X2_Xm        DW        8 DUP(0)
           0000
                      ]

21F2  0008[                     X3_Xm        DW        8 DUP(0)
           0000
                      ]

2202  0008[                     X4_Xm        DW        8 DUP(0)
           0000
                      ]

2212  0008[                     SIG_Xi_Xm_SQ_2         DW    8 DUP(0) ;RASULT OF D'S DENOMIINATION.
           0000
                      ]

2222  0008[                     SIG_Xi_Xm_Yi_Ym_BIN    DW    8 DUP(0) ;BIN OF D'S NOMINATION
           0000
```

```
                        ]

 2232  0008[                  SIG_Xi_Xm_BIN          DW    8 DUP(0) ;BIN OF D'S DENOMINATION
            0000
                        ]

 2242  0008[                  D_BCD        DW        8 DUP(0)       ;BCD OF D
            0000
                        ]

 2252  0008[                  D_BCD_FRACTION   DW    8 DUP(0)       ;BCD FRACTION OF D
            0000
                        ]

 2262  0008[                  D_15258      DW        8 DUP(0)       ;SET IN BIN2BCD FOR FRAGMENT PART
            0000
                        ]

 2272  0008[                  EXP_b        DW        8 DUP(0)       ;FOR EXP b BCD
            0000
                        ]

 2282  0008[                  EXPa_MUL_EXPb_INTEGER  DW    8 DUP(0) ;
            0000
                        ]

 2292  0008[                  EXPa_MUL_EXPb_FRAGMENT DW    8 DUP(0) ;
            0000
                        ]

 22A2  0008[                  C_BIN        DW        8 DUP(0)       ; C = Ymean - D * Xmean
            0000
                        ]

 22B2  0008[                  C_BCD        DW        8 DUP(0)       ;
            0000
                        ]

 22C2  0008[                  MEAN_LnYi_FOR_C  DW    8 DUP(0)       ;STORE C_BCD FOR C_BIN CONVERT
            0000
                        ]

 22D2  0004[                  TIME_START   DW        4 DUP(0)       ;TO TEST THE TIME THE CALCULATE
            0000
                        ]

 22DA  0004[                  TIME_FINISH  DW        4 DUP(0)       ;EXCUTING NEEDED.
            0000
                        ]

 22E2  0000                   Z1             DW      0
 22E4  0000                   Z2             DW      0
 22E6  0000                   Z3             DW      0
 22E8  0004[                  SIG_Zi         DW      4 DUP(0)
            0000
                        ]

 22F0  0003[                  SIG_XiZi       DW      3 DUP(0)
            0000
                        ]

 22F6  0000                   SYMBLE_SIGXiZi DW      0           ;SYMBLE 0 =+1 =- FOR SIGMA XiZi
 22F8  0004[                  SIG_Xi_SQ2_Zi  DW      4 DUP(0)
            0000
                        ]

 2300  0002[                  A_DIV_B        DW      2 DUP(0)    ;STORE A/B
```

```
          0000
                     ]

 2304  0002[                 NEGA_B_X0      DW      2 DUP(0)     ;STORE -B*X0
          0000
                     ]

 2308  0002[                 EXP_N_B_X0     DW      2 DUP(0)     ;STORE EXP (-B*X0)
          0000
                     ]

 230C  0004[                 Z_MEAN         DW      4 DUP(0)
          0000
                     ]

 2314  0003[                 A_PARA         DW      3 DUP(0)
          0000
                     ]

 231A  0000                  SYMBLE_A_PARA  DW      0            ;SYMBLE 0 =+1=-FOR A PAOAMETER
 231C  0003[                 B_PARA         DW      3 DUP(0)
          0000
                     ]

 2322  0000                  SYMBLE_B_PARA  DW      0            ;SYMBLE 0 =+1=-FOR B PARAMETER
 2324  0003[                 C_PARA         DW      3 DUP(0)
          0000
                     ]

 232A  0000                  SYMBLE_C_PARA  DW      0            ;SYMBLE 0 =+1=-FOR C PARAMETER

 2330                                       ORG      2330H
 2330  0000                  SENSOR_1        DW      0           ;RAM FOR SENSORS ORIGINAL DATA
 2332  0000                  SENSOR_2        DW      0
 2334  0000                  SENSOR_3        DW      0
 2336  0000                  SENSOR_4        DW      0
 2338  0000                  SENSOR_5        DW      0
 233A  0000                  SENSOR_6        DW      0
 233C  0000                  SENSOR_7        DW      0
 233E  0000                  SENSOR_8        DW      0
 2340  0000                  SENSOR_9        DW      0
 2342  0000                  SENSOR_0        DW      0

 2344  0003[                 A_PARA_1        DW       3 DUP(0)
          0000
                     ]

 234A  0000                  SYMBLE_A_PARA_1 DW       0          ;SYMBLE 0 =+1=- FOR A PAOAMETER
 234C  0003[                 B_PARA_1        DW       3 DUP(0)
          0000
                     ]

 2352  0000                  SYMBLE_B_PARA_1 DW       0          ;SYMBLE 0 =+1=- FOR B PARAMETER
 2354  0003[                 C_PARA_1        DW       3 DUP(0)
          0000
                     ]

 235A  0000                  SYMBLE_C_PARA_1 DW       0          ;SYMBLE 0 =+1=- FOR C PARAMETER
 235C  0003[                 A_PARA_2        DW       3 DUP(0)
          0000
                     ]

 2362  0000                  SYMBLE_A_PARA_2 DW       0          ;SYMBLE 0 =+1=- FOR A PAOAMETER
 2364  0003[                 B_PARA_2        DW       3 DUP(0)
          0000
                     ]
```

```
 236A  0000                     SYMBLE_B_PARA_2 DW        0         ;SYMBLE 0 =+1=- FOR B PARAMETER
 236C  0003[                    C_PARA_2        DW        3 DUP(0)
            0000
                    ]

 2372  0000                     SYMBLE_C_PARA_2 DW        0         ;SYMBLE 0 =+1=- FOR C PARAMETER
 2374  0003[                    A_PARA_3        DW        3 DUP(0)
            0000
                    ]

 237A  0000                     SYMBLE_A_PARA_3 DW        0         ;SYMBLE 0 =+1=- FOR A PAOAMETER
 237C  0003[                    B_PARA_3        DW        3 DUP(0)
            0000
                    ]

 2382  0000                     SYMBLE_B_PARA_3 DW        0         ;SYMBLE 0 =+1=- FOR B PARAMETER
 2384  0003[                    C_PARA_3        DW        3 DUP(0)
            0000
                    ]

 238A  0000                     SYMBLE_C_PARA_3 DW        0         ;SYMBLE 0 =+1=- FOR C PARAMETER
 238C  0004[                    Z_MEAN_1        DW        4 DUP(0)  ;MEAN VALUE OF SENSORS IN ARC
            0000
                    ]

                                                                    ;DIRECTION
 2394  0004[                    Z_MEAN_2        DW        4 DUP(0)
            0000
                    ]

 239C  0004[                    Z_MEAN_3        DW        4 DUP(0)
            0000
                    ]

 23A4  0004[                    Z_MEAN_1_L      DW         4 DUP(0) ;RAM RESERVED FOR LOGRITHM
            0000
                    ]

 23AC  0004[                    Z_MEAN_2_L      DW         4 DUP(0)
            0000
                    ]

 23B4  0004[                    Z_MEAN_3_L      DW         4 DUP(0)
            0000
                    ]

 23BC  0004[                    A_COEF          DW         4 DUP(0) ;STORE A COEF. OF A/B*EXP(-B*X0)
            0000
                    ]

 23C4  0003[                    D_BIN           DW         3 DUP(0) ;BIN OF D = B
            0000
                    ]

 23CA  0000                     SYMBLE_B        DW         0        ;SYMBLE 0 = +1 =-FOR C PARAMETER
 23CC  0004[                    LOSS_DATA_1     DW         4 DUP(0) ;STORE A/B*EXP(-B*X0)
            0000
                    ]

 23D4  0004[                    LOSS_DATA       DW         4 DUP(0) ;STORE K*A/B*EXP(-B*X0)
            0000
                    ]

                                ;-----------------------------------------------------------------------;
                                ;              THE FOLLOWING MEMORY WAS TO STORE THE FORMAT OF           ;
                                ;         THE PRINTED FILE.  THE FORMAT WOULD BE SENT TO PRINTER         ;
```

```
                                   ;        (RADIO SHAK TP-10) WITH THE DATA FILLED IN THE BLANK        ;
                                   ;        SPACES IN THE FORMAT FILE.                                  ;
                                   ;----------------------------------------------------------------------;
 23E0                                          ORG       23E0H      ;FORMAT OF PRINTED FILE
 23E0  53 41 4D 50 4C 45 20        OUT_PUT_1   DB        'SAMPLE  -------------  '
       20 2D 2D 2D 2D 2D 2D
       2D 2D 2D 2D 2D 2D 2D
       20 20
 23F7  20 20 20 20 20 20 20        SAMPLE      DB        '         '
       20 20
 2400  20 20 20 20 20 20 20                    DB        '                                '
       20 20 20 20 20 20 20
       20 20 20 20 20 20 20
       20 20 20 20 20 20 20
       20 20 20 20
 2420  53 31 20 3D 20                          DB        'S1 = '
 2425  30 30 30 30 20 20 20        S1          DB        '0000        S2 = '
       20 20 20 20 53 32 20
       3D 20
 2435  30 30 30 30 20 20 20        S2          DB        '0000       '
       20 20 20 20
 2440  53 33 20 3D 20                          DB        'S3 = '
 2445  30 30 30 30 20 20 20        S3          DB        '0000        S4 = '
       20 20 20 20 53 34 20
       3D 20
 2455  30 30 30 30 20 20 20        S4          DB        '0000       '
       20 20 20 20
 2460  53 35 20 3D 20                          DB        'S5 = '
 2465  30 30 30 30 20 20 20        S5          DB        '0000        S6 = '
       20 20 20 20 53 36 20
       3D 20
 2475  30 30 30 30 20 20 20        S6          DB        '0000       '
       20 20 20 20
 2480  53 37 20 3D 20                          DB        'S7 = '
 2485  30 30 30 30 20 20 20        S7          DB        '0000        S8 = '
       20 20 20 20 53 38 20
       3D 20
 2495  30 30 30 30 20 20 20        S8          DB        '0000       '
       20 20 20 20
 24A0  53 39 20 3D 20                          DB        'S9 = '
 24A5  30 30 30 30 20 20 20        S9          DB        '0000                          '
       20 20 20 20 20 20 20
       20 20 20 20 20 20 20
       20 20 20 20 20 20
 24C0  20 20 20 20 20 20 20                    DB        '                                '
       20 20 20 20 20 20 20
       20 20 20 20 20 20 20
       20 20 20 20 20 20 20
       20 20 20 20
 24E0  41 31 20 3D 20                          DB        'A1 = '
 24E5  2B 31 32 33 34 2E 35        A1          DB        '+1234.5678  B1 = '
       36 37 38 20 20 42 31
       20 3D 20
 24F6  2B 31 32 33 34 2E 35        B11         DB        '+1234.5678'
       36 37 38
 2500  43 31 20 3D 20                          DB        'C1 = '
 2505  2B 31 32 33 34 2E 35        C11         DB        '+1234.5678  Z1 = '
       36 37 38 20 20 5A 31
       20 3D 20
 2516  20 31 32 33 34 2E 30        ZZ1         DB        ' 1234.0000'
       30 30 30
 2520  20 20 20 20 20 20 20                    DB        '                                '
       20 20 20 20 20 20 20
       20 20 20 20 20 20 20
       20 20 20 20 20 20 20
       20 20 20 20
 2540  41 32 20 3D 20                          DB        'A2 = '
```

```
2545  2B 31 32 33 34 2E 35      A2          DB        '+1234.5678  B2 = '
      36 37 38 20 20 42 32
      20 3D 20
2556  2B 31 32 33 34 2E 35      B2          DB        '+1234.5678'
      36 37 38
2560  43 32 20 3D 20                        DB        'C2 = '
2565  2B 31 32 33 34 2E 35      C22         DB        '+1234.5678  Z2 = '
      36 37 38 20 20 5A 32
      20 3D 20
2576  20 31 32 33 34 2E 30      ZZ2         DB        ' 1234.0000'
      30 30 30
2580  20 20 20 20 20 20 20                  DB        '                              '
      20 20 20 20 20 20 20
      20 20 20 20 20 20 20
      20 20 20 20 20 20 20
      20 20 20 20
25A0  41 33 20 3D 20                        DB        'A3 = '
25A5  2B 31 32 33 34 2E 35      A3          DB        '+1234.5678  B3 = '
      36 37 38 20 20 42 33
      20 3D 20
25B6  2B 31 32 33 34 2E 35      B3          DB        '+1234.5678'
      36 37 38
25C0  43 33 20 3D 20                        DB        'C3 = '
25C5  2B 31 32 33 34 2E 35      C3          DB        '+1234.5678  Z3 = '
      36 37 38 20 20 5A 33
      20 3D 20
25D6  20 31 32 33 34 2E 30      ZZ3         DB        ' 1234.0000'
      30 30 30
25E0  20 20 20 20 20 20 20                  DB        '                              '
      20 20 20 20 20 20 20
      20 20 20 20 20 20 20
      20 20 20 20 20 20 20
      20 20 20 20
2600  4C 6E 5A 31 3D 20                     DB        'LnZ1= '
2606  30 31 32 33 2E 34 35      LnZ1        DB        '0123.4567  LnZ2= '
      36 37 20 20 4C 6E 5A
      32 3D 20
2617  30 31 32 33 2E 34 35      LnZ2        DB        '0123.4567'
      36 37
2620  4C 6E 5A 33 3D 20                     DB        'LnZ3= '
2626  30 31 32 33 2E 34 35      LnZ3        DB        '0123.4567                 '
      36 37 20 20 20 20 20
      20 20 20 20 20 20 20
      20 20 20 20 20
2640  20 20 41 3D 20                        DB        '  A= '
2645  2B 30 30 30 30 2E 30      AA          DB        '+0000.0000     B= '
      30 30 30 20 20 20 20
      42 3D 20
2656  2B 30 30 30 30 2E 30      BB          DB        '+0000.0000'
      30 30 30
2660  20 47 4C 3D 20                        DB        ' GL= '
2665  2B 30 30 30 30 2E 30      GGLL        DB        '+0000.0000                '
      30 30 30 20 20 20 20
      20 20 20 20 20 20 20
      20 20 20 20 20 20
2680  20 20 20 20 20 20 20                  DB        '                              '
      20 20 20 20 20 20 20
      20 20 20 20 20 20 20
      20 20 20 20 20 20 20
      20 20 20 20

                                ;--------------------------------------------------------------;
                                ;            THE MAIN PROGRAM STARTS HERE                       ;
                                ;--------------------------------------------------------------;
0000                                        ORG       0000H
0000                            OUR_PROG    PROC      NEAR
```

```
 0000  B8 0000            START:      MOV     AX,STACK_BS ;SET STACK SEGMENT.
 0003  8E D0                          MOV     SS,AX
 0005  BC 27F0                        MOV     SP,27F0H     ;SET STACK POINT
 0008  B0 90                          MOV     AL,CWIOO     ;INNITIALIZATION PIO.
 000A  E6 03                          OUT     IOCTR,AL
 000C  B9 0014                        MOV     CX,20        ;DELAY 20mS.
 000F  E8 1EE8 R                      CALL    DN1MS
 0012  E8 1B33 R                      CALL    LCD_INI      ;INICTIALIZATION OF LCD
 0015  E8 1AEF R                      CALL    DIS_TITLE    ;DISPLAY TITLE (THIS SUBROUTINE
                                                           ;WAS USED FOR THE SYSTEM BASIC
                                                           ;FUNCTION TEST.).
 0018  FC                             CLD
 0019  BE CEC6                        MOV     SI,0CEC6H    ;POINT TO PRINT FILE ADDRESS.
 001C  8D 3E 23E0 R                   LEA     DI,OUT_PUT_1 ;THE ADDRESS OF TABLE TO BE FILLED
 0020  B9 0160                        MOV     CX,22*16     ;22 RAWS EACH RAW 16 WORDS
 0023  F3/ A5             REP         MOVSW                ;THE SOURCE PRINT FILE WAS AT
 0025                     RUN_I:                           ;EPROM
 0025  B0 99                          MOV     AL,CWIOI     ;COMMAND WORDS FOR 8255-2
 0027  E6 53                          OUT     IOCTR_2,AL   ;I/O A IN, I/O B OUT, I/O C IN
 0029  B0 04                          MOV     AL,00000100B ;8255-2 I/O B BIT 2 OUTPUT HIGH
 002B  E6 51                          OUT     IOB_2,AL
 002D  B0 70                          MOV     AL,01110000B ;SET OUT(INTERRUPT REQUEST) ZERO
 002F  E6 23                          OUT     TIMER_M,AL   ;MODE 0 INTERRUPT ON COUNT
                                                           ;TERMINATION
 0031  B0 00                          MOV     AL,00H
 0033  A2 2008 R                      MOV     [FLAG1],AL   ;COUNTER OF THE LCD POSITION
 0036  A2 2009 R                      MOV     [FLAG2],AL   ;FLAG2 1 FOR TEST BEGIN
 0039  A2 200A R                      MOV     [FLAG3],AL   ;REAL TIME CLOCK DIS
 003C  A2 200C R                      MOV     [FLAG6],AL   ;YEAR OR HOUR SET
 003F  A2 200D R                      MOV     [FLAG7],AL   ;INTERVAL HAS BEEN SET (FFH)
 0042  A2 200B R                      MOV     [FLAG5],AL   ;GATE TIME = 10 S.
 0045  8D 3E 2002 R                   LEA     DI,SAMPLE_No ;SAMPLES NUMBER

 0049  B8 0000                        MOV     AX,00H
 004C  89 05                          MOV     [DI],AX      ;CLEAR
 004E  8D 3E 2004 R                   LEA     DI,SAMPLE_No_1
 0052  89 05                          MOV     [DI],AX

 0054  8D 3E 2330 R                   LEA     DI,SENSOR_1  ;CLEAR THE SENSORS AREA.
 0058  B9 000A                        MOV     CX,10
 005B  B8 0000                        MOV     AX,0
 005E  89 05              GPPE:       MOV     [DI],AX
 0060  47                             INC     DI
 0061  47                             INC     DI
 0062  E2 FA                          LOOP    GPPE

 0064  E8 03A7 R                      CALL    CLEAR_DATA_AREA ;CLEAR DATA AREA BEFOR SAMPLLING

 0067  B8 0000                        MOV     AX,0000H
 006A  A3 202B R                      MOV     [PRESS_T],AX ;CLEAR KEY PRESS TIMES

 006D  BE D24E                        MOV     SI,0D24EH    ;SEND TIME IN EPROM TO RAM
 0070  8D 3E 202F R                   LEA     DI,YEAR_RAM
 0074  B9 0020                        MOV     CX,32
 0077  A4                 SYS:        MOVS    BYTE PTR[DI],BYTE PTR[SI]
 0078  E2 FD                          LOOP    SYS

 007A  8D 3E 0400 R                   LEA     DI,OUT_PUT   ;STORE THE FIRST ADDRESS OF
 007E  89 3E 2006 R                   MOV     [OUT_MEM],DI ;SAMPLE & RESULT OUTPUT FILE.

 0082  8D 1E 2072 R                   LEA     BX,DATA_DISPLAY ;POINT TO DATA DISPLAY UNIT
 0086  B8 5250                        MOV     AX,'RP'      ;DISPLAY 'PRESS F2 TO TEST'
 0089  89 07                          MOV     [BX],AX
 008B  B8 5345                        MOV     AX,'SE'
 008E  89 47 02                       MOV     [BX+2],AX
 0091  B8 2053                        MOV     AX,' S'
 0094  89 47 04                       MOV     [BX+4],AX
```

```
0097  B8 3246                      MOV     AX, '2F'
009A  89 47 06                     MOV     [BX+6],AX
009D  B8 5420                      MOV     AX,'T '
00A0  89 47 08                     MOV     [BX+8],AX
00A3  B8 204F                      MOV     AX,' O'
00A6  89 47 0A                     MOV     [BX+10],AX
00A9  B8 4554                      MOV     AX,'ET'
00AC  89 47 0C                     MOV     [BX+12],AX
00AF  B8 5453                      MOV     AX,'TS'
00B2  89 47 0E                     MOV     [BX+14],AX
00B5  E8 1EBC R                    CALL    DISPLAY_1     ;DIS 'PRESS F2 TO TEST' WITHOUT
                                                         ;1s TIME DELAY.
00B8  E4 50           K_00:        IN      AL,IOA_2      ;CHECK THE DIP SWITCH.
00BA  B2 00                        MOV     DL,00H
00BC  3A C2                        CMP     AL,DL         ;IF = 00H THEN CLEAR DATA AREA.
00BE  74 F8                        JZ      K_00

00C0  E4 50           K_80:        IN      AL,IOA_2      ;AT THE END OF MAIN PROGRAM
00C2  B2 80                        MOV     DL,80H        ;JMP TO HERE.
00C4  3A C2                        CMP     AL,DL
00C6  75 F0                        JNZ     K_00

00C8  8D 1E 2009 R                 LEA     BX,FLAG2      ;CHECK FLAG2 IF 1 TEST IS BEGIN.
00CC  B0 01                        MOV     AL,01H        ;FLAG2 WAS SET IN WORKING.
00CE  3A 07                        CMP     AL,[BX]
00D0  74 04                        JZ      III           ;JUMP TO INTERRUPT WAITING LOOP
00D2  FA                           CLI                   ;IF FLAG2 NO EQUAL TO 1,
                                                         ;CLEAR INTERRUPT.
00D3  E9 15BD R                    JMP     KEY_SCANNING ;=80H TO KEY SCANNING

00D6  B0 FE           III:         MOV     AL,0FEH
00D8  E6 31                        OUT     31H,AL        ;OCW1

00DA  B0 60                        MOV     AL,60H
00DC  E6 30                        OUT     30H,AL        ;OCW2

00DE  B0 08                        MOV     AL,08H
00E0  E6 30                        OUT     30H,AL        ;OCW3

00E2  B0 00                        MOV     AL,0          ;IF FLAG2 = 0
00E4  3A 06 2009 R                 CMP     AL,[FLAG2]
00E8  74 03                        JZ      INT_OFF       ;DISABLE INTERRUPT
00EA  EB 05 90                     JMP     INT_ON
00ED  FA              INT_OFF:     CLI
00EE  E9 0025 R                    JMP     RUN_I         ;GO TO THE BEGINNING FOR NEXT RUN
00F1  FB              INT_ON:      STI
00F2  EB E2                        JMP     III

                      OUR_PROG     ENDP
                      ;
                      ;
00F4                  WORKING      PROC    NEAR
                                   ;--------------------------------------------;
                                   ; 1. CALL INTERRUPT_INI.  2. CALL COUNTER_INI ;
                                   ; 3. SET THE FLAG2 IN TO 1.                   ;
                                   ; 4. DISPLAY TEST ON LCD.                     ;
                                   ;--------------------------------------------;
00F4  E8 0138 R                    CALL    INTERRUPT_INI
00F7  E8 0182 R                    CALL    COUNTER_INI
00FA  E8 01DD R                    CALL    PULS_GENERATE ;GENERATE A PULS TO TRIGGE COUNTER
                                                         ;AFTER CALL THE S & COUN GATE ARE
                                                         ;OPEND.
00FD  8D 3E 2009 R                 LEA     DI,FLAG2
0101  C6 05 01                     MOV     BYTE PTR[DI],01H ;IF THIS FLAG KEEP 1, THE
                                                            ;INTERRUPT WILL BE ENABLE
0104  8D 1E 2072 R                 LEA     BX,DATA_DISPLAY  ;POINT TO DATA DISPLAY UNIT
0108  B8 4554                      MOV     AX,'ET'
```

```
 010B  89 07                         MOV      [BX],AX
 010D  B8 5453                       MOV      AX,'TS'
 0110  89 47 02                      MOV      [BX+2],AX
 0113  B8 2020                       MOV      AX,'  '
 0116  89 47 04                      MOV      [BX+4],AX
 0119  B8 4542                       MOV      AX, 'EB'
 011C  89 47 06                      MOV      [BX+6],AX
 011F  B8 4947                       MOV      AX,'IG'
 0122  89 47 08                      MOV      [BX+8],AX
 0125  B8 204E                       MOV      AX,' N'
 0128  89 47 0A                      MOV      [BX+10],AX
 012B  B8 2020                       MOV      AX,'  '
 012E  89 47 0C                      MOV      [BX+12],AX
 0131  89 47 0E                      MOV      [BX+14],AX
 0134  E8 1EBC R                     CALL     DISPLAY_1      ;DIS 'TEST  BEGING'
                                                             ;WITHOUT 1s DELAY.
 0137  C3                            RET

                         WORKING     ENDP
                         ;
                         ;
 0138                    INTERRUPT_INI         PROC        NEAR
                               ;-----------------------------------------------------------;
                               ; THIS SUB[illegible]ITIALISE                                 ;
                               ; 1. PROGRAMMA[illegible] CONTROLLER 8259A                    ;
                               ; 2. TIMER AND COU[illegible]                                 ;
                               ;    a) BAUD RA[illegible] b) SECOND SQUARE WAVE GENERATER.;
                               ;-----------------------------------------------------------;
                                     ;PART 1[illegible] entry address.
 0138  BB 0060                       MOV      [illegible]
 013B  BE FC1D                       MOV      [illegible],0FC1DH      ;ENTRY ADDRESS
 013E  89 37                         MOV      [BX],SI        ;VECTOR IS 18H 18H*4=60H

 0140  BB 0062                       MOV      BX,0062H       ;SET CS VALUE FOR INTERRUPT
 0143  B8 0000                       MOV      AX,0000H
 0146  89 07                         MOV      [BX],AX

                                     ;PART 2: INICIALIZE FOR 8259 INTERRRUPT FUNCTION.
 0148  B0 13                         MOV      AL,13H
 014A  E6 30                         OUT      COM_WRD_1,AL   ;ICW1   COMMAVD WORD 1

 014C  B0 18                         MOV      AL,18H
 014E  E6 31                         OUT      COM_WRD_234,AL ;ICW2

 0150  B0 0D                         MOV      AL,0DH
 0152  E6 31                         OUT      COM_WRD_234,AL ;ICW4

                                     ;PART 3: INICIALIZE 8253
 0154  B0 36                         MOV      AL,00110110B   ;INICIALIZE TIMER CHENNEL 0
 0156  E6 23                         OUT      TIMER_M,AL     ;AS BAUD RATE GENERATER
 0158  B0 04                         MOV      AL,04H         ;LOAD LSB    04H
 015A  E6 20                         OUT      TIMER_0,AL
 015C  B0 01                         MOV      AL,01H         ;LOAD MSB    01H
 015E  E6 20                         OUT      TIMER_0,AL

 0160  B0 70                         MOV      AL,01110000B   ;CHENNEL 1
 0162  E6 23                         OUT      TIMER_M,AL     ;MODE 0 INTERRUPT ON TERMINAL COUNT
 0164  B0 1F                         MOV      AL,1FH         ;IT IS USED TO BE 1FH FOR IS.
 0166  E6 21                         OUT      TIMER_1,AL     ;LOAD LSB
 0168  B0 C3                         MOV      AL,0C3H        ;IT USED TO BE 0C3H FOR 1S.
 016A  E6 21                         OUT      TIMER_1,AL     ;LOAD MSB

 016C  B0 B6                         MOV      AL,10110110B   ;CHENNEL 2
 016E  E6 23                         OUT      TIMER_M,AL     ;MODE 3 AS A SQUARE WAVE GENERATOTR
 0170  B0 64                         MOV      AL,64H         ;IT USED TO BE 32 FOR 1S.
 0172  E6 22                         OUT      TIMER_2,AL     ;LOAD LSB
 0174  B0 00                         MOV      AL,00H
```

```
 0176  E6 22                         OUT     TIMER_2,AL      ;LOAD MSB

 0178  B0 99                         MOV     AL,CWIOI
 017A  E6 53                         OUT     IOCTR_2,AL
 017C  B0 8C                         MOV     AL,10001100B    ;AFTER INICIALISATION OPEN
 017E  E6 51                         OUT     IOB_2,AL        ;THE SECOND AND SENSORS GATA.

 0180  FA                            CLI                     ;CLEAR INTERRUPT FLAG

 0181  C3                            RET

                     INTERRUPT_INI           ENDP
                     ;
                     ;
 0182                COUNTER_INI     PROC    NEAR
                                     ;-------------------------------;
                                     ;PROGRAMMABLE COUNTERS 1 -- 9   ;
                                     ;INICIALIZATION                 ;
                                     ;-------------------------------;
 0182  B0 30                         MOV     AL,00110000B
 0184  E6 43                         OUT     MODE_1,AL       ;CHANNEL 1 MODE
 0186  B0 00                         MOV     AL,00H
 0188  E6 40                         OUT     COUNTER_1,AL    ;LSB LOAD
 018A  E6 40                         OUT     COUNTER_1,AL    ;MSB LOAD

 018C  B0 70                         MOV     AL,01110000B
 018E  E6 43                         OUT     MODE_1,AL       ;CHANNEL 2 MODE
 0190  B0 00                         MOV     AL,00H
 0192  E6 41                         OUT     COUNTER_2,AL    ;LSB LOAD
 0194  E6 41                         OUT     COUNTER_2,AL    ;MSB LOAD

 0196  B0 B0                         MOV     AL,10110000B
 0198  E6 43                         OUT     MODE_1,AL       ;CHANNEL 3 MODE
 019A  B0 00                         MOV     AL,00H
 019C  E6 42                         OUT     COUNTER_3,AL    ;LSB LOAD
 019E  E6 42                         OUT     COUNTER_3,AL    ;MSB LOAD

 01A0  B0 30                         MOV     AL,00110000B
 01A2  E6 63                         OUT     MODE_2,AL       ;CHANNEL 4 MODE
 01A4  B0 00                         MOV     AL,00H
 01A6  E6 60                         OUT     COUNTER_4,AL    ;LSB LOAD
 01A8  E6 60                         OUT     COUNTER_4,AL    ;MSB LOAD

 01AA  B0 70                         MOV     AL,01110000B
 01AC  E6 63                         OUT     MODE_2,AL       ;CHANNEL 5 MODE
 01AE  B0 00                         MOV     AL,00H
 01B0  E6 61                         OUT     COUNTER_5,AL    ;LSB LOAD
 01B2  E6 61                         OUT     COUNTER_5,AL    ;MSB LOAD

 01B4  B0 30                         MOV     AL,10110000B
 01B6  E6 63                         OUT     MODE_2,AL       ;CHANNEL 6 MODE
 01B8  B0 00                         MOV     AL,00H
 01BA  E6 62                         OUT     COUNTER_6,AL    ;LSB LOAD
 01BC  E6 62                         OUT     COUNTER_6,AL    ;MSB LOAD

 01BE  B0 30                         MOV     AL,00110000B
 01C0  E6 73                         OUT     MODE_3,AL       ;CHANNEL 7 MODE
 01C2  B0 00                         MOV     AL,00H
 01C4  E6 70                         OUT     COUNTER_7,AL    ;LSB LOAD
 01C6  E6 70                         OUT     COUNTER_7,AL    ;MSB LOAD

 01C8  B0 70                         MOV     AL,01110000B
 01CA  E6 73                         OUT     MODE_3,AL       ;CHANNEL 8 MODE
 01CC  B0 00                         MOV     AL,00H
 01CE  E6 71                         OUT     COUNTER_8,AL    ;LSB LOAD
 01D0  E6 71                         OUT     COUNTER_8,AL    ;MSB LOAD
```

```
01D2  B0 B0                             MOV     AL,10110000B
01D4  E6 73                             OUT     MODE_3,AL      ;CHANNEL 9 MODE
01D6  B0 00                             MOV     AL,00H
01D8  EC 72                             OUT     COUNTER_9,AL   ;LSB LOAD
01DA  E6 72                             OUT     COUNTER_9,AL   ;MSB LOAD

01DC  C3                                RET

                        COUNTER_INI     ENDP
                        ;
                        ;
01DD                    PULS_GENERATE  PROC    NEAR
                                        ;-------------------------------------;
                                        ;THIS SUBROUTINE IS TO GENERATE A PULS ;
                                        ;TRIGGE THE COUNTERS THE FUNCTION IS   ;
                                        ;EQUAL TO CLEAR COUNTERS.              ;
                                        ;-------------------------------------;
01DD  B0 99                             MOV     AL,CWIOI
01DF  E6 53                             OUT     IOCTR_2,AL
01E1  B0 8C                             MOV     AL,10001100B
01E3  E6 51                             OUT     IOB_2,AL
01E5  90                                NOP                    ;TIME DELAY
01E6  90                                NOP
01E7  90                                NOP
01E8  90                                NOP
01E9  B0 88                             MOV     AL,10001000B   ;SEND A NEGATIVE PULS
01EB  E6 51                             OUT     IOB_2,AL       ;TO CLEAR COUNTER.
01ED  90                                NOP
01EE  90                                NOP                    ;TIME DELAY.
01EF  90                                NOP
01F0  90                                NOP
01F1  B0 8C                             MOV     AL,10001100B   ;NOW COUNTERS HAS BEEN CLEARSD
01F3  E6 51                             OUT     IOB_2,AL

01F5  C3                                RET

                        PULS_GENERATE           ENDP

                        ;-------------------------------------------------------------------;
                        ;               PART 2. REAL TIME DATA ACQUISATION AND PROCESSING.   ;
                        ;-------------------------------------------------------------------;

01F6                    REAL_TIME_DATA_PROCESSING   PROC   NEAR
                                ;---------------------------------------------------------;
                                ;    This program was to conduct the best fit and regression ;
                                ;calculation.                                              ;
                                ; 1. The best fit in arc direction was conducted at  three ;
                                ;    different cross section positions.  The results were  ;
                                ;    parameters A B & C for the model Z = A + BX + CX^2.   ;
                                ;    The meanvalues Z_MEAN_1, Z_MEAN_2 & Z_MEAN_3 would be ;
                                ;    obtained.                                             ;
                                ; 2. The axial diraction regression was conducted.  The    ;
                                ;    results were A & B for the  model    Z = A EXP (-BY)   ;
                                ; 3. The actual grain loss then would be predicted.        ;
                                ;    the formula for this calculation is:                  ;
                                ;              GL = K * A/B EXP (-BY0)                     ;
                                ;---------------------------------------------------------;
                        ;       LEA     BX,TIME_START  ;THIS 6 INSTRUCTIONS WERE FOR
                        ;       PUSH    BX             ;TSTING THE RTDP EXECUTION TIME
                        ;       PUSH    CX
                        ;       CALL    CLOCK_READ
                        ;       POP     CX
                        ;       POP     BX

01F6  8D 36 2330 R              LEA     SI,SENSOR_1    ;IF SENSOR_1 -- 9 = 0
01FA  B8 0000                   MOV     AX,0           ;DO NOT CALCULATE.
01FD  39 04                     CMP     [SI],AX
```

```
 01FF  75 2B                            JNZ     TO_CALCU

 0201  39 44 02                         CMP     [SI+2],AX
 0204  75 26                            JNZ     TO_CALCU

 0206  39 44 04                         CMP     [SI+4],AX
 0209  75 21                            JNZ     TO_CALCU

 020B  39 44 06                         CMP     [SI+6],AX
 020E  75 1C                            JNZ     TO_CALCU

 0210  39 44 08                         CMP     [SI+8],AX
 0213  75 17                            JNZ     TO_CALCU

 0215  39 44 0A                         CMP     [SI+10],AX
 0218  75 12                            JNZ     TO_CALCU

 021A  39 44 0C                         CMP     [SI+12],AX
 021D  75 0D                            JNZ     TO_CALCU

 021F  39 44 0E                         CMP     [SI+14],AX
 0222  75 08                            JNZ     TO_CALCU

 0224  39 44 10                         CMP     [SI+16],AX
 0227  75 03                            JNZ     TO_CALCU
 0229  E9 0317 R                        JMP     RESULT_MOVING

 022C  FC               TO_CALCU:       CLD
 022D  BE CEC6                          MOV     SI,0CEC6H     ;POINT OUT THE REAL ADDRESS OF TABLE
 0230  8D 3E 23E0 R                     LEA     DI,OUT_PUT_1  ;THE ADDRESS OF TABLE TO BE FILLED.
 0234  B9 0160                          MOV     CX,22*16      ;TABLE HAS 22 RAWS EACH RAW 16 WORDS
 0237  F3/ A5           REP             MOVSW                 ;THE SOURCE TABLE IS AT EPROM
                                        ;-------------------------------------------------------;
                                        ;    NEXT PART IS BEST FIT IN ARC DIRECTION.            ;
                                        ; 1. THE MODEL IS Z = A + B*X + C*X^2                   ;
                                        ; 2. THE PARAMETERS TO BE CALCULATED ARE A, B & C.      ;
                                        ; 3. Z_MEAN 1 THROUGH 3 ARE STORED IN Z_MEAN_1--Z_MEAN_3 ;
                                        ;    FOR LOGARISM CALCULATION.                          ;
                                        ; 4. ALL THE PARAMETERS & Z_MEANS ARE SENT TO OUT_PUT_1 ;
                                        ;    FOR UPLOAD OR PRINT, 200H STORAGE SPACE WERE RESERVED ;
                                        ;-------------------------------------------------------;
 0239  FC                               CLD
 023A  8D 36 2330 R                     LEA     SI,SENSOR_1
 023E  8D 3E 22E2 R                     LEA     DI,Z1
 0242  B9 0003                          MOV     CX,03
 0245  F3/ A5           REP             MOVSW                   ;BEFOR CALL DATA IN SENSORS,
 0247  8D 3E 238C R                     LEA     DI,Z_MEAN_1     ;SENT DATA TO Z1 Z2 Z3.
 024B  89 3E 20A4 R                     MOV     [Z_MEAN_RM],DI
 024F  E8 03DE R                        CALL    ARC_REG

 0252  FC                               CLD
 0253  8D 36 2314 R                     LEA     SI,A_PARA
 0257  8D 3E 2344 R                     LEA     DI,A_PARA_1
 025B  B9 000C                          MOV     CX,12                   ;AFTER ARC DIRECTION REGRESSION
 025E  F3/ A5           REP             MOVSW                           ;A B C ARE SENT TO OUT PUT AREA
 0260  E8 040A R                        CALL    CLEAR_A_B_C

 0263  FC                               CLD
 0264  8D 36 2336 R                     LEA     SI,SENSOR_4
 0268  8D 3E 22E2 R                     LEA     DI,Z1
 026C  B9 0003                          MOV     CX,03
 026F  F3/ A5           REP             MOVSW                   ;BEFOR CALL DATA IN SENSORS
 0271  8D 3E 2394 R                     LEA     DI,Z_MEAN_2     ;SENT DATA TO Z1 Z2 Z3 .
 0275  89 3E 20A4 R                     MOV     [Z_MEAN_RM],DI
 0279  E8 03DE R                        CALL    ARC_REG

 027C  FC                               CLD
```

```
027D  8D 36 2314 R              LEA     SI,A_PARA
0281  8D 3E 235C R              LEA     DI,A_PARA_2
0285  B9 000C                   MOV     CX,12          ;AFTER ARC DIRECTION REGRESSION
0288  F3/ A5            REP     MOVSW                  ;A B C ARE SENT TO OUT PUT AREA
028A  E8 040A R                 CALL    CLEAR_A_B_C

028D  FC                        CLD
028E  8D 36 233C R              LEA     SI,SENSOR_7
0292  8D 3E 22E2 R              LEA     DI,Z1
0296  B9 0003                   MOV     CX,03
0299  F3/ A5            REP     MOVSW                  ;BEFOR CALL DATA IN SENSORS
029B  8D 3E 239C R              LEA     DI,Z_MEAN_3    ; SENT TO Z1 Z2 Z3.
029F  89 3E 20A4 R              MOV     [Z_MEAN_RM],DI
02A3  E8 03DE R                 CALL    ARC_REG

02A6  FC                        CLD
02A7  8D 36 2314 R              LEA     SI,A_PARA
02AB  8D 3E 2374 R              LEA     DI,A_PARA_3
02AF  B9 000C                   MOV     CX,12          ;AFTER ARC DIRECTION REGRESSION
02B2  F3/ A5            REP     MOVSW                  ;A B C ARE SENT TO OUT PUT AREA
02B4  E8 040A R                 CALL    CLEAR_A_B_C

                        ;----------------------------------------------------------;
                        ; NEXT PART IS REGRESSION IN AXIAL DIRECTION FOR PARAMETERS ;
                        ; A & B.                                                    ;
                        ;----------------------------------------------------------;
02B7  E8 041B R                 CALL    SENSOR_LOOK    ;LnYi LOOK_UP SUBROUTINE
02BA  E8 06EC R                 CALL    SIGMA_LnYi     ;SIGMA LnYi
02BD  E8 0704 R                 CALL    MEAN_VALUE_Yi  ;MEAN VALUE LnYi
02C0  E8 0728 R                 CALL    MEAN_VALUE_Xi  ;MEAN VALUE Xi
02C3  E8 0789 R                 CALL    D_NOMINATION   ;CALCULATE THE NOMIINATION OF
                                                       ;PARAMETER D.
02C6  E8 0944 R                 CALL    D_DENOMINATION ;CALCULATE THE DENOMIINATION OF
                                                       ;PARAMETER D.
02C9  8D 36 21A2 R              LEA     SI,SIG_Xi_Xm_Yi_Ym ;DATA NEED TO BE CONVERTED
02CD  8D 1E 2222 R              LEA     BX,SIG_Xi_Xm_Yi_Ym_BIN ;BINERY RESULT STORAGE
02D1  E8 0AC2 R                 CALL    BCD_BIN                ;CONVERT D'S NOMINATION
                                                               ;FROM BCD TO BINERY
02D4  8D 36 2212 R              LEA     SI,SIG_Xi_Xm_SQ_2      ;DATA NEED TO BE CONVERTED
02D8  8D 1E 2232 R              LEA     BX,SIG_Xi_Xm_BIN       ;BINERY RESULT STORAGE
02DC  E8 0AC2 R                 CALL    BCD_BIN                ;CONVERT D'S DENOMINATION
                                                               ;FROM BCD TO BINERY
02DF  8D 36 2222 R              LEA     SI,SIG_Xi_Xm_Yi_Ym_BIN ;THE ADDRESS OF NOMINATION
02E3  8D 3E 2232 R              LEA     DI,SIG_Xi_Xm_BIN       ;THE ADDRESS OF THE
                                                               ; DENOMINATION
02E7  8D 1E 23C4 R              LEA     BX,D_BIN               ;THE ADDRESS OF D
                                                               ;PARAMETER RESULT
02EB  E8 0979 R                 CALL    D_RESULT               ;CALCULATE THE PARAMETER
                                                               ;D (BIN)
02EE  8D 36 23C4 R              LEA     SI,D_BIN       ;THE DATA TO BE CONVERTED
02F2  8D 1E 2242 R              LEA     BX,D_BCD       ;THE FINAL RESULT ADDRESS
02F6  8D 3E 2252 R              LEA     DI,D_BCD_FRACTION ;DATA IN [D_BCD_FRACTION]
02FA  E8 0A1A R                 CALL    BIN_BCD        ;NEED TO * 0.000015258

02FD  E8 09EF R                 CALL    C_RESULT       ;CALCULATE THE PARAMETER C (BCD)
                                                       ;THE RESULT WILL PUT IN
                                                       ;[MEAN_Yi_FOR_C]*
0300  8D 36 22C2 R              LEA     SI,MEAN_LnYi_FOR_C ;DATA NEED TO BE CONVERTED
0304  8D 1E 22A2 R              LEA     BX,C_BIN           ;BINERY RESULT STORAGE
0308  E8 0B7A R                 CALL    BCD_BIN_C          ;CONVERT C BCD TO BIN

030B  E8 0BC8 R                 CALL    A_COEF_A_D_B   ;CALCULATE A COEF & A/B
030E  E8 0C1F R                 CALL    NEGA_B_MUL_X0  ;CALCULATE -B*X0
0311  E8 [illegible] R          CALL    EXP_NEGA_B_X0  ;CALCULATE EXP(-B*X0)
0314  E8 [illegible]E R         CALL    LOSS           ;CALCULATE A/B*EXP(-B*X0)
                        ;       CALL    GRAIN_LOSS     ;CALCULATE K*A/B*EXP(-B*X0)
```

```
0317                       RESULT_MOVING:
                                 ;----------NEXT PART IS RESULT MOVING TO UP LOAD AREA------;
0317  8D 3E 2002 R               LEA    DI,SAMPLE_No  ;POINT TO SAMPLES NUMBER
031B  8B 05                      MOV    AX,[DI]
031D  40                         INC    AX            ;NEXT SAMPLE
031E  37                         AAA
031F  89 05                      MOV    [DI],AX
0321  0D 3030                    OR     AX,3030H
0324  8D 36 23F9 R               LEA    SI,SAMPLE+2
0328  8B D0                      MOV    DX,AX
032A  8A C6                      MOV    AL,DH
032C  8A E2                      MOV    AH,DL
032E  89 04                      MOV    [SI],AX
0330  BA 3131                    MOV    DX,3131H
0333  3B C2                      CMP    AX,DX         ;SAMPLES NUMBER = 10 ?
0335  75 03                      JNZ    KFT           ;NO, CONTINOUE.

0337  EB 58 90                   JMP    KFT_1         ;AFTER TEST AND PRINT OUT GO TO
                                                      ;CLEARE DATA AREA.
033A  E8 1442 R            KFT:  CALL   SENSOR_MOVING ;MOVE SENSOR DATA TO UP LODING AREA
033D  E8 14B6 R                  CALL   ABC_MOVING    ;MOVE A B &C PARAMETERS TO UP LOAD
0340  E8 152F R                  CALL   Z_MEAN_Ln_MOV ;MOVE Z_MEAN Ln VALUE TO UP LOAD

0343  8D 36 23BC R               LEA    SI,A_COEF     ;THE DATA TO BE CONVERTED
0347  8D 3E 2645 R               LEA    DI,AA
034B  89 3E 20A8 R               MOV    [DI_MEM],DI
034F  8D 1E 20AA R               LEA    BX,S1_TEMP    ;THE FINAL RESULT TEMPERARY STORSGE
0353  8D 3E 2252 R               LEA    DI,D_BCD_FRACTION ;DATA IN [D_BCD_FRACTION]
                                                      ; NEED TO * 0.000015258
0357  E8 0A1A R                  CALL   BIN_BCD
035A  E8 157A R                  CALL   M_F

035D  8D 36 23C4 R               LEA    SI,D_BIN
0361  8D 3E 2656 R               LEA    DI,BB
0365  89 3E 20A8 R               MOV    [DI_MEM],DI
0369  8D 1E 20AA R               LEA    BX,S1_TEMP
036D  8D 3E 2252 R               LEA    DI,D_BCD_FRACTION
0371  E8 0A1A R                  CALL   BIN_BCD
0374  E8 157A R                  CALL   M_F

0377  8D 36 23CC R               LEA    SI,LOSS_DATA_1
037B  8D 3E 2665 R               LEA    DI,GGLL
037F  89 3E 20A8 R               MOV    [DI_MEM],DI
0383  8D 1E 20AA R               LEA    BX,S1_TEMP
0387  8D 3E 2252 R               LEA    DI,D_BCD_FRACTION
038B  E8 0A1A R                  CALL   BIN_BCD
038E  E8 157A R                  CALL   M_F

0391  E8 03A7 R            KFT_1: CALL  CLEAR_DATA_AREA ;CLEAR DATA AREA
                                                        ;AFTER EVERY SAMPLES.

                                 ;----------NEXT PART IS TO MOVE BLOCK TO OUT_PUT-------;
0394  FC                         CLD
0395  8D 36 23E0 R               LEA    SI,OUT_PUT_1
0399  8B 3E 2006 R               MOV    DI,[OUT_MEM]
039D  B9 0160                    MOV    CX,16*22
03A0  F3/ A5               REP   MOVSW
03A2  89 3E 2006 R               MOV    [OUT_MEM],DI

                           ;     LEA    BX,TIME_FINISH ;THIS PART IS TO CHECK THE SYSTEM
                           ;     PUSH   BX             ;CLOCK AFTER THE RTDP WAS END.
                           ;     PUSH   CX
                           ;     CALL   CLOCK_READ
                           ;     POP    CX
                           ;     POP    BX

03A6  C3                         RET
```

```
                                        ;-------------------------------------;
                                        ;     THE RTDP PROGRAM END HERE.      ;
                                        ;-------------------------------------;
                         REAL_TIME_DATA_PROCESSING     ENDP
                          ;----------------------------------------------------------------;
                          ;  THE NEXT 46 SUBROUTINES WERE CALLED BY REAL_TIME_DATA_PROCESSING. ;
                          ;  THE STRUCTURE WERE HIERARCHICAL AND NESTED MIXTURE.              ;
                          ;----------------------------------------------------------------;
03A7                     CLEAR_DATA_AREA PROC    NEAR          ;SUBROUTINE No.1 FOR THE RTDP
                                        ;-------------------------------------;
                                        ; CLEAR DATA AREA AFTER EVERY SAMPLES ;
                                        ;-------------------------------------;
03A7  8D 3E 2C8? R                      LEA     DI,RANGE
03AB  B9 0153                           MOV     CX,339
03AE  B8 0000            ALL:           MOV     AX,0
03B1  89 05                             MOV     [DI],AX
03B3  47                                INC     DI
03B4  47                                INC     DI
03B5  E2 F7                             LOOP    ALL

03B7  8D 3E 2330 R                      LEA     DI,SENSOR_1
03BB  B9 0053                           MOV     CX,53H
03BE  B8 0000            ALL_1:         MOV     AX,0
03C1  89 05                             MOV     [DI],AX
03C3  47                                INC     DI
03C4  47                                INC     DI
03C5  E2 F7                             LOOP    ALL_1

03C7  C3                                RET

                         CLEAR_DATA_AREA        ENDP
                         ;
                         ;
03C8                     ARC_REGRESSION  PROC   NEAR          ;SUBROUTINE No.2 FOR THE RTDP.

03C8  E8 0FA0 R                         CALL    SIGMA_Zi       ;CALCULATE SIGMA Zi(i=1--3)
                                                               ;SENSOR'S DATA
03CB  E8 0FB3 R                         CALL    SIGMA_XiZi     ;Xi(i=1--3) ARE THE LOCATIONS OF
                                                               ;THE SENEORS
03CE  E8 1010 R                         CALL    SIGMA_Xi_SQ_2_Zi
03D1  E8 1040 R                         CALL    A_PARAMETER    ;CALCULATE PARAMETER A
03D4  E8 1149 R                         CALL    B_PARAMETER    ;CALCULATE PARAMETER B
03D7  E8 1230 R                         CALL    C_PARAMETER    ;CALCULATE PARAMETER C
03DA  E8 1338 R                         CALL    Z_MEAN_VALUE   ;CALCULATE MEAN VALUE OF Zi

03DD  C3                                RET

                         ARC_REGRESSION  ENDP
                         ;
                         ;
03DE                     ARC_REG        PROC    NEAR          ;SUBROUTINE No.3 FOR RTDP
                              ;------------------------------------------------------------;
                              ;THIS SUBROUTINE IS TO CONDUCT ARC REGRESSION                 ;
                              ; 1. BEFOR CALLING, 3 SENSORS DATA ARE STORED IN Z1, Z2, Z3.  ;
                              ; 2. Z_MEAN_1 ADDRESS IS STORED IN [Z_MEAN_RM]                ;
                              ; 3. OUT_PUT_1 ADDRESS FOR PRINT IS ALSO STORED IN [OUT_PUT_RM];
                              ;------------------------------------------------------------;
03DE  E8 03C8 R                         CALL    ARC_REGRESSION
03E1  8D 36 2?0C R                      LEA     SI,Z_MEAN
03E5  8B 04                             MOV     AX,[SI]
03E7  BA 8000                           MOV     DX,8000H
03EA  3B C2                             CMP     AX,DX          ;IF AX EQUAL TO OR ABOUVE .8000H(0.5)

03EC  73 03                             JAE     FUL
03EE  EB 10 90                          JMP     FUM
03F1  FF 44 02           FUL:           INC     WORD PTR[SI+2] ;Z_MEAN PLUS 1
03F4  8B 44 02                          MOV     AX,[SI+2]      ;Z_MEAN IN AX
03F7  8B 3E 20A4 R                      MOV     DI,[Z_MEAN_RM]
```

```
 03FB  89 05                            MOV    [DI],AX        ;Z_MEAN NOW IN Z_MEAN_1, _2,_3
 03FD  EB 0A 90                         JMP    DOA

 0400  8B 44 02           FUM:          MOV    AX,[SI+2]      ;Z_MEAN IN AX
 0403  8B 3E 20A4 R                     MOV    DI,[Z_MEAN_RM]
 0407  89 05                            MOV    [DI],AX        ;Z_MEAN NOW IN Z_MEAN_1, _2,_3

 0409  C3                 DOA:          RET

                          ARC_REG       ENDP
                          ;
                          ;
 040A                     CLEAR_A_B_C   PROC   NEAR           ;SUBROUTINE No.4 FOR THE RTDP.

 040A  B9 000C                          MOV    CX,12
 040D  B8 0000                          MOV    AX,0
 0410  8D 1E 2314 R                     LEA    BX,A_PARA
 0414  89 07              KLD:          MOV    [BX],AX        ;CLEAR DATA AREA FOR NEXT TIME USE
 0416  43                               INC    BX
 0417  43                               INC    BX
 0418  E2 FA                            LOOP   KLD

 041A  C3                               RET

                          CLEAR_A_B_C   ENDP
                          ;
                          ;
 041B                     SENSOR_LOOK   PROC   NEAR           ;SUBROUTINE No.5 FOR THE RTDP.

 041B  B9 0003                          MOV    CX,3           ;THE NUMBER OF Z_MEAN
 041E  8D 1E 238C R                     LEA    BX,Z_MEAN_1    ;POINT THE FIRST SENSOR
 0422  89 1E 208A R                     MOV    [SEN_RM],BX    ;SENSOR ADDRESS IN RAM

 0426  8D 1E 23A4 R                     LEA    BX,Z_MEAN_1_L  ;POINT TO FIRST Ln(Z_MEAN_1)
 042A  89 1E 208C R                     MOV    [SEN_L_RM],BX  ;ADDRESS OF Lnx IN RAM

 042E  8B 1E 208A R       SEN_1:        MOV    BX,[SEN_RM]    ;POINT TO SENSOR
 0432  8B 07                            MOV    AX,[BX]
 0434  A3 2084 R                        MOV    [RANGE],AX     ;ORIGANAL DATA IN [RANGE]

 0437  E8 049C R                        CALL   DATA_ENLARGE   ;DATA*100 (0 - 9); DATA*10 (10 -99)

 043A  51                               PUSH   CX             ;DATA_RANGE WILL USE CX
 043B  E8 04FF R                        CALL   DATA_RANGE     ;+0 OR +10000 OR +20000
 043E  59                               POP    CX

 043F  8B 1E 208A R                     MOV    BX,[SEN_RM]    ;GET Z_MEAN_1 -- 3 ADDRESS
 0443  83 C3 08                         ADD    BX,8           ;POINT TO NEXT ADDRESS
 0446  89 1E 208A R                     MOV    [SEN_RM],BX    ;KEEP NEXT ADDRESS IN RAM

 044A  8D 1E 20B2 R                     LEA    BX,TABLE_M     ;POINT TO TEMPERORY LOOK_UP
                                                              ;TABLE STORAGE UNIT
 044E  8B 07                            MOV    AX,[BX]        ;FIRST TABLE WORD
 0450  8B 1E 208C R                     MOV    BX,[SEN_L_RM]  ;IN RAM
 0454  89 07                            MOV    [BX],AX
 0456  43                               INC    BX
 0457  43                               INC    BX
 0458  53                               PUSH   BX
 0459  8D 1E 20B4 R                     LEA    BX,TABLE_M+2   ; POINT NEXT WORD
 045D  8B 07                            MOV    AX,[BX]
 045F  5B                               POP    BX
 0460  89 07                            MOV    [BX],AX
 0462  43                               INC    BX
 0463  43                               INC    BX
 0464  53                               PUSH   BX
 0465  8D 1E 20B6 R                     LEA    BX,TABLE_M+4   ;POINT TO INTEGER (*.)
 0469  8B 07                            MOV    AX,[BX]
```

```
 046B  5B                                      POP     BX
 046C  89 07                                   MOV     [BX],AX
 046E  43                                      INC     BX
 046F  43                                      INC     BX
 0470  53                                      PUSH    BX
 0471  8D 1E 20B8 R                            LEA     BX,TABLE_M+6  ;POINT TO INTEGER (*.)
 0475  8B 07                                   MOV     AX,[BX]
 0477  5B                                      POP     BX
 0478  89 07                                   MOV     [BX],AX

 047A  B8 3030                                 MOV     AX,3030H      ;CLEAR INTEGER MEMORY UNIT FOR
 047D  A3 20B6 R                               MOV     [TABLE_M+4],AX ;NEXT TIME USE.

 0480  83 C3 02                                ADD     BX,2          ; POINT TO NEXT Lnx ADDRESS
 0483  89 1E 208C R                            MOV     [SEN_L_RM],BX

 0487  E2 A5                                   LOOP    SEN_1

                                               ;---------------------------------------------;
                                               ; THIS PART IS FOR CHANGE ASCII CODE INTO BCD ;
                                               ; FOR CNOVINEINCE OF FOURTHER CALCULATION.    ;
                                               ;---------------------------------------------;
 0489  B9 0018                                 MOV     CX,24         ;CHANGE 3 X 8 BYTES ASCII IN TO BCD
 048C  8D 1E 23A4 R                            LEA     BX,Z_MEAN_1_L
 0490  B0 0F                                   MOV     AL,0FH
 0492  8A 17                   KS:             MOV     DL,[BX]
 0494  22 D0                                   AND     DL,AL
 0496  88 17                                   MOV     [BX],DL
 0498  43                                      INC     BX
 0499  E2 F7                                   LOOP    KS

 049B  C3                                      RET

                               SENSOR_LOOK     ENDP
                               ;
                               ;
 049C                          DATA_ENLARGE PROC       NEAR          ;SUBROUTINE No.6 FOR THE RTDP.

 049C  8D 1E 2084 R                            LEA     BX,RANGE
 04A0  8B 07                                   MOV     AX,[BX]
 04A2  3D 000A                                 CMP     AX,10
 04A5  72 12                                   JB      DATA_1        ; DATA BELOW 10 THEN *100
 04A7  3D 0064                                 CMP     AX,100
 04AA  72 1E                                   JB      DATA_10       ; DATA BELOW 100 THEN *10
 04AC  3D 03E8                                 CMP     AX,1000
 04AF  72 2A                                   JB      DATA_100      ; DATA BELOW 1000 THEN *1
 04B1  3D 2710                                 CMP     AX,10000
 04B4  72 37                                   JB      DATA_DIV_10   ; DATA BELOW 10000  THEN *0.1

 04B6  EB 46 90                                JMP     DATA_END      ; ABOUVE 10000 THEN RETURN

 04B9  BB 0064                 DATA_1:         MOV     BX,100
 04BC  F7 E3                                   MUL     BX
 04BE  A3 2088 R                               MOV     [RANGE_1],AX
 04C1  B8 0000                                 MOV     AX,00H
 04C4  A3 208E R                               MOV     [N],AX
 04C7  EB 35 90                                JMP     DATA_END

 04CA  BB 000A                 DATA_10:        MOV     BX,10
 04CD  F7 E3                                   MUL     BX
 04CF  A3 2088 R                               MOV     [RANGE_1],AX
 04D2  B8 0001                                 MOV     AX,01
 04D5  A3 208E R                               MOV     [N],AX
 04D8  EB 24 90                                JMP     DATA_END

 04DB  B8 0002                 DATA_100:       MOV     AX,02H
 04DE  A3 208E R                               MOV     [N],AX
```

```
 04E1  8D 1E 2088 R                      LEA   BX,RANGE_1
 04E5  A1 2084 R                         MOV   AX,[RANGE]
 04E8  89 07                             MOV   [BX],AX
 04EA  EB 12 90                          JMP   DATA_END

 04ED  BB 000A           DATA_DIV_10:    MOV   BX,10
 04F0  BA 0000                           MOV   DX,0000H       ;DX FOR SAVE DIV REMAINDER IF
 04F3  F7 F3                             DIV   BX             ;NO ZERO OVERFLOW.
 04F5  A3 2088 R                         MOV   [RANGE_1],AX

 04F8  B8 0003                           MOV   AX,03H
 04FB  A3 208E R                         MOV   [N],AX

 04FE  C3                DATA_END:       RET

                         DATA_ENLARGE ENDP
                         ;
                         ;
 04FF                    DATA_RANGE      PROC  NEAR           ;SUBROUTINE No.7 FOR THE RTDP.

 04FF  8D 1E 2088 R                      LEA   BX,RANGE_1     ;SENSORS' DATA STORE IN [RANGE_1]
 0503  8B 07                             MOV   AX,[BX]        ;BEFOR RANGE DETERMINATION.
 0505  3D 0110                           CMP   AX,272
 0508  72 0A                             JB    LOOK           ;IF DATA BELOW 271 LOOK_UP TABLE
 050A  3D 02E3                           CMP   AX,739         ;(Lnx)
 050D  72 16                             JB    LOOK_1         ;IF DATA BELOW 739
 050F  3D 03E8                           CMP   AX,1000        ;& ABOUVE 271 +10000.
 0512  72 22                             JB    LOOK_2         ;IF DATA BELOW 999
                                                              ;& ABOUVE 739 +20000.
 0514  E8 06BC R         LOOK:           CALL  LOOK_UP
 0517  8D 1E 20B6 R                      LEA   BX,TABLE_M+4
 051B  8B 07                             MOV   AX,[BX]
 051D  0D 0030                           OR    AX,'0'
 0520  89 07                             MOV   [BX],AX
 0522  EB 20 90                          JMP   LOOK_END

 0525  E8 06BC R         LOOK_1:         CALL  LOOK_UP
 0528  8D 1E 20B6 R                      LEA   BX,TABLE_M+4
 052C  8B 07                             MOV   AX,[BX]
 052E  0D 0031                           OR    AX,'1'         ;ADD       AX,10000
 0531  89 07                             MOV   [BX],AX
 0533  EB 0F 90                          JMP   LOOK_END

 0536  E8 06BC R         LOOK_2:         CALL  LOOK_UP
 0539  8D 1E 20B6 R                      LEA   BX,TABLE_M+4
 053D  8B 07                             MOV   AX,[BX]
 053F  0D 0032                           OR    AX,'2'         ;ADD       AX,20000
 0542  89 07                             MOV   [BX],AX
                                         ;---------------------------------------------;
                                         ;THE NEXT PART IS LnY=LnY/10^n+Ln10^n        ;
                                         ;FIRST STEP TABLE_M 0 -- 7  ASCII TO BCD     ;
                                         ;SECOND STEP +Ln10^n                         ;
                                         ;---------------------------------------------;
 0544  B9 0004           LOOK_END:       MOV   CX,4
 0547  8D 1E 20B2 R                      LEA   BX,TABLE_M     ; ASCII --- BCD
 054B  8B 07             AS_BCD:         MOV   AX,[BX]
 054D  25 0F0F                           AND   AX,0F0FH
 0550  89 07                             MOV   [BX],AX
 0552  43                                INC   BX
 0553  43                                INC   BX
 0554  E2 F5                             LOOP  AS_BCD

 0556  B8 0000                           MOV   AX,00
 0559  3B 06 208E R                      CMP   AX,[N]
 055D  74 15                             JZ    N_0
 055F  40                                INC   AX
 0560  3B 06 208E R                      CMP   AX,[N]
```

```
0564  74 11                        JZ      N_1
0566  40                           INC     AX
0567  3B 06 208E R                 CMP     AX,[N]
056B  74 17                        JZ      N_2
056D  40                           INC     AX
056E  3B 06 208E R                 CMP     AX,[N]
0572  74 1D                        JZ      N_3

0574  EB 25 90          N_0:       JMP     D_END

0577  8D 1E 20B2 R      N_1:       LEA     BX,TABLE_M
057B  BE CE10                      MOV     SI,0CE10H      ;LEA       SI,Ln10_1+100H
057E  E8 05CD R                    CALL    NINE_BYTE_ADD
0581  EB 18 90                     JMP     D_END

0584  8D 1E 20B2 R      N_2:       LEA     BX,TABLE_M
0588  BE CE18                      MOV     SI,0CE18H      ;LEA       SI,Ln10_2+100H
058B  E8 05CD R                    CALL    NINE_BYTE_ADD
058E  EB 0B 90                     JMP     D_END

0591  8D 1E 20B2 R      N_3:       LEA     BX,TABLE_M
0595  BE CE20                      MOV     SI,0CE20H      ;LEA       SI,Ln10_3+100H
0598  E8 05CD R                    CALL    NINE_BYTE_ADD

059B  8D 1E 208E R      D_END:     LEA     BX,N
059F  B8 0000                      MOV     AX,00H
05A2  89 07                        MOV     [BX],AX

05A4  B9 0004                      MOV     CX,04
05A7  8D 1E 20B2 R                 LEA     BX,TABLE_M     ; BCD --- ASCII
05AB  8B 07             BCD_AS:    MOV     AX,[BX]
05AD  0D 3030                      OR      AX,3030H
05B0  89 07                        MOV     [BX],AX
05B2  43                           INC     BX
05B3  43                           INC     BX
05B4  E2 F5                        LOOP    BCD_AS

05B6  C3                           RET

                        DATA_RANGE  ENDP
                        ;
                        ;
05B7                    EIGHT_BYTE_ADD PROC  NEAR        ;SUBROUTINE No. 8 FOR THE RTDP.
                               ;-------------------------------------------------------------;
                               ; BEFOR CALL THIS PROCEDURE                                   ;
                               ; 1. TOW ADDRESS (LSB) SHOULD BE MOV INTO BX & SI RESPECTIVELY.;
                               ; 2. THE RESULT IS SENT TO [BX]                               ;
                               ; 3. AFTER CALLING SI POINT TO [BX]+7                         ;
                               ; 4. BEFOR CALLING THIS PROCEIDURE PUSH CX IS RECONMANDED.    ;
                               ;-------------------------------------------------------------;

05B7  8A 07                        MOV     AL,BYTE PTR[BX] ;LSB IS SENT TO AL
05B9  02 04                        ADD     AL,BYTE PTR[SI] ;ADD AL WITH THE BYTE POINTED
                                                           ;BY SI
05BB  37                           AAA                     ;ASCII ADD ADJUSTED
05BC  88 07                        MOV     [BX],AL         ;RESULT IN [BX]
05BE  B9 0007                      MOV     CX,07H          ;ANOTHER 7 BYTE OTHER THAN LSB
05C1  43                HJ:        INC     BX              ;POINT TO NEXT BYTE
05C2  46                           INC     SI
05C3  8A 07                        MOV     AL,BYTE PTR[BX] ;NEXT BYTE TO AL
05C5  12 04                        ADC     AL,BYTE PTR[SI] ;ADD WITH CARRY
05C7  37                           AAA                     ;ADJUST
05C8  88 07                        MOV     [BX],AL         ;RESULT IN [BX]
05CA  E2 F5                        LOOP    HJ

05CC  C3                           RET
```

```
                           EIGHT_BYTE_ADD   ENDP
                           ;
                           ;
 05CD                      NINE_BYTE_ADD PROC    NEAR          ;SUBROUTINE No. 9 FOR THE RTDP.
                                ;------------------------------------------------------------;
                                ; BEFOR CALL THIS PROCEDURE                                  ;
                                ; 1. TOW ADDRESS (LSB) SHOULD BE MOV INTO BX & SI RESPECTIVE ;
                                ; 2. THE RESULT IS SENT TO [BX]                              ;
                                ; 3. BEFOR CALLING THIS PROCEIDURE PUSH CX IS RECONMANDED.   ;
                                ;------------------------------------------------------------;

 05CD  8A 07                       MOV     AL,BYTE PTR[BX] ;LSB IS SENT TO AL
 05CF  02 04                       ADD     AL,BYTE PTR[SI] ;ADD AL WITH THE BYTE POINTED BY
 05D1  37                          AAA                     ;ASCII ADD ADJUSTED
 05D2  88 07                       MOV     [BX],AL         ;RESULT IN [BX]
 05D4  B9 0008                     MOV     CX,08H          ;ANOTHER 7 BYTE OTHER THAN LSB
 05D7  43                  HIJ:    INC     BX              ;POINT TO NEXT BYTE
 05D8  46                          INC     SI
 05D9  8A 07                       MOV     AL,BYTE PTR[BX] ;NEXT BYTE TO AL
 05DB  12 04                       ADC     AL,BYTE PTR[SI] ;ADD WITH CARRY
 05DD  37                          AAA                     ;ADJUST
 05DE  88 07                       MOV     [BX],AL         ;RESULT IN [BX]
 05E0  E2 F5                       LOOP    HIJ

 05E2  C3                          RET

                           NINE_BYTE_ADD   ENDP
                           ;
                           ;
 05E3                      SIXTEEN_BYTE_ADD PROC    NEAR       ;SUBROUTINE No. 10 FOR THE RTDP.
                                ;--------------------------------------------------------;
                                ; 1. BEFOR CALL THIS PROCEDURE TOW ADDRESS (LSB) SHOULD BE ;
                                ;    MOV INTO BX & SI RESPECTIVELY.                      ;
                                ; 2. THE RESULT IS SENT TO [BX]. .                        ;
                                ; 3. BEFOR CALLING THIS PROCEIDURE PUSH CX IS RECONMANDED. ;
                                ;--------------------------------------------------------;

 05E3  8A 07                       MOV     AL,BYTE PTR[BX] ;LSB IS SENT TO AL
 05E5  02 04                       ADD     AL,BYTE PTR[SI] ;ADD AL WITH THE BYTE POINTED
                                                           ;BY SI
 05E7  37                          AAA                     ;ASCII ADD ADJUSTED
 05E8  88 07                       MOV     [BX],AL         ;RESULT IN [BX]
 05EA  B9 000F                     MOV     CX,15           ;ANOTHER 15 BYTE OTHER THAN LSB
 05ED  43                  HHJ:    INC     BX              ;POINT TO NEXT BYTE
 05EE  46                          INC     SI
 05EF  8A 07                       MOV     AL,BYTE PTR[BX] ;NEXT BYTE TO AL
 05F1  12 04                       ADC     AL,BYTE PTR[SI] ;ADD WITH CARRY
 05F3  37                          AAA                     ;ADJUST
 05F4  88 07                       MOV     [BX],AL         ;RESULT IN [BX]
 05F6  E2 F5                       LOOP    HHJ

 05F8  C3                          RET

                           SIXTEEN_BYTE_ADD        ENDP
                           ;
                           ;
 05F9                      EIGHT_BYTE_SUB PROC    NEAR         ;SUBROUTINE No. 11 FOR THE RTDP.
                                   ;-----------------------------------------------------;
                                   ; BEFOR CALL THIS PROCEDURE TOW ADDRESSES SHOULD BE    ;
                                   ; MOV INTO BX & SI RESPECTIVELY. THE RESULT IS IN [BX].;
                                   ; BEFOR CALLING THIS PROCEIDURE PUSH CX IS RECONMANDED ;
                                   ;            [BX] - [SI] => [BX]                      ;
                                   ;-----------------------------------------------------;
 05F9  8A 07                       MOV     AL,BYTE PTR[BX] ;LSB IS SENT TO AL
 05FB  2A 04                       SUB     AL,BYTE PTR[SI] ;SUBTRACT AL WITH THE BYTE
                                                           ;POINTED BY SI
 05FD  3F                          AAS                     ;ASCII ADJUST FOR SUBTRUCTION
```

```
05FE  88 07                        MOV    [BX],AL           ;RESULT IN [BX]
0600  B9 0007                      MOV    CX,07H            ;ANOTHER 7 BYTE OTHER THAN LSB
0603  43                 THJ:      INC    BX                ;POINT TO NEXT BYTE
0604  46                           INC    SI
0605  8A 07                        MOV    AL,BYTE PTR[BX]   ;NEXT BYTE TO AL
0607  1A 04                        SBB    AL,BYTE PTR[SI]   ;SUBTRUCT WITH BORROW
0609  3F                           AAS                      ;ASCII ADJUST FOR SUBTRUCTION
060A  88 07                        MOV    [BX],AL           ;RESULT IN [BX]
060C  E2 F5                        LOOP   THJ

060E  C3                           RET

                         EIGHT_BYTE_SUB         ENDP
                         ;
                         ;
060F                     SIXTEEN_BYTE_SUB PROC    NEAR      ;SUBROUTINE No. 12 FOR THE RTDP.
                                   ;-----------------------------------------------------;
                                   ; BEFOR CALL THIS PROCEDURE TOW ADDRESSES SHOULD BE    ;
                                   ; MOV INTO BX & SI RESPECTIVELY. THE RESULT IS IN [BX].;
                                   ; BEFOR CALLING THIS PROCEIDURE PUSH CX IS RECONMANDED ;
                                   ;          [BX] - [SI] => [BX]                         ;
                                   ;-----------------------------------------------------;
060F  8A 07                        MOV    AL,BYTE PTR[BX]   ;LSB IS SENT TO AL
0611  2A 04                        SUB    AL,BYTE PTR[SI]   ;SUBTRACT AL WITH THE BYTE
                                                            ;POINTED BY SI
0613  3F                           AAS                      ;ASCII ADJUST FOR SUBTRUCTION
0614  88 07                        MOV    [BX],AL           ;RESULT IN [BX]
0616  B9 000F                      MOV    CX,15             ;ANOTHER 15 BYTE OTHER THAN LSB
0619  43                 GHJ:      INC    BX                ;POINT TO NEXT BYTE
061A  46                           INC    SI
061B  8A 07                        MOV    AL,BYTE PTR[BX]   ;NEXT BYTE TO AL
061D  1A 04                        SBB    AL,BYTE PTR[SI]   ;SUBTRUCT WITH BORROW
061F  3F                           AAS                      ;ASCII ADJUST FOR SUBTRUCTION
0620  88 07                        MOV    [BX],AL           ;RESULT IN [BX]
0622  E2 F5                        LOOP   GHJ

0624  C3                           RET

                         SIXTEEN_BYTE_SUB         ENDP
                         ;
                         ;
0625                     EIGHT_MUL_ONE   PROC   NEAR        ;SUBROUTINE No. 13 FOR THE RTDP.
                              ;--------------------------------------------------------------;
                              ; THIS PROCEDURE WAS FOR 8 DIGITS X 1 DIGIT ASCII MULTIPLICATION.;
                              ; THE ADDRESS OF THE 8 DIGIT DATA MUST BE SENT TO SI AND 1 DIGIT ;
                              ; MUST BE SENT TO DL REGISTER, BEFOR CALLING THIS PROCEDURE.     ;
                              ; THE RESULT WAS SENT TO [PAR_PRODUCT].                          ;
                              ; PUSH AX BX CX  DX SI ARE RECOMMANDED BEFOR CALLING.            ;
                              ;--------------------------------------------------------------;
0625  8D 1E 20C2 R                 LEA    BX,PAR_PRODUCT    ;POINT TO PARTIAL PRODUCT
0629  B8 0000                      MOV    AX,00H
062C  8A 04                        MOV    AL,[SI]           ;LSB IS AT [SI+0]
062E  F6 E2                        MUL    DL                ;1 DIGIT IS AT DL
0630  D4 0A                        AAM                      ;ASCII ADJUSTMENT FOR
                                                            ;MULTIPLICATION
0632  88 07                        MOV    [BX],AL           ;BX POINTS TO PAR_PRODUCT
0634  8A F4                        MOV    DH,AH             ;KEEP HIGH DIGIT PRODUCED
                                                            ;BY AAM IN DH
0636  B9 0008                      MOV    CX,08H            ;1 OF 8 DIGITS HAS BEEN
                                                            ;CALCULATED, BUT THE CARRY MUST
                                                            ;BE CONSIDERDED
0639  46                 RE_CAL:   INC    SI                ;POINT TO NEXT DIGIT
063A  8A 04                        MOV    AL,[SI]
063C  F6 E2                        MUL    DL
063E  D4 0A                        AAM                      ;ASCII ADJUSTMENT FOR
                                                            ;MULTIPLICATION
0640  02 C6                        ADD    AL,DH             ;NEXT LSB IN AL
```

```
0642  37                        AAA                          ;ASCII ADJUSTMENT FOR ADDATION
0643  43                        INC     BX                   ;POINT TO NEXT LSB OF PAR_PRODUCT
0644  88 07                     MOV     [BX],AL
0646  8A F4                     MOV     DH,AH                ;KEEP HIGH DIGIT PRODUCED
                                                             ;BY AAM IN DH
0648  E2 EF                     LOOP    RE_CAL

064A  C3                        RET

                  EIGHT_MUL_ONE    ENDP
                  ;
                  ;
064B              EIGH_MUL_EIGH   PROC    NEAR          ;SUBROUTINE No. 14 FOR THE RTDP.
                  ;----------------------------------------------------------------;
                  ;THIS PROCEDURE WAS FOR TOW EIGH-DIGIT ASCII (BCD) MULTIPLICATION   ;
                  ;  1.16 UNITS ARE RESERVED FOR PARTIAL PRODUCTS [PAR_PRODUCT]+0 -- +15 ;
                  ;  2.AFTER PARTIAL PRODUCTS ARE ADDED TOGETHER. (THIS RESULTS WERE    ;
                  ;    CALLED INTERMIDIATE AND STORED IN [INTERMIDIATE] +0 -- +15)      ;
                  ;  3.THE PROCEDURE WOULD CHECK THE 5th DIGIT OF THE FRACTIONAL PART,  ;
                  ;    IF IT WAS < 4 THEN OMITTDED, IF IT WAS >6 THEN 4th DIGIT OF THE  ;
                  ;    FRACTIONAL PART PLUS 1.                                         ;
                  ;  4.THE PRODUCT OF MULTIPLICATION WAS STORED IN A 16 DIGIT UNITS    ;
                  ;    [PRODUCT]+0 -- +15.                                             ;
                  ;  5.THE FINAL RESULT WOULD BE SENT TO [PRODUCT]                     ;
                  ;----------------------------------------------------------------;
                  ;    [SI](8 DIGITS) * [DI](8 DIGITS) = [BX](16 DIGITS)                ;
                  ;    [INTERMIDIATE]+PAR_PRODUCT+...+ => 4 OMMI; 6 PLUS --> [PRODUCT]  ;
                  ;----------------------------------------------------------------;
064B  8D 1E 20D2 R              LEA     BX,INTERMIDIATE
064F  B9 0008                   MOV     CX,08H
0652  8A 15       RE_MULTI:     MOV     DL,[DI]              ;LSB TO DL
0654  53                        PUSH    BX                   ;[BX] = ADDRESS: INTERMIDIATE
0655  51                        PUSH    CX
0656  56                        PUSH    SI
0657  E8 0625 R                 CALL    EIGHT_MUL_ONE        ;THE RESULT IS AT [PAR_PRODUCT]
065A  5E                        POP     SI
065B  59                        POP     CX
065C  5B                        POP     BX                   ;[BX] = ADDRESS: INTERMIDIATE

065D  53                        PUSH    BX
065E  51                        PUSH    CX
065F  56                        PUSH    SI
0660  8D 36 20C2 R              LEA     SI,PAR_PRODUCT       ;PART PRODUCTION ADD PROPAIRATION
0664  E8 05CD R                 CALL    NINE_BYTE_ADD        ;RESULT WAS IN
                                                             ;[INTERMIDIATION]=[[BX]]
0667  5E                        POP     SI
0668  59                        POP     CX
0669  5B                        POP     BX
066A  43                        INC     BX                   ;POINT TO NEXT LSB OF
                                                             ;INTERMIDINATION
066B  47                        INC     DI                   ;POINT TO NEXT LSB OF "* XXXXXXX"

066C  E2 E4                     LOOP    RE_MULTI

066E  8D 1E 20D2 R              LEA     BX,INTERMIDIATE ;4 OMMITED, 6 PLUS 1 IN 4th DIGIT
0672  B8 0000                   MOV     AX,00H
0675  8A 47 03                  MOV     AL,BYTE PTR[BX+3] ;TEST 5th DIGIT
0678  3D 0005                   CMP     AX,05H
067B  72 02                     JB      OMMIT                ;BELOW OR EQUAL 5 OMMITED
067D  77 11                     JA      PLUS                 ;ABOUVE 5 PIUS 1 ON 4th DIGIT

067F  FC          OMMIT:        CLD                          ;SET DF=0 TO MOVE FORWARD
0680  8D 36 20D6 R              LEA     SI,INTERMIDIATE+4
0684  8D 3E 20E2 R              LEA     DI,PRODUCT
0688  B9 000C                   MOV     CX,12
068B  F3/ A4      REP           MOVSB                        ;MOVES 12 BYTE FROM [SI] TO [DI]
068D  EB 1E 90                  JMP     EME
```

```
0690  FC                PLUS:         CLD                          ;SET DF=0 TO MOVE FORWARD
0691  8D 36 20D6 R                    LEA     SI,INTERMIDIATE+4
0695  8D 3E 20E2 R                    LEA     DI,PRODUCT
0699  B9 000C                         MOV     CX,12
069C  F3/ A4            REP           MOVSB                        ;MOVES 12 BYTE FROM [SI] TO [DI]

069E  8D 36 20F2 R                    LEA     SI,FIVE_PLUS     ;ONLY 1 IN THE LSB [FIVE_PLUS+0]
06A2  8D 1E 20E2 R                    LEA     BX,PRODUCT
06A6  B0 01                           MOV     AL,01H
06A8  88 04                           MOV     [SI],AL
06AA  E8 05E3 R                       CALL    SIXTEEN_BYTE_ADD ;THE DATA AT [PRODUCT] LOOKS
                                                              ;LIKE
                                                     ; X X X X . X X X X X _ _ _ _ _ _ _
                                                     ;LSB    POINT                MSB

06AD  B9 0010           EME:          MOV     CX,16        ;** AFTER CALL EIGH_MUL_EIGH MUST**
06B0  B0 00                           MOV     AL,00H       ;CLEAR INTERMIDIATE
06B2  8D 1E 20D2 R                    LEA     BX,INTERMIDIATE
06B6  88 07             C_INER:       MOV     [BX],AL
06B8  43                              INC     BX
06B9  E2 FB                           LOOP    C_INER

06BB  C3                              RET

                        EIGH_MUL_EIGH   ENDP
                        ;
                        ;
06BC                    LOOK_UP       PROC    NEAR             ;SUBROUTINE No. 15 FOR THE RTDP.
                                      ;--------------------------------------------------;
                                      ; DATA IN [RANGE] MUL BY 4 PLUS A_0 -- TABLE ADDRESS ;
                                      ; AFTER LOOK UP Lnx STORE IN TABLE_M TEMPERORYLY    ;
                                      ;--------------------------------------------------;
                                      MOV     BX,0C000H        ;LEA BX,A_0+100H THE EFFECTIVE
                                      PUSH    BX               ;ADDRESS OF FIRST WORD
                                      LEA     BX,RANGE_1       ;ENLARGED ORANGINAL SENSOR DATA
                                      MOV     AX,[BX]          ;SENSORS DATA = RELATIVE ADDR IN DI
                                      SUB     AX,100           ;THE FIRST ADDRESS OF THE TABLE 100
                                      MOV     BX,4             ;4 ASCII CHARECTERS
                                      MUL     BX               ;VALU*4= REAL ADDRESS
                                      MOV     DI,AX            ;OFFSET IN DI
                                      POP     BX               ;TABLE BIGINNING ADDRESS
                                      MOV     DX,WORD PTR[BX][DI] ; FETCH **__
06..  .A E2                           MOV     AH,DL            ;EXCANGE ASCII BITS
06D5  8A C6                           MOV     AL,DH
06D7  8B D0                           MOV     DX,AX
06D9  89 16 20B4 R                    MOV     [TABLE_M+2],DX ;Lnx IS AT TABLE_M+2 (MSB)
06DD  47                              INC     DI               ;POINT TO NEXT WORD
06DE  47                              INC     DI
06DF  8B 11                           MOV     DX,WORD PTR[BX][DI] ; FETCH __**
06E1  8A E2                           MOV     AH,DL            ;EXCHANGE ASCII BITS
06E3  8A C6                           MOV     AL,DH
06E5  8B D0                           MOV     DX,AX
06E7  89 16 20B2 R                    MOV     [TABLE_M],DX   ;Lnx IS AT TABLE_M

06EB  C3                              RET

                        LOCK_UP       ENDP
                        ;
                        ;
06EC                    SIGMA_LnYi    PROC    NEAR             ;SUBROUTINE No. 16 FOR THE RTDP.
                              ;----------------------------------------------------------;
                              ; THIS SUBROUTINE WAS FOR CALCULATE SIGMA LnYi (i=0 -- n)   ;
                              ; 1. THE RESULT OF THE CALCULATION ARE STORE IN SIGMA_LnYi_M ;
                              ; 2. BEFOR RET SI POINT TO NEXT Yi AUTOMATICALLY.           ;
                              ;----------------------------------------------------------;
06EC  B9 0003                         MOV     CX,03            ;FOUR SENSORS ARE IN ONE LINE
```

```
 06EF  8D 1E 2102 R                     LEA     BX,SIGMA_LnYi_M ;SIGMA RESULT ADDRESS IN BX
 06F3  8D 36 23A4 R                     LEA     SI,Z_MEAN_1_L   ;LnYi ADDRESS

 06F7  51               AGAIN_ADD:      PUSH    CX              ;EIGH_BYTE_ADD USE CX
 06F8  E8 05B7 R                        CALL    EIGHT_BYTE_ADD  ;8 DIGIT ADD
 06FB  59                               POP     CX
 06FC  8D 1E 2102 R                     LEA     BX,SIGMA_LnYi_M ;SIGMA RESULT ADDRESS IN BX
 0700  46                               INC     SI              ;POINT TO NEXT SOURCE BYTE
 0701  E2 F4                            LOOP    AGAIN_ADD

 0703  C3                               RET

                        SIGMA_LnYi      ENDP
                        ;
                        ;
 0704                   MEAN_VALUE_Yi PROC        NEAR          ;SUBROUTINE No. 17 FOR THE RTDP.
                                        ;-------------------------------------------------------;
                                        ;THIS SUBROUTINE WAS FOR LnYi MEAN VALUE CALCULATION.   ;
                                        ; 1. IT WAS 8 DIGITS DIVIDED BY 1 DIGIT ASCII DIVISION  ;
                                        ; 2. THE SEQUENCE OF CALCULATION WAS FROM MSB TO LSB    ;
                                        ; 3. THE RESULT WOULD BE PUT IN THE MEAN_LnYi.          ;
                                        ; 4. BEFOR CALLING THIS PROCEIDURE PUSH CX WAS          ;
                                        ;    RECONMANDED.                                       ;
                                        ;    NOMINATOR / DENOMINATOR = QUOTINET + REMAINDER     ;
                                        ;-------------------------------------------------------;
 0704  8D 36 2109 R                     LEA     SI,SIGMA_LnYi_M+7 ;POINT TO SOURSE ADDRESS (MSB)
 0708  8D 1E 2119 R                     LEA     BX,MEAN_LnYi+7  ;POINT TO RESULT UNIT ADDR (MSB)
 070C  B8 0003                          MOV     AX,03           ;DENOMINATOR REPRESENTED
 070F  A3 2090 R                        MOV     [N_DIVIDE],AX   ;THE SENSORS No. 4 SENSORS
 0712  B4 00                            MOV     AH,00H          ;CLEAR AH FOR THE REMAINDER
 0714  8A 04                            MOV     AL,BYTE PTR[SI] ;NOMINATOR IN AL

 0716  B9 0008                          MOV     CX,08H          ;NOMINATOR IS A 8 DIGIT ASCII
                                                                ;NUMBER
 0719  F6 36 2090 R     AGAIN_DIVIDE:DIV        BYTE PTR[N_DIVIDE] ;DIVIDING
 071D  88 07                            MOV     [BX],AL         ;SAVE THE QUOTIENT
 071F  4B                               DEC     BX              ;POINT TO NEXT RESULT
 0720  4E                               DEC     SI              ;POINT TO NEXT DENOMINATOR
 0721  8A 04                            MOV     AL,BYTE PTR[SI] ;NEXT DIGIT IS SENT TO AL
 0723  D5 0A                            AAD                     ;ASCII ADJUSTMENT FOR DIVIDE
                                                                ;(AL)*10+(AL), (AH) <-- 0
 0725  E2 F2                            LOOP    AGAIN_DIVIDE

 0727  C3                               RET

                        MEAN_VALUE_Yi  ENDP                     ;SUBROUTINE No.  FOR THE RTDP
                        ;
                        ;
 0728                   MEAN_VALUE_Xi PROC      NEAR            ;SUBROUTINE No. 18 FOR THE RTDP
                                        ;-------------------------------------------------------;
                                        ; 1. DU TO Xi ARE FIXT POSITIONS IN LOGITUDINAL DIRECTION. ;
                                        ; 2. MEAN VALUE OF Xi WAS CALCULATED BY HAND.           ;
                                        ; 3. IN THIS SUBROUTINE MEANVALUE OF Xi WAS SIMPLY MOVED ;
                                        ;    IN TO [MEAN_Xi+0] -- [MEAN_Xi+7]. i.e. 0000.5050   ;
                                        ; 4. Xi ARE ALSO SENT TO X1 -- X3.                      ;
                                        ;-------------------------------------------------------;
                                        ;PART 1: STORE MEAN_Xi = 0000.5050
 0728  8D 1E 2172 R                     LEA     BX,MEAN_Xi      ;POINT TO RESULT UNIT (MSB)
 072C  B8 0500                          MOV     AX,0500H
 072F  89 07                            MOV     [BX],AX
 0731  43                               INC     BX
 0732  43                               INC     BX
 0733  B8 0500                          MOV     AX,0500H        ;00 00 00 00 . 05 00 05 00
 0736  89 07                            MOV     [BX],AX

                              ;PART 2: STORE X1=0000.2270; X2=0000.4610; X3=0000.8270;
                              ;                 X0=0000.F26FH
```

```
0738  8D 1E 2122 R                    LEA     BX,X1
073C  B8 0700                         MOV     AX,0700H
073F  89 07                           MOV     [BX],AX
0741  43                              INC     BX
0742  43                              INC     BX
0743  B8 0202                         MOV     AX,0202H
0746  89 07                           MOV     [BX],AX          ;X1=0000.2270

0748  8D 1E 2132 R                    LEA     BX,X2
074C  B8 0100                         MOV     AX,0100H
074F  89 07                           MOV     [BX],AX
0751  43                              INC     BX
0752  43                              INC     BX
0753  B8 0406                         MOV     AX,0406H
0756  89 07                           MOV     [BX],AX          ;X2=0000.4610

0758  8D 1E 2142 R                    LEA     BX,X3
075C  B8 0700                         MOV     AX,0700H
075F  89 07                           MOV     [BX],AX
0761  43                              INC     BX
0762  43                              INC     BX
0763  B8 0802                         MOV     AX,0802H
0766  89 07                           MOV     [BX],AX          ;X3=0000.8270

0768  8D 1E 2152 R                    LEA     BX,X0
076C  B8 F26F                         MOV     AX,0F26FH        ;0.F26FH = 0.9470
076F  89 07                           MOV     [BX],AX

0771  8D 1E 2162 R                    LEA     BX,SIGMA_Xi_M
0775  B8 0500                         MOV     AX,0500H         ;SIGMA_Xi_M = 1.5150
0778  89 07                           MOV     [BX],AX
077A  43                              INC     BX
077B  43                              INC     BX
077C  B8 0501                         MOV     AX,0501H
077F  89 07                           MOV     [BX],AX
0781  43                              INC     BX
0782  43                              INC     BX
0783  B8 0001                         MOV     AX,0001H
0786  89 07                           MOV     [BX],AX

0788  C3                              RET

                      MEAN_VALUE_Xi ENDP
                      ;
                      ;
0789                  D_NOMINATION PROC      NEAR             ;SUBROUTINE No. 19 FOR THE RTDP
                        ;----------------------------------------------------------------;
                        ; THIS SUBROUTINE WAS FOR CALCULATING NOMINATION OF THE PRAMETER D ;
                        ;      SIGMA (Xi - Xmean) (Yi - Ymean), i= 1 -- n                   ;
                        ; 1. THE RESULT WAS STORED IN SIG_Xi_Xm_Yi_Ym ( 8W, 16 BYTE RAM )  ;
                        ; 2. THE INTERMIDIATES WERE STORED IN (Xi-Xm) & (Yi-Ym) RESPECTIVELY;
                        ; 3. AFTER FINISHING (Xi-Xm)*(Yi-Ym) THE PRODUCT WAS ADDED ON      ;
                        ;    SIG_Xi_Xm_Yi_Ym .                                             ;
                        ; 4. THE SIGN OF D NOMINATION ONLY DEPENDED ON THE VALUE OF        ;
                        ;    [SIGN_SIGMA], THAT WAS THE SIGN OF (Xi-Xm)(Yi-Ym).            ;
                        ;----------------------------------------------------------------;
0789  B9 0003                         MOV     CX,3             ;3 SENSORS IN A LINE
078C  8D 1E 2122 R                    LEA     BX,X1            ;X1
0790  89 1E 2092 R                    MOV     [Xi_M],BX        ;SAVE ADDRESS FOR FORTHER USE
0794  8D 1E 23A4 R                    LEA     BX,Z_MEAN_1_L    ;LnYi
0798  89 1E 2094 R                    MOV     [LnYi_M],BX      ;SAVE ADDRESS FOR FORTHER USE
079C  8D 1E 21D2 R                    LEA     BX,Xi_Xm         ;Xi-Xm
07A0  89 1E 2099 R                    MOV     [Xi_Xm_M],BX     ;SAVE ADDRESS FOR FORTHER USE

07A4  51              SG_AGAIN:       PUSH    CX               ;SAVE CX

                        ;-------- CALCULATE (Xi - Xmean) ---------------------------------;
```

```
07A5  B9 0008                          MOV     CX,8              ;MOVE Xi TO INTERMIDIATE_SUB FOR
07A8  8D 3E 2192 R                     LEA     DI,INTERMIDIATE_SUB ;SUB
07AC  8B 36 2092 R                     MOV     SI,[Xi_M]
07B0  F3/ A4                REPZ       MOVSB

07B2  8D 1E 2192 R                     LEA     BX,INTERMIDIATE_SUB ;GET Xi ADDRESS
07B6  8D 36 2172 R                     LEA     SI,MEAN_Xi
07BA  E8 05F9 R                        CALL    EIGHT_BYTE_SUB  ;[BX] - [SI] => [BX]

07BD  8A 07                            MOV     AL,BYTE PTR[BX] ;TEST MSB
07BF  3C 09                            CMP     AL,09H            ;EQUALS TO 9?
07C1  75 42                            JNZ     S_RESULT_X        ;NO, SEND RESULT

07C3  8D 3E 2182 R                     LEA     DI,COMPLETE       ;YES RESULT IS MINUS ( - )
07C7  C6 45 08 01                      MOV     BYTE PTR[DI+8],01H ;PROPARE FOR SUBTRUCT
07CB  8B DF                            MOV     BX,DI
07CD  8D 36 2192 R                     LEA     SI,INTERMIDIATE_SUB ;[BX] - [SI] = [BX]
07D1  E8 05F9 R                        CALL    EIGHT_BYTE_SUB  ;CALCU. ABSOLUT VALU

07D4  B9 0008                          MOV     CX,8              ;8 BYTE MOVING
07D7  8D 3E 21B2 R                     LEA     DI,Xi_Xm          ;RESULT DISTINATION
07DB  8D 36 2182 R                     LEA     SI,COMPLETE       ;GET 100,000,000 ADDRESS
07DF  F3/ A4                REPZ       MOVSB                     ;RESULT NOW IN (Xi_Xm)

07E1  B9 0008                          MOV     CX,8              ;8 BYTE MOVING
07E4  8B 3E 2099 R                     MOV     DI,[X1_Xm_M]      ;DISTINATION FOR CALCU (Xi-Xm)^2
07E8  8D 36 2182 R                     LEA     SI,COMPLETE       ;GET 100,000,000 ADDRESS
07EC  F3/ A4                REPZ       MOVSB                     ;RESULT NOW IN (X1_Xm)
                                                                 ;OR (X2-Xm)...
07EE  B9 0010                          MOV     CX,16
07F1  8D 1E 2182 R                     LEA     BX,COMPLETE       ;CLEAR [COMPLEATE] FOR NEXT TIME
07F5  C6 07 00              CAA:       MOV     [BX],BYTE PTR 00H ;USE
07F8  43                               INC     BX
07F9  E2 FA                            LOOP    CAA

07FB  8D 3E 2096 R                     LEA     DI,SIGN_X         ;SIGN OF SUB RESULT
07FF  C6 05 01                         MOV     BYTE PTR[DI],01H ;IF 1 MINUS
0802  EB 22 90                         JMP     CAL_Y

0805  8D 3E 2096 R          S_RESULT_X: LEA    DI,SIGN_X
0809  C6 05 00                         MOV     BYTE PTR[DI],0

080C  B9 0008                          MOV     CX,8              ;8 BYTE MOVING
080F  8D 3E 21B2 R                     LEA     DI,Xi_Xm          ;RESULT DISTINATION
0813  8D 36 2192 R                     LEA     SI,INTERMIDIATE_SUB ;GET Xi-Xm ADDRESS
0817  F3/ A4                REPZ       MOVSB                     ;RESULT NOW IN (Xi_Xm)

0819  B9 0008                          MOV     CX,8              ;8 BYTE MOVING
081C  8B 3E 2099 R                     MOV     DI,[X1_Xm_M]      ;DISTINATION FOR CALCU (Xi-Xm)^2
0820  8D 36 2192 R                     LEA     SI,INTERMIDIATE_SUB ;GET Xi-Xm ADDRESS
0824  F3/ A4                REPZ       MOVSB                     ;RESULT NOW IN (X1_Xm)
                                                                 ;OR (X2-Xm)...

                            ;---------- CALCULATE (Yi - Ymean) ----------------------------------;

0826  B9 0008               CAL_Y:     MOV     CX,8              ;MOVE LnYi TO INTERMIDIATE_SUB
0829  8D 3E 2192 R                     LEA     DI,INTERMIDIATE_SUB ;FOR SUB
082D  8B 36 2094 R                     MOV     SI,[LnYi_M]
0831  F3/ A4                REPZ       MOVSB

0833  8D 1E 2192 R                     LEA     BX,INTERMIDIATE_SUB
0837  8D 36 2112 R                     LEA     SI,MEAN_LnYi
083B  E8 05F9 R                        CALL    EIGHT_BYTE_SUB  ;[BX] -[SI] => [BX]

083E  8A 07                            MOV     AL,BYTE PTR[BX] ;TEST MSB
0840  3C 09                            CMP     AL,09H            ;EQUALS TO 9?
```

```
0842  75 35                          JNZ     S_RESULT_Y        ;NO, SEND RESULT

0844  8D 3E 2182 R                   LEA     DI,COMPLETE       ;YES RESULT IS MINUS ( - )
0848  C6 45 08 01                    MOV     BYTE PTR[DI+8],01H ;PROPARE FOR SUBTRUCT
084C  8B DF                          MOV     BX,DI
084E  8D 36 2192 R                   LEA     SI,INTERMIDIATE_SUB
0852  E8 05F9 R                      CALL    EIGHT_BYTE_SUB

0855  B9 0008                        MOV     CX,8              ;8 BYTE MOVING
0858  8D 3E 21C2 R                   LEA     DI,Yi_Ym          ;RESULT DISTINATION
085C  8D 36 2182 R                   LEA     SI,COMPLETE       ;GET Xi ADDRESS
0860  F3/ A4              REPZ       MOVSB                     ;RESULT NOW IN (Xi_Xm)

0862  B9 0010                        MOV     CX,16
0865  8D 1E 2182 R                   LEA     BX,COMPLETE       ;CLEAR [COMPLEATE] FOR NEXT TIME
0869  C6 07 00            CAB:       MOV     [BX],BYTE PTR 00H ;USE
086C  43                             INC     BX
086D  E2 FA                          LOOP    CAB

086F  8D 3E 2097 R                   LEA     DI,SIGN_Y         ;SIGN OF SUB RESULT
0873  C6 05 01                       MOV     BYTE PTR[DI],01H ;IF 1 MINUS
0876  EB 15 90                       JMP     CAL_MUL

0879  8D 3E 2097 R        S_RESULT_Y: LEA    DI,SIGN_Y         ;NO, THE SIGN IS PLUS ( + )
087D  C6 05 00                       MOV     BYTE PTR[DI],0

0880  B9 0008                        MOV     CX,8              ;8 BYTE MOVING
0883  8D 3E 21C2 R                   LEA     DI,Yi_Ym          ;RESULT DISTINATION
0887  8D 36 2192 R                   LEA     SI,INTERMIDIATE_SUB ;GET LnYi ADDRESS
088B  F3/ A4              REPZ       MOVSB                     ;THE RESULT NOW IN (Yi_Ym)

                                     ;-----------------------------------------------------;
                                     ; NEXT PART WAS MULTIPLICATION                        ;
                                     ; BEFOR CALCULATING LOGIC CALCULATION MUST BE DONE i.e. ;
                                     ; SIGN_X XOR SIGN_Y IF = 0 THEN PLUS IF 1 THEN MINUS  ;
                                     ;-----------------------------------------------------;
088D  A0 2096 R           CAL_MUL:   MOV     AL,[SIGN_X]
0890  8A 0E 2097 R                   MOV     CL,[SIGN_Y]
0894  32 C1                          XOR     AL,CL
0896  8D 1E 2098 R                   LEA     BX,SIGN_SIGMA     ;[SIGN_SIGMA] IS THE SIGN OF
089A  88 07                          MOV     [BX],AL           ;Xi-Xm)(Yi-Ym)

089C  8D 36 21B2 R                   LEA     SI,Xi_Xm
08A0  8D 3E 21C2 R                   LEA     DI,Yi_Ym
08A4  8D 1E 20D2 R                   LEA     BX,INTERMIDIATE

08A8  E8 064B R                      CALL    EIGH_MUL_EIGH     ;(Xi - Xmean)*(Yi - Ymean)

                                     ;-----------------------------------------------;
                                     ;NEXT PART WAS CALCULATE SIGMA (Xi-Xm)(Yi-Yi)   ;
                                     ;-----------------------------------------------;
08AB  8D 1E 21A2 R                   LEA     BX,SIG_Xi_Xm_Yi_Ym ;RESULT ADDRESS
08AF  8D 36 20E2 R                   LEA     SI,PRODUCT        ;ADDRESS OF MULTIPLICATION

08B3  B0 00                          MOV     AL,00H
08B5  3A 06 2098 R                   CMP     AL,[SIGN_SIGMA]
08B9  75 06                          JNZ     MULTY_SUB         ;IF (Xi-Xm)(Yi-Ym)<0, THEN
                                                               ;SUBTRUCT
08BB  E8 05E3 R                      CALL    SIXTEEN_BYTE_ADD  ;ELSE ADDITION
08BE  EB 04 90                       JMP     FF
08C1  E8 060F R           MULTY_SUB: CALL    SIXTEEN_BYTE_SUB

08C4  8B 1E 2092 R        FF:        MOV     BX,[Xi_M]         ;GET THE XI ADDRESS
08C8  83 C3 10                       ADD     BX,16             ;POINT TO NEXT Xi
08CB  89 1E 2092 R                   MOV     [Xi_M],BX

08CF  8B 1E 2094 R                   MOV     BX,[LnYi_M]       ;GET THE LnYi ADDRESS
```

```
08D3  83 C3 08                         ADD      BX,08H          ;POINT TO NEXT LnYi
08D6  89 1E 2094 R                     MOV      [LnYi_M],BX

08DA  8B 1E 2099 R                     MOV      BX,[Xi_Xm_M]    ;GET ADDRESS OF Xi-Xm -> X4-Xm
08DE  83 C3 10                         ADD      BX,16           ;POINT TO NEXT UNIT
08E1  89 1E 2099 R                     MOV      [Xi_Xm_M],BX

08E5  59                               POP      CX
08E6  E2 03                            LOOP     SG_AGAIN_1
08E8  EB 04 90                         JMP      LA
08EB  E9 07A4 R          SG_AGAIN_1:   JMP      SG_AGAIN

                                       ;-----------------------------------------------------;
                                       ; NEXT PART WAS CHECK THE SIGN OF SIGMA(Xi-Xm)(Yi-Ym)  ;
                                       ;      1. IF>=0 RET. [SING_SIGMA]=0.                   ;
                                       ;      2. IF<0, 1 0000 0000 - SIGMA(Xi-Xm)(Yi-Ym)      ;
                                       ;         [SIGN_SIGMA]=1.                              ;
                                       ; THE FINAL RESULT WAS STILL STORED IN SIG_Xi_Xm_Yi_Ym.;
                                       ;-----------------------------------------------------;
08EE  8D 1E 21A2 R       LA:           LEA      BX,SIG_Xi_Xm_Yi_Ym
08F2  B0 09                            MOV      AL,9
08F4  3A 47 07                         CMP      AL,[BX]+7       ;CHECK 8th DIGIT IF 9 MINUS,
08F7  75 3C                            JNZ      PL              ;IF 0 PLUS, IF PLUS RET.

08F9  8D 3E 2182 R                     LEA      DI,COMPLETE     ;YES RESULT IS MINUS ( - )
08FD  C6 45 08 01                      MOV      BYTE PTR[DI+8],01H ;PROPARE FOR SUBTRUCT
0901  8B DF                            MOV      BX,DI
0903  8D 36 21A2 R                     LEA      SI,SIG_Xi_Xm_Yi_Ym ;CALCULATE THE OBSULUTE VALUE
0907  E8 05F9 R                        CALL     EIGHT_BYTE_SUB  ;-|OBSU|

090A  B9 0008                          MOV      CX,8            ;8 BYTE MOVING
090D  8D 3E 21A2 R                     LEA      DI,SIG_Xi_Xm_Yi_Ym ;RESULT DISTINATION
0911  8D 36 2182 R                     LEA      SI,COMPLETE     ;GET Xi ADDRESS
0915  F3/ A4             REPZ          MOVSB                    ;RESULT NOW IN [SIG_Xi_Xm_Yi_Ym]

0917  B9 0010                          MOV      CX,16
091A  8D 1E 2182 R                     LEA      BX,COMPLETE     ;CLEAR [COMPLEATE] FOR NEXT TIME
091E  C6 07 00           CAC:          MOV      [BX],BYTE PTR 00H ;USE
[illegible]  [illegible]3              INC      BX
[illegible]  [illegible] FA            LOOP     CAC

09[illegible]  [illegible] 1E 2098 R   LEA      BX,SIGN_SIGMA   ;[SIGN_SIGMA]=1
09[illegible]  [illegible] 3E 23CA R   LEA      DI,SYMBLE_B
092C  [illegible] 01                   MOV      AL,1
092E  [illegible] 07                   MOV      [BX],AL
0930  88 05                            MOV      [DI],AL
0932  EB 0F 90                         JMP      PLL

0935  8D 1E 2098 R       PL:           LEA      BX,SIGN_SIGMA   ;[SIGN_SIGMA]=1
0939  8D 3E 23CA R                     LEA      DI,SYMBLE_B
093D  B0 00                            MOV      AL,0
093F  88 07                            MOV      [BX],AL
0941  88 05                            MOV      [DI],AL

0943  C3                 PLL:          RET

                         D_NOMINATION ENDP
                         ;
                         ;
0944                     D_DENOMINATION PROC    NEAR            ;SUBROUTINE No. 20 FOR THE RTDP
                                       ;-------------------------------------------;
                                       ;THIS PROGRAM WAS TO CALCULATE SIGMA (Xi-Xm) ;
                                       ; 1. GET THE DATA FROM (X1_Xm) -- (X4_Xm)    ;
                                       ; 2. (Xi-Xm)^2 CALCULATION RSULT IN PRODUCT. ;
                                       ; 3. THE PRODUCTS ARE ADDED ON SIG_Xi_Xm^2   ;
                                       ;-------------------------------------------;
0944  B9 0003                          MOV      CX,03           ;SIGMA I = 1 -- 4
```

```
0947  8D 1E 21D2 R                    LEA     BX,X1_Xm
094B  89 1E 2099 R                    MOV     [X1_Xm_M],BX     ;X1_XM ADDRESS STORAGE

094F  51                PZ:           PUSH    CX
0950  8B 36 2099 R                    MOV     SI,[X1_Xm_M]
0954  8B FE                           MOV     DI,SI
0956  8D 1E 20D2 R                    LEA     BX,INTERMIDIATE

095A  E8 064B R                       CALL    EIGH_MUL_EIGH    ;THE RESULT WAS AT PRODUCTION.

095D  8D 1E 2212 R                    LEA     BX,SIG_X1_Xm_SQ_2 ;GET THE ADDRESS OF
0961  8D 36 20E2 R                    LEA     SI,PRODUCT       ;SIGMA (X1-Xm)^2
0965  51                              PUSH    CX
0966  E8 05CD R                       CALL    NINE_BYTE_ADD
0969  59                              POP     CX
096A  8B 1E 2099 R                    MOV     BX,[X1_Xm_M]     ;X1_XM ADDRESS STORAGE
096E  83 C3 10                        ADD     BX,16
0971  89 1E 2099 R                    MOV     [X1_Xm_M],BX     ;X1_XM ADDRESS STORAGE
0975  59                              POP     CX
0976  E2 D7                           LOOP    PZ

0978  C3                              RET

                        D_DENOMINATION ENDP
                        ;
                        ;
0979                    D_RESULT      PROC    NEAR             ;SUBROUTINE No. 21 FOR THE RTDP.
                              ;-----------------------------------------------------------;
                              ; THIS SUBROUTINE WAS TO CALCULATE THE PARAMETER D IN BINARY;
                              ;   i.e. D'S NOMINATION / D'S DENOMINATION=D (BIN)          ;
                              ;   IT WAS 32 BIT DIVIDED BY 32 BIT CALCULATION             ;
                              ; 1. THE DIVISOR WAS AT [DI] -- [DI+5]                      ;
                              ;    IF THE DIVISOR WAS EQUAL OR ABOUVE 65535               ;
                              ;    THEN DIVISOR/16, THEN DIVIDEND/16 AGAIN.               ;
                              ; 2. THE DIVIDEND WAS AT [SI] -- [SI+5]                     ;
                              ; 3. THE QUOTINET WAS AT [BX] -- [BX+5] THE REMAINDER IS    ;
                              ;-----------------------------------------------------------;
0979  8B 45 02                        MOV     AX,[DI+2]        ;TEST THE DIVISOR
097C  3D 0001                         CMP     AX,1
097F  73 03                           JAE     DIVISOR_16
0981  EB 1F 90                        JMP     NORMAL

0984  8B 05             DIVISOR_16:   MOV     AX,[DI]          ;THE ACTION EQUALS TO DATA/16
0986  B1 04                           MOV     CL,4
0988  D3 E8                           SHR     AX,CL
098A  50                              PUSH    AX
098B  8B 45 02                        MOV     AX,[DI+2]
098E  B1 0C                           MOV     CL,12
0990  D3 E0                           SHL     AX,CL
0992  8B C8                           MOV     CX,AX
0994  58                              POP     AX
0995  0B C1                           OR      AX,CX
0997  89 05                           MOV     [DI],AX

0999  53                              PUSH    BX
099A  8D 1E 209B R                    LEA     BX,ABOVE_16
099E  C6 07 01                        MOV     BYTE PTR[BX],1
09A1  5B                              POP     BX
                                      ;-------------- INTEGER PART ----------------------
09A2  BA 0000           NORMAL:       MOV     DX,00H           ;CLEAR DX FOR STORING REMAINDER
09A5  8B 44 02                        MOV     AX,[SI+2]        ;FETCH THE MSW OF DIVIDEND
09A8  8B 0D                           MOV     CX,[DI]          ;FETCH THE DIVISOR
09AA  F7 F1                           DIV     CX               ;DX:AX/CX = AX ......DX

09AC  89 47 04                        MOV     [BX+4],AX        ;THE QUOTINET MSW
09AF  8B 04                           MOV     AX,[SI]          ;FETCH THE LSW OF DIVIDEND
09B1  F7 F1                           DIV     CX               ;DX:AX/CX = AX ......DX
```

```
 09B3  89 47 02                    MOV     [BX+2],AX

 09B6  B8 0000                     MOV     AX,00H          ;CLEAR AX FOR DICIMAL FRACTION
 09B9  F7 F1                       DIV     CX
 09BB  89 07                       MOV     [BX],AX         ;DICIMAL PART IN [BX]

 09BD  53                          PUSH    BX
 09BE  8D 1E 209B R                LEA     BX,ABOVE_16     ;CHECK FLAG
 09C2  B0 01                       MOV     AL,1
 09C4  3A 07                       CMP     AL,[BX]
 09C6  5B                          POP     BX
 09C7  75 1C                       JNZ     NORMAL_1        ;IF 1 DIVISOR/16, IF 0 NORMAL_1

 09C9  BA 0000                     MOV     DX,00H          ;DIVISOR/16
 09CC  8B 47 04                    MOV     AX,[BX+4]
 09CF  B9 0010                     MOV     CX,16
 09D2  F7 F1                       DIV     CX
 09D4  89 47 04                    MOV     [BX+4],AX

 09D7  8B 47 02                    MOV     AX,[BX+2]
 09DA  F7 F1                       DIV     CX
 09DC  89 47 02                    MOV     [BX+2],AX

 09DF  8B 07                       MOV     AX,[BX]
 09E1  F7 F1                       DIV     CX
 09E3  89 07                       MOV     [BX],AX

 09E5  53               NORMAL_1:  PUSH    BX
 09E6  8D 1E 209B R                LEA     BX,ABOVE_16     ;RESET THE FLAG FOR NEXT TIME USE
 09EA  C6 07 00                    MOV     BYTE PTR[BX],0
 09ED  5B                          POP     BX

 09EE  C3                          RET

                        D_RESULT   ENDP
                        ;
                        ;
 09EF                   C_RESULT   PROC    NEAR            ;SUBROUTINE No. 22 FOR THE RTDP.
                                   ;-------------------------------------------------------;
                                   ; THIS SUBROUTINE WAS TO CALCULATE THE PARAMETER C (BCD).;
                                   ;      C = MEAN_LnYi + D * MEAN_Xi                       ;
                                   ;  1. D * MEAN_Xi                                        ;
                                   ;  2. MEAN_LnYi + D * MEAN_Xi                            ;
                                   ;-------------------------------------------------------;
                                   ;-------- CALCULATE D * MEAN_Xi ----------
 09EF  8D 36 2242 R                LEA     SI,D_BCD        ;POINT TO D_BCD
 09F3  8D 3E 2172 R                LEA     DI,MEAN_Xi
 09F7  8D 1E 20D2 R                LEA     BX,INTERMIDIATE
 09FB  E8 064B R                   CALL    EIGH_MUL_EIGH   ;NOW THE RESULT WAS IN [PRODUTOIN]

 09FE  FC                          CLD                     ;SET DF=0 TO MOVE FORWARD
 09FF  8D 36 2112 R                LEA     SI,MEAN_LnYi    ;MOV THIS DATA TO NEXT DATA AREA
 0A03  8D 3E 22C2 R                LEA     DI,MEAN_LnYi_FOR_C ;TO KEEP THE MEAN_LnYi VALUE
 0A07  B9 0010                     MOV     CX,16
 0A0A  F3/ A4           REP        MOVSB                   ;MOVES 16 BYTE FROM [SI] TO [DI]

 0A0C  8D 36 20E2 R                LEA     SI,PRODUCT
 0A10  8D 1E 22C2 R                LEA     BX,MEAN_LnYi_FOR_C
 0A14  51                          PUSH    CX
 0A15  E8 05E3 R                   CALL    SIXTEEN_BYTE_ADD ;NOW THE RESULT OF C WAS
 0A18  59                          POP     CX              ;IN [MEAN_LnYi_FOR_C].

 0A19  C3                          RET

                        C_RESULT   ENDP
                        ;
                        ;
```

```
 0A1A                             BIN_BCD    PROC    NEAR         ;SUBROUTINE No. 23 FOR THE R1DP.
                                   ;-------------------------------------------------------------;
                                   ;THIS PROGRAM WAS TO CONVERT A BINERY DATA INTO 8 DIGIT       ;
                                   ;     BCD DATA                                                ;
                                   ; 1. THE DATA TO BE CONVERTED WAS AT [SI] -- [SI+3] 2 WORDS   ;
                                   ;          ------------------                                 ;
                                   ;       i.e.   WORD . WORD                                    ;
                                   ;          ------------------                                 ;
                                   ; 2. [SI+3] -- [SI+2] ARE THE INTEGER PART ( WORD )           ;
                                   ;    [SI+1] -- [SI]  ARE DISIMAL FRACTION PART ( WORD )       ;
                                   ; 3. DICIMAL PART BIN-BCD STORE IN [DI] -- [DI+4] TEMPERARELY ;
                                   ; 4. THE RESULT WIL BE STORED AT [BX] -- [BX+8]               ;
                                   ;-------------------------------------------------------------;

                                      ;---------- INTERGR PART -----------
 0A1A  BA 0000                        MOV        DX,0
 0A1D  8B 44 02                       MOV        AX,[SI+2]
 0A20  B9 2710                        MOV        CX,10000
 0A23  F7 F1                          DIV        CX
 0A25  88 47 08                       MOV        [BX+8],AL

 0A28  8B C2                          MOV        AX,DX
 0A2A  BA 0000                        MOV        DX,00H
 0A2D  B9 03E8                        MOV        CX,1000
 0A30  F7 F1                          DIV        CX
 0A32  88 47 07                       MOV        [BX+7],AL

 0A35  8B C2                          MOV        AX,DX
 0A37  BA 0000                        MOV        DX,00H
 0A3A  B9 0064                        MOV        CX,100
 0A3D  F7 F1                          DIV        CX
 0A3F  88 47 06                       MOV        [BX+6],AL

 0A42  8B C2                          MOV        AX,DX
 0A44  BA 0000                        MOV        DX,00H
 0A47  B9 000A                        MOV        CX,10
 0A4A  F7 F1                          DIV        CX
 0A4C  88 47 05                       MOV        [BX+5],AL

 0A4F  88 57 04                       MOV        [BX+4],DL

                                    ;------------- DISIMAL FRACTION PART -------------;
                                    ;                 1. CONVERT BIN TO BCD           ;
                                    ;                 2. BCD * 0.000015258            ;
                                    ;-------------------------------------------------;
 0A52  BA 0000                        MOV        DX,00H
 0A55  8B 04                          MOV        AX,[SI]
 0A57  B9 2710                        MOV        CX,10000
 0A5A  F7 F1                          DIV        CX
 0A5C  88 45 04                       MOV        [DI+4],AL

 0A5F  8B C2                          MOV        AX,DX
 0A61  BA 0000                        MOV        DX,00H
 0A64  B9 03E8                        MOV        CX,1000
 0A67  F7 F1                          DIV        CX
 0A69  88 45 03                       MOV        [DI+3],AL

 0A6C  8B C2                          MOV        AX,DX
 0A6E  BA 0000                        MOV        DX,00H
 0A71  B9 0064                        MOV        CX,100
 0A74  F7 F1                          DIV        CX
 0A76  88 45 02                       MOV        [DI+2],AL

 0A79  8B C2                          MOV        AX,DX
 0A7B  BA 0000                        MOV        DX,00H
 0A7E  B9 000A                        MOV        CX,10
 0A81  F7 F1                          DIV        CX
```

```
 0A83  88 45 01                   MOV     [DI+1],AL

 0A86  88 15                      MOV     [DI],DL

                                  ;-------NEXT PART WAS DISIMAL FRACTION * 0.000015258-----;
 0A88  8D 1E 2262 R               LEA     BX,D_15258      ;SET 0.000015258
 0A8C  B0 08                      MOV     AL,8
 0A8E  88 07                      MOV     BYTE PTR[BX],AL
 0A90  43                         INC     BX
 0A91  B0 05                      MOV     AL,5
 0A93  88 07                      MOV     BYTE PTR[BX],AL
 0A95  43                         INC     BX
 0A96  B0 02                      MOV     AL,2
 0A98  88 07                      MOV     BYTE PTR[BX],AL
 0A9A  43                         INC     BX
 0A9B  B0 05                      MOV     AL,5
 0A9D  88 07                      MOV     BYTE PTR[BX],AL
 0A9F  43                         INC     BX
 0AA0  B0 01                      MOV     AL,1
 0AA2  88 07                      MOV     BYTE PTR[BX],AL
 0AA4  43                         INC     BX

 0AA5  8D 36 2262 R               LEA     SI,D_15258
 0AA9  8D 3E 2252 R               LEA     DI,D_BCD_FRACTION
 0AAD  8D 1E 20E2 R               LEA     BX,PRODUCT
 0AB1  E8 064B R                  CALL    EIGH_MUL_EIGH

 0AB4  B9 0004                    MOV     CX,4            ;4 BYTE MOVING
 0AB7  8D 3E 2242 R               LEA     DI,D_BCD        ;RESULT DISTINATION
 0ABB  8D 36 20E3 R               LEA     SI,PRODUCT+1    ;GET THE REAL PRODUCT ADDRESS
 0ABF  F3/ A4             REPZ    MOVSB                   ;THE RESULT NOW IN .XXXX

 0AC1  C3                         RET

                          BIN_BCD ENDP
                          ;
                          ;
 0AC2                     BCD_BIN PROC    NEAR            ;SUBROUTINE No. 24 FOR THE RTDP.
                                  ;-----------------------------------------------------;
                                  ;THIS SUBROUTINE WAS TO CONVERT ONE 6 DIGIT BCD DATA;
                                  ;IN TO BINERY DATA.                                  ;
                                  ; 1.THE DATA TO BE CONVERTED WAS AT [SI] -- [SI]+8   ;
                                  ; 2.THE RESULT WILL BE PUT IN [BX] WORD              ;
                                  ;-----------------------------------------------------;
 0AC2  BA 0000                    MOV     DX,00H
 0AC5  B8 0000                    MOV     AX,00H
 0AC8  8A 04                      MOV     AL,BYTE PTR[SI] ;LSB OF THE DATA TO BE CONVERTED
 0ACA  88 07                      MOV     [BX],AL

 0ACC  46                         INC     SI
 0ACD  8A 04                      MOV     AL,BYTE PTR[SI] ;SECOND LSB OF THE DATA TO BE
 0ACF  B9 000A                    MOV     CX,10           ;CONVERTED
 0AD2  F7 E1                      MUL     CX              ;THE DIGIT WAS MULTIPLIED BY 10
 0AD4  03 07                      ADD     AX,[BX]         ;CONVERTED DATA ADDS LSB
 0AD6  89 07                      MOV     [BX],AX         ;SEND TO RAM
 0AD8  13 57 02                   ADC     DX,[BX+2]       ;IF THERE WAS CARRY ADD
 0ADB  89 57 02                   MOV     [BX+2],DX       ;STORE HIGH WORD OF RESULT
 0ADE  B4 00                      MOV     AH,00H          ;CLEAR AH FOR NEXT TIME USE

 0AE0  46                         INC     SI
 0AE1  8A 04                      MOV     AL,BYTE PTR[SI] ;THIRD LSB OF THE DATA TO BE
 0AE3  B9 0064                    MOV     CX,100          ;CONVERTED
 0AE6  F7 E1                      MUL     CX              ;THE DIGIT WAS MULTIPLIED BY 100
 0AE8  03 07                      ADD     AX,[BX]         ;CONVERTED DATA ADDS LSB
 0AEA  89 07                      MOV     [BX],AX         ;SEND TO RAM
 0AEC  13 57 02                   ADC     DX,[BX+2]       ;IF THERE WAS CARRY ADD
 0AEF  89 57 02                   MOV     [BX+2],DX       ;STORE HIGH WORD OF RESULT
```

```
 0AF2  B4 00                          MOV     AH,00H             ;CLEAR AH FOR NEXT TIME USE

 0AF4  46                             INC     SI
 0AF5  8A 04                          MOV     AL,BYTE PTR[SI]  ;FOURTH LSB OF THE DATA TO BE
 0AF7  B9 03E8                        MOV     CX,1000          ;CONVERTED
 0AFA  F7 E1                          MUL     CX               ;THE DIGIT WAS MULTIPLIED BY 1000
 0AFC  03 07                          ADD     AX,[BX]          ;CONVERTED DATA ADDS LSB
 0AFE  89 07                          MOV     [BX],AX          ;SEND TO RAM
 0B00  13 57 02                       ADC     DX,[BX+2]        ;IF THERE WAS CARRY ADD
 0B03  89 57 02                       MOV     [BX+2],DX        ;STORE HIGH WORD OF RESULT
 0B06  B4 00                          MOV     AH,00H           ;CLEAR AH FOR NEXT TIME USE

 0B08  46                             INC     SI
 0B09  8A 04                          MOV     AL,BYTE PTR[SI]  ;FIFTH LSB OF THE DATA TO BE
 0B0B  B9 2710                        MOV     CX,10000         ;CONVERTED
 0B0E  F7 E1                          MUL     CX               ;THE DIGIT WAS MULTIPLIED BY 10000
 0B10  03 07                          ADD     AX,[BX]          ;CONVERTED DATA ADDS LSB
 0B12  89 07                          MOV     [BX],AX          ;SEND TO RAM
 0B14  13 57 02                       ADC     DX,[BX+2]        ;IF THERE WAS CARRY ADD
 0B17  89 57 02                       MOV     [BX+2],DX        ;STORE HIGH WORD OF RESULT
 0B1A  B4 00                          MOV     AH,00H           ;CLEAR AH FOR NEXT TIME USE

                                      ;--- NEST PART WAS DIGIT*100,000=DIGIT*(5000+50000)---;
 0B1C  B9 0002                        MOV     CX,2
 0B1F  46                             INC     SI               ;SIXTH DIGIT OF DATA
 0B20  51                   NN:       PUSH    CX
 0B21  8A 04                          MOV     AL,BYTE PTR[SI]
 0B23  B9 C350                        MOV     CX,50000
 0B26  F7 E1                          MUL     CX
 0B28  03 07                          ADD     AX,[BX]
 0B2A  89 07                          MOV     [BX],AX
 0B2C  13 57 02                       ADC     DX,[BX+2]
 0B2F  89 57 02                       MOV     [BX+2],DX
 0B32  B4 00                          MOV     AH,00H
 0B34  59                             POP     CX
 0B35  E2 E9                          LOOP    NN

 0B37  C3                             RET

                            BCD_BIN   ENDP
                            ;
                            ;
 0B38                       BIN_BCD_INTEGER     PROC    NEAR  ;SUBROUTINE No. 25 FOR THE RTDP.
                                      ;-------------------------------------------------------;
                                      ; 1. BCD IN [SI] WAS A WORD TO BE CONVERDED             ;
                                      ; 2. RESULT IN [BX] -- [BX+4] ARE 5 DIGITS ASCII CODE   ;
                                      ;-------------------------------------------------------;
 0B38  BA 0000                        MOV     DX,0
 0B3B  8B 04                          MOV     AX,[SI]
 0B3D  B9 2710                        MOV     CX,10000
 0B40  F7 F1                          DIV     CX
 0B42  0C 30                          OR      AL,30H
 0B44  88 07                          MOV     [BX],AL

 0B46  8B C2                          MOV     AX,DX
 0B48  BA 0000                        MOV     DX,00H
 0B4B  B9 03E8                        MOV     CX,1000
 0B4E  F7 F1                          DIV     CX
 0B50  0C 30                          OR      AL,30H
 0B52  88 47 01                       MOV     [BX+1],AL

 0B55  8B C2                          MOV     AX,DX
 0B57  BA 0000                        MOV     DX,00H
 0B5A  B9 0064                        MOV     CX,100
 0B5D  F7 F1                          DIV     CX
 0B5F  0C 30                          OR      AL,30H
 0B61  88 47 02                       MOV     [BX+2],AL
```

```
0B64  8B C2                          MOV        AX,DX
0B66  BA 0000                        MOV        DX,00H
0B69  B9 000A                        MOV        CX,10
0B6C  F7 F1                          DIV        CX
0B6E  0C 30                          OR         AL,30H
0B70  88 47 03                       MOV        [BX+3],AL
0B73  80 CA 30                       OR         DL,30H
0B76  88 57 04                       MOV        [BX+4],DL

0B79  C3                             RET

                   BIN_BCD_INTEGER              ENDP
                   ;
                   ;
0B7A               BCD_BIN_C    PROC      NEAR            ;SUBROUTINE No. 26 FOR THE RTDP.
                                ;-----------------------------------------------------;
                                ;THIS SUBROUTINE WAS TO CONVERT ONE 6 DIGIT BCD DATA ;
                                ;IN TO BINERY DATA.  X X . X X X X                   ;
                                ; 1.THE DATA TO BE CONVERTED WAS AT [SI] -- [BX]+8   ;
                                ; 2.THE RESULT WILL BE PUT IN [BX+2].[BX] WORD       ;
                                ;-----------------------------------------------------;
0B7A  8A 04                          MOV        AL,BYTE PTR[SI] ;LSB OF FRAGMENT PART OF
0B7C  B4 00                          MOV        AH,0
0B7E  B9 0006                        MOV        CX,6H           ;THE DATA TO BE CONVERTED
0B81  F7 E1                          MUL        CX              ;THE DIGIT WAS MULTIPLIED
0B83  89 07                          MOV        [BX],AX         ;BY 0.0007H, RESULT WAS IN [BX].

0B85  46                             INC        SI
0B86  8A 04                          MOV        AL,BYTE PTR[SI] ;SECOND LSB OF THE DATA
0B88  B4 00                          MOV        AH,0
0B8A  B9 0042                        MOV        CX,42H
0B8D  F7 E1                          MUL        CX             ;THE DIGIT WAS MULTIPLIED BY 0.0042H
0B8F  03 07                          ADD        AX,[BX]        ;CONVERTED DATA ADDS LSB
0B91  89 07                          MOV        [BX],AX        ;SEND TO RAM

0B93  46                             INC        SI
0B94  8A 04                          MOV        AL,BYTE PTR[SI] ;THIRD LSB OF THE DATA TO BE
0B96  B4 00                          MOV        AH,0            ;CONVERTED
0B98  B9 028F                        MOV        CX,28FH
0B9B  F7 E1                          MUL        CX             ;THE DIGIT WAS MULTIPLIED BY 0.028FH
0B9D  03 07                          ADD        AX,[BX]        ;CONVERTED DATA ADDS LSB
0B9F  89 07                          MOV        [BX],AX        ;SEND TO RAM

0BA1  46                             INC        SI
0BA2  8A 04                          MOV        AL,BYTE PTR[SI] ;FOURTH LSB OF THE DATA TO BE
0BA4  B4 00                          MOV        AH,0            ;CONVERTED
0BA6  B9 199A                        MOV        CX,199AH
0BA9  F7 E1                          MUL        CX             ;THE DIGIT WAS MULTIPLIED BY 0.199AH
0BAB  03 07                          ADD        AX,[BX]        ;CONVERTED DATA ADDS LSB
0BAD  89 07                          MOV        [BX],AX        ;SEND TO RAM

0BAF  46                             INC        SI
0BB0  8A 04                          MOV        AL,BYTE PTR[SI] ;LSB OF THE INTEGER PART
0BB2  B4 00                          MOV        AH,0
0BB4  89 47 02                       MOV        [BX+2],AX       ;SEND TO RAM

0BB7  46                             INC        SI              ;SIXTH DIGIT OF DATA
0BB8  8A 04                          MOV        AL,BYTE PTR[SI]
0BBA  B4 00                          MOV        AH,0
0BBC  B9 000A                        MOV        CX,10
0BBF  F7 E1                          MUL        CX
0BC1  03 47 02                       ADD        AX,[BX+2]
0BC4  89 47 02                       MOV        [BX+2],AX

0BC7  C3                             RET
```

```
                                        BCD_BIN_C     ENDP
                                        ;
                                        ;
0BC8                                    A_COEF_A_D_B PROC     NEAR           ;SUBROUTINE No. 27 FOR THE RTDP.
                                                  ;-------------------------------------------------;
                                                  ; THIS PROCEDURE WAS TO CALCULATE A_COEF AND A/B   ;
                                                  ; BECAUSE C = LnA, COEF. A = EXP(C) = EXP(a+b)     ;
                                                  ; EXP(a+b) = EXPa * EXPb. EXPa: LOOK UP TABLE;     ;
                                                  ; 1. EXPb WAS CALCULATED WITH INTERACTIVE OPRATION.;
                                                  ; 2. THE RESULT WILL BE IN [EXP_N+2].[EXP_N]       ;
                                                  ; 3. A WILL BE IN [A_COEF].[A_COEF+2]              ;
                                                  ; 4. A/B WILL BE IN [A_DIV_B+2].[A_DIV_B]          ;
                                                  ;-------------------------------------------------;
0BC8  8D 36 22A2 R                                LEA      SI,C_BIN
0BCC  8D 1E 2272 R                                LEA      BX,EXP_b
0BD0  E8 0CAD R                                   CALL     EXP_a_PLUS_b    ;CALCULATE EXP a+b WITH
                                                                           ;INTERACTIVE OPRATION
0BD3  8D 3E 23BC R                                LEA      DI,A_COEF       ;POINT TO A COEFFCIENT
0BD7  8D 36 209C R                                LEA      SI,EXP_N        ;MOVE A_COEF FOR FOUTHER PROCESS
0BDB  8B 04                                       MOV      AX,[SI]
0BDD  89 05                                       MOV      [DI],AX
0BDF  8B 44 02                                    MOV      AX,[SI+2]
0BE2  89 45 02                                    MOV      [DI+2],AX       ;A_COEF WAS IN [A_COEF+2].[A_COEF]

0BE5  B8 0000                                     MOV      AX,0            ;CLEAR [EXP_N] FOR NEXT CALCULAT
0BE8  89 04                                       MOV      [SI],AX
0BEA  89 44 02                                    MOV      [SI+2],AX

0BED  8D 36 23BC R                                LEA      SI,A_COEF       ;POINT TO A COEF
0BF1  8D 3E 23C4 R                                LEA      DI,D_BIN        ;POINT TO B COEF
0BF5  8D 1E 2300 R                                LEA      BX,A_DIV_B      ;POINT TO RESULT ADDRESS A/B

0BF9  53                                          PUSH     BX
0BFA  8B 05                                       MOV      AX,[DI]         ;PROTECT DATA IN [DI+2].[DI]
0BFC  8D 1E 20B2 R                                LEA      BX,TABLE_M      ;BORROW THOS AREA
0C00  89 07                                       MOV      [BX],AX
0C02  8B 45 02                                    MOV      AX,[DI+2]
0C05  89 47 02                                    MOV      [BX+2],AX
0C08  5B                                          POP      BX

0C09  E8 0979 R                                   CALL     D_RESULT        ;CALCULATE A/B. THE RESULT NOW IN
                                                                           ;[A_DIV_B+2].[A_DIV_B]
0C0C  8D 1E 20B2 R                                LEA      BX,TABLE_M      ;RESUME THE ORIGANAL DATA
0C10  8D 3E 23C4 R                                LEA      DI,D_BIN
0C14  8B 07                                       MOV      AX,[BX]
0C16  89 05                                       MOV      [DI],AX
0C18  8B 47 02                                    MOV      AX,[BX+2]
0C1B  89 45 02                                    MOV      [DI+2],AX

0C1E  C3                                          RET

                                        A_COEF_A_D_B ENDP
                                        ;
                                        ;
0C1F                                    NEGA_B_MUL_X0       PROC      NEAR    ;SUBROUTINE No. 28 FOR THE RTDP.
                                                  ;-------------------------------------
                                                  ;THIS SUBROUTINE WAS TO CALCULATE -B*X0
                                                  ;THE RESULT WILLBE IN NEGA_B_X0
                                                  ;-------------------------------------
0C1F  8D 36 2152 R                                LEA      SI,X0           ;CALCULATE -B*X0
0C23  8D 1E 23C4 R                                LEA      BX,D_BIN        ;D=B
0C27  8D 3E 2304 R                                LEA      DI,NEGA_B_X0    ;THE RESULT WILL BE IN
                                                                           ;[NEGA_B_X0+2].[NEGA_B_X0]
0C2B  8B 04                                       MOV      AX,[SI]         ;X0 FRAGMENT PART ONLY
0C2D  8B 0F                                       MOV      CX,[BX]         ;B'S FRAGMENT PART IN CX
0C2F  F7 E1                                       MUL      CX              ;DX:AX FRAGMENT:FRAGMENT AX
0C31  89 15                                       MOV      [DI],DX         ;OMITTED
```

```
 0C33  8B 04                          MOV       AX,[SI]
 0C35  8B 4F 02                       MOV       CX,[BX+2]        ;B'S INTEGER PART IN CX
 0C38  F7 E1                          MUL       CX               ;DX:AX INTEGER:FRAGMENT
 0C3A  03 05                          ADD       AX,[DI]
 0C3C  89 05                          MOV       [DI],AX
 0C3E  13 55 02                       ADC       DX,[DI+2]
 0C41  89 55 02                       MOV       [DI+2],DX

 0C44  C3                             RET

                    NEGA_B_MUL_X0  ENDP
                    ;
                    ;
 0C45               EXP_NEGA_B_X0   PROC    NEAR             ;SUBROUTINE No. 29 FOR THE RTDP.

 0C45  8D 1E 2098 R                   LEA       BX,SIGN_SIGMA    ,TEST + OR - SYMBLE OF B=D
 0C49  B0 01                          MOV       AL,1             ;SINCE D DENOMINATION WAS +
 0C4B  3A 07                          CMP       AL,BYTE PTR[BX]  ;D NOMINATION WAS - OR +
 0C4D  74 03                          JZ        FU
 0C4F  EB 0F 90                       JMP       ZHENG

 0C52  8D 36 2304 R   FU:             LEA       SI,NEGA_B_X0
 0C56  8D 1E 2272 R                   LEA       BX,EXP_b
 0C5A  E8 0E7A R                      CALL      EXP_a_PLUS_b_NEGA
 0C5D  EB 0C 90                       JMP       FU_1

 0C60  8D 36 2304 R   ZHENG:          LEA       SI,NEGA_B_X0
 0C64  8D 1E 2272 R                   LEA       BX,EXP_b
 0C68  E8 0CAD R                      CALL      EXP_a_PLUS_b

 0C6B  8D 3E 2308 R   FU_1:           LEA       DI,EXP_N_B_X0
 0C6F  8D 36 209C R                   LEA       SI,EXP_N
 0C73  8B 04                          MOV       AX,[SI]
 0C75  89 05                          MOV       [DI],AX
 0C77  8B 44 02                       MOV       AX,[SI+2]
 0C7A  89 45 02                       MOV       [DI+2],AX

 0C7D  C3                             RET

                    EXP_NEGA_B_X0   ENDP
                    ;
                    ;
 0C7E               LOSS            PROC      NEAR           ;SUBROUTINE No. 30 FOR THE RTDP.
                                    ;----------------------------------------------------------;
                                    ;THIS SUBROUTINE WAS TO CALCULATE A/B*EXP(-B*X0)            ;
                                    ; 1. A_DIV_B * EXP_N_B_X0 = LOSS_DATA_1                     ;
                                    ; 2. THE RESULT WILL BE IN [LOSS_DATA_1+2].[LOSS_DATA_1]    ;
                                    ;----------------------------------------------------------;
 0C7E  8D 36 2300 R                   LEA       SI,A_DIV_B       ;POINT TO A/B
 0C82  8D 1E 2308 R                   LEA       BX,EXP_N_B_X0    ;POINT TO EXP(-B*X0)
 0C86  8D 3E 23CC R                   LEA       DI,LOSS_DATA_1   ;POINT TO RESULT LOSS_DATA_1
 0C8A  8B 04                          MOV       AX,[SI]          ;FRAGMENT PART OF A/B
 0C8C  F7 27                          MUL       WORD PTR[BX]     ;MUL FRAGMENT PART OF EXP(-B*X0)
 0C8E  89 15                          MOV       [DI],DX          ;AX WAS OMITTED
 0C90  8B 44 02                       MOV       AX,[SI+2]        ;INTEGER PART OF A/B
 0C93  F7 27                          MUL       WORD PTR[BX]     ;MUL FRAGMENT OF EXP(-B*X0)
 0C95  03 05                          ADD       AX,[DI]          ;DX:AX INTEGER:FRAGMENT
 0C97  89 05                          MOV       [DI],AX
 0C99  13 55 02                       ADC       DX,[DI+2]
 0C9C  89 55 02                       MOV       [DI+2],DX

 0C9F  8B 44 02                       MOV       AX,[SI+2]        ;INTEGER PART OF A/B
 0CA2  F7 67 02                       MUL       WORD PTR[BX+2]   ;MUL INTEGER PART OF EXP(-B*X0)
 0CA5  03 45 02                       ADD       AX,[DI+2]
 0CA8  89 45 02                       MOV       [DI+2],AX

 0CAB  C3                             RET
```

```
                        LOSS         ENDP
                        ;
                        ;
0CAC                    GRAIN_LOSS   PROC     NEAR

0CAC  C3                             RET

                        GRAIN_LOSS   ENDP
                        ;
                        ;
0CAD                    EXP_a_PLUS_b PROC     NEAR          ;SUBROUTINE NO. 31 FOR THE RTDP.
                                     ;-----------------------------------------------------;
                                     ; THIS SUBROUTINE WAS TO CALCULATE EXP (a+b). a AND b ARE;
                                     ; INTEGER AND DISIMAL PART RESPECTIVELLY.              ;
                                     ; 1.EXP a WILL BE FOUND IN LOOK UP TABLE               ;
                                     ; 2.EXP b WILL BE CALCULATED WITH INTERACTIVE OPRATION ;
                                     ; 3.BEFOR CALCULATION THE DATA TO BE CONVERTED WAS AT  ;
                                     ;   [SI+2].[SI]                                        ;
                                     ; 4.THE RESULT THEN WAS PUT IN [BX] i.e. [EXP_N] FRAC  ;
                                     ;   [EXP_N+2] INTEGER                                  ;
                                     ;-----------------------------------------------------;
                                     ;----------THE FIRST PART WAS EXP(b) -----------
0CAD  8D 1E 209E R                   LEA      BX,EXP_N+2     ;RESULT INTEGER ADDRESS
0CB1  B8 0001                        MOV      AX,01          ;THE FIRST ITERM
0CB4  89 07                          MOV      [BX],AX        ;INTEGER PART
0CB6  8D 1E 209C R                   LEA      BX,EXP_N       ;FRACTION PART
0CBA  8B 04                          MOV      AX,[SI]        ;1+D_BIN+...
0CBC  89 07                          MOV      [BX],AX        ;RESULT IN [EXP_N] LOOKS LIKE 1.b
0CBE  8B 04                          MOV      AX,[SI]        ;FETCH b (BIN)
0CC0  BA 0000                        MOV      DX,0           ;CLEAR DX FOR DIV REMAINDER
0CC3  B9 0001                        MOV      CX,1           ; ^2
0CC6  E8 0F92 R                      CALL     DATA_POWER     ; DATA POWER
0CC9  B9 0002                        MOV      CX,CM_2        ; 2!
0CCC  F7 F1                          DIV      CX             ; DATA POWER / 2!
0CCE  03 07                          ADD      AX,[BX]        ; +D_BIN^2/2!
0CD0  89 07                          MOV      [BX],AX        ;THE RESULT NOW IN [EXP_N]
0CD2  8B 47 02                       MOV      AX,[BX+2]
0CD5  15 0000                        ADC      AX,0           ;ADD THE CARRY
0CD8  89 47 02                       MOV      [BX+2],AX      ;NEW INTEGER PART

0CDB  8B 04                          MOV      AX,[SI]        ;FETCH b (BIN)
0CDD  BA 0000                        MOV      DX,0           ;CLEAR DX FOR DIV REMAINDER
0CE0  B9 0002                        MOV      CX,2           ; ^3
0CE3  E8 0F92 R                      CALL     DATA_POWER     ; DATA POWER
0CE6  B9 0006                        MOV      CX,CM_3        ; 3!
0CE9  F7 F1                          DIV      CX             ; DATA POWER / 3!
0CEB  8D 1E 209C R                   LEA      BX,EXP_N
0CEF  03 07                          ADD      AX,[BX]        ; +D_BIN^2/3!
0CF1  89 07                          MOV      [BX],AX        ;THE RESULT NOW IN [EXP_N]
0CF3  8B 47 02                       MOV      AX,[BX+2]
0CF6  15 0000                        ADC      AX,0           ;ADD THE CARRY
0CF9  89 47 02                       MOV      [BX+2],AX      ;NEW INTEGER PART

0CFC  8B 04                          MOV      AX,[SI]        ;FETCH b (BIN)
0CFE  BA 0000                        MOV      DX,0           ;CLEAR DX FOR DIV REMAINDER
0D01  B9 0003                        MOV      CX,3           ; ^4
0D04  E8 0F92 R                      CALL     DATA_POWER     ; DATA POWER
0D07  B9 0018                        MOV      CX,CM_4        ; 4!
0D0A  F7 F1                          DIV      CX             ; DATA POWER / 4!
0D0C  8D 1E 209C R                   LEA      BX,EXP_N
0D10  03 07                          ADD      AX,[BX]        ; +D_BIN^2/4!
0D12  89 07                          MOV      [BX],AX        ;THE RESULT NOW IN [EXP_N]
0D14  8B 47 02                       MOV      AX,[BX+2]
0D17  15 0000                        ADC      AX,0           ;ADD THE CARRY
0D1A  89 47 02                       MOV      [BX+2],AX      ;NEW INTEGER PART
```

```
0D1D  8B 04                   MOV    AX,[SI]          ;FETCH b (BIN)
0D1F  BA 0000                 MOV    DX,0             ;CLEAR DX FOR DIV REMAINDER
0D22  B9 0004                 MOV    CX,4             ; ^5
0D25  E8 0F92 R               CALL   DATA_POWER       ; DATA POWER
0D28  B9 0078                 MOV    CX,CM_5          ; 5!
0D2B  F7 F1                   DIV    CX               ; DATA POWER / 5!
0D2D  8D 1E 209C R            LEA    BX,EXP_N
0D31  03 07                   ADD    AX,[BX]          ; +D_BIN^2/5!
0D33  89 07                   MOV    [BX],AX          ;THE RESULT NOW IN [EXP_N]
0D35  8B 47 02                MOV    AX,[BX+2]
0D38  15 0000                 ADC    AX,0             ;ADD THE CARRY
0D3B  89 47 02                MOV    [BX+2],AX        ;NEW INTEGER PART

0D3E  8B 04                   MOV    AX,[SI]          ;FETCH b (BIN)
0D40  BA 0000                 MOV    DX,0             ;CLEAR DX FOR DIV REMAINDER
0D43  B9 0005                 MOV    CX,5             ; ^6
0D46  E8 0F92 R               CALL   DATA_POWER       ; DATA POWER
0D49  B9 02D0                 MOV    CX,CM_6          ; 6!
0D4C  F7 F1                   DIV    CX               ; DATA POWER / 6!
0D4E  8D 1E 209C R            LEA    BX,EXP_N
0D52  03 07                   ADD    AX,[BX]          ; +D_BIN^2/6!
0D54  89 07                   MOV    [BX],AX          ;THE RESULT NOW IN [EXP_N]
0D56  8B 47 02                MOV    AX,[BX+2]
0D59  15 0000                 ADC    AX,0             ;ADD THE CARRY
0D5C  89 47 02                MOV    [BX+2],AX        ;NEW INTEGER PART

0D5F  8B 04                   MOV    AX,[SI]          ;FETCH b (BIN)
0D61  BA 0000                 MOV    DX,0             ;CLEAR DX FOR DIV REMAINDER
0D64  B9 0006                 MOV    CX,6             ; ^7
0D67  E8 0F92 R               CALL   DATA_POWER       ; DATA POWER
0D6A  B9 13B0                 MOV    CX,CM_7          ; 7!
0D6D  F7 F1                   DIV    CX               ; DATA POWER / 7!
0D6F  8D 1E 209C R            LEA    BX,EXP_N
0D73  03 07                   ADD    AX,[BX]          ; +D_BIN^2/7!
0D75  89 07                   MOV    [BX],AX          ;THE RESULT NOW IN [EXP_N]
0D77  8D 1E 209E R            LEA    BX,EXP_N+2       ;INTEGER PART ADDRESS
0D7B  8B 47 02                MOV    AX,[BX+2]
0D7E  15 0000                 ADC    AX,0             ;ADD THE CARRY
0D81  89 47 02                MOV    [BX+2],AX        ;NEW INTEGER PART

0D84  8B 04                   MOV    AX,[SI]          ;FETCH b (BIN)
0D86  BA 0000                 MOV    DX,0             ;CLEAR DX FOR DIV REMAINDER
0D89  B9 0007                 MOV    CX,7             ; ^8
0D8C  E8 0F92 R               CALL   DATA_POWER       ; DATA POWER
0D8F  B9 9D80                 MOV    CX,CM_8          ; 8!
0D92  F7 F1                   DIV    CX               ; DATA POWER / 8!
0D94  8D 1E 209C R            LEA    BX,EXP_N
0D98  03 07                   ADD    AX,[BX]          ; +D_BIN^2/8!
0D9A  89 07                   MOV    [BX],AX          ;THE RESULT NOW IN [EXP_N]
0D9C  8B 47 02                MOV    AX,[BX+2]
0D9F  15 0000                 ADC    AX,0             ;ADD THE CARRY
0DA2  89 47 02                MOV    [BX+2],AX        ;NEW INTEGER PART

                 ;--- SO FAR THE RESULT OF EXP b WAS STORED IN [EXP_N+2].[EXP_N] ---;
                 ;    NEXT STEP WAS TO CALCULATE (EXP a) * (EXP b)                  ;
                 ; 1. EXP a, a = 1--15, ARE DEFINEDED DATA AREA (2 X DD OR 8 BYTE). ;
                 ; 2. EXP a * EXP b (INTEGER) + EXP a * EXP b(FRAGMENT)             ;
                 ; 3. THE RESULT WAS STILL STORED IN [EXP_N+4 ] [EXP_N+2].[EXP_N]   ;
                 ;------------------------------------------------------------------;
0DA5  8B 5C 02                MOV    BX,[SI+2]        ;BX WAS BASE ADDRESS FOR LOOK UP
                                                      ;EXP_N
0DA8  BE CE30                 MOV    SI,0CE30H        ;LEA SI,EXP_0+100H THE FIRST
                                                      ;ADDRESS OF LOOK UP TABLE
0DAB  8B C3                   MOV    AX,BX            ;PROPARE BASE * No. OF BYTE
0DAD  B9 0008                 MOV    CX,8
0DB0  F7 E1                   MUL    CX               ;BASE * No. OF BYTE
0DB2  8B D8                   MOV    BX,AX
```

```
0DB4  8B 00                  MOV    AX,[BX][SI]        ;POINT TO THE TABLE

                             ;-----MULTIPLY THE INTEGER PART OF EXP b-------
0DB6  8D 3E 209E R           LEA    DI,EXP_N+2         ;INTEGER ADDRESS OF EXPb
0DBA  BA 0000                MOV    DX,0
0DBD  8B 0D                  MOV    CX,[DI]
0DBF  F7 E1                  MUL    CX                 ;NOW THE RESULT WAS IN DX:AX
0DC1  53                     PUSH   BX                 ;SAVE BASE ADDRESS
0DC2  8D 1E 2282 R           LEA    BX,EXPa_MUL_EXPb_INTEGER
0DC6  89 07                  MOV    [BX],AX
0DC8  89 57 02               MOV    [BX+2],DX
0DCB  5B                     POP    BX                 ;RESUME BASE ADDRESS

0DCC  8B 40 02               MOV    AX,[BX][SI+2]      ;NEXT WORD
0DCF  BA 0000                MOV    DX,0
0DD2  F7 E1                  MUL    CX
0DD4  53                     PUSH   BX                 ;SAVE BASE ADDRESS
0DD5  8D 1E 2282 R           LEA    BX,EXPa_MUL_EXPb_INTEGER
0DD9  03 47 02               ADD    AX,[BX+2]          ;       DX:AX
0DDC  89 47 02               MOV    [BX+2],AX          ;  + DX:AX  ;<---[BX]
0DDF  83 D2 00               ADC    DX,0               ;---------------
0DE2  89 57 04               MOV    [BX+4],DX
0DE5  5B                     POP    BX                 ;RESUME BASE ADDRESS

0DE6  8B 40 04               MOV    AX,[BX][SI+4]      ;NEXT WORD
0DE9  BA 0000                MOV    DX,0
0DEC  F7 E1                  MUL    CX
0DEE  53                     PUSH   BX
0DEF  8D 1E 2282 R           LEA    BX,EXPa_MUL_EXPb_INTEGER
0DF3  03 47 04               ADD    AX,[BX+4]          ;       DX:AX
0DF6  89 47 04               MOV    [BX+4],AX          ;  + DX:AX  ;<---[BX]
0DF9  83 D2 00               ADC    DX,0               ;+DX:AX
0DFC  89 57 06               MOV    [BX+6],DX          ;---------------------
0DFF  5B                     POP    BX

0E00  8B 00                  MOV    AX,[BX][SI]        ;POINT TO THE TABLE
0E02  8D 3E 209C R           LEA    DI,EXP_N           ;FRAGMENT ADDRESS OF EXPb
0E06  BA 0000                MOV    DX,0
0E09  8B 0D                  MOV    CX,[DI]
0E0B  F7 E1                  MUL    CX                 ;NOW THE RESULT WAS IN DX:AX
0E0D  53                     PUSH   BX                 ;SAVE BASE ADDRESS
0E0E  8D 1E 2292 R           LEA    BX,EXPa_MUL_EXPb_FRAGMENT
0E12  89 07                  MOV    [BX],AX
0E14  89 57 02               MOV    [BX+2],DX
0E17  5B                     POP    BX                 ;RESUME BASE ADDRESS

0E18  8B 40 02               MOV    AX,[BX][SI+2]      ;NEXT WORD
0E1B  BA 0000                MOV    DX,0
0E1E  F7 E1                  MUL    CX
0E20  53                     PUSH   BX                 ;SAVE BASE ADDRESS
0E21  8D 1E 2292 R           LEA    BX,EXPa_MUL_EXPb_FRAGMENT
0E25  03 47 02               ADD    AX,[BX+2]          ;       DX:AX
0E28  89 47 02               MOV    [BX+2],AX          ;  + DX:AX  ;<---[BX]
0E2B  83 D2 00               ADC    DX,0               ;---------------
0E2E  89 57 04               MOV    [BX+4],DX
0E31  5B                     POP    BX                 ;RESUME BASE ADDRESS

0E32  8B 40 04               MOV    AX,[BX][SI+4]      ;NEXT WORD
0E35  BA 0000                MOV    DX,0
0E38  F7 E1                  MUL    CX
0E3A  53                     PUSH   BX
0E3B  8D 1E 2292 R           LEA    BX,EXPa_MUL_EXPb_FRAGMENT
0E3F  03 47 04               ADD    AX,[BX+4]          ;       DX:AX
0E42  89 47 04               MOV    [BX+4],AX          ;  + DX:AX  ;<---[BX]
0E45  83 D2 00               ADC    DX,0               ;+DX:AX
0E48  89 57 06               MOV    [BX+6],DX          ;---------------------
0E4B  5B                     POP    BX
```

```
                                        ;----EXPa * EXPb INTEGER PLUS EXPa * EXPb FRAGMENT----
 0E4C  8D 36 2282 R                     LEA     SI,EXPa_MUL_EXPb_INTEGER
 0E50  8D 3E 2294 R                     LEA     DI,EXPa_MUL_EXPb_FRAGMENT+2 ;KEEP 16 BIT FRAGMENT
 0E54  8D 1E 209C R                     LEA     BX,EXP_N

 0E58  8B 04                            MOV     AX,[SI]
 0E5A  03 05                            ADD     AX,[DI]
 0E5C  89 07                            MOV     [BX],AX

 0E5E  8B 44 02                         MOV     AX,[SI+2]
 0E61  13 45 02                         ADC     AX,[DI+2]
 0E64  89 47 02                         MOV     [BX+2],AX

 0E67  8B 44 04                         MOV     AX,[SI+4]
 0E6A  13 45 04                         ADC     AX,[DI+4]
 0E6D  89 47 04                         MOV     [BX+4],AX

 0E70  8B 44 06                         MOV     AX,[SI+6]
 0E73  13 45 06                         ADC     AX,[DI+6]
 0E76  89 47 06                         MOV     [BX+6],AX

 0E79  C3                               RET

                              EXP_a_PLUS_b ENDP
                              ;
                              ;
 0E7A                         EXP_a_PLUS_b_NEGA  PROC      NEAR    ;SUBROUTINE No. 32 FOR THE RTDP.
                                        ;-------------------------------------------------------;
                                        ; THIS SUBROUTINE WAS TO CALCULATE EXP (a+b). a+b<0    ;
                                        ; a & b ARE INTEGER AND DISIMAL PART RESPECTIVELLY.     ;
                                        ; 1.EXP a WILL BE FOUND IN LOOK UP TABLE                ;
                                        ; 2.EXP b WILL BE CALCULATED WITH INTERACTIVE OPRATION  ;
                                        ; 3.BEFOR CALCULATION THE DATA TO BE CONVERTED WAS AT   ;
                                        ;   [SI]                                                ;
                                        ; 4.THE RESULT THEN WAS PUT IN [BX] i.e. [EXP_N] FRAC   ;
                                        ;   [EXP_N+2] INTEGER                                   ;
                                        ;-------------------------------------------------------;
                                        ;----------THE FIRST PART WAS EXP(b) -----------
 0E7A  8D 1E 209C R                     LEA     BX,EXP_N        ;RESULT INTEGER ADDRESS
 0E7E  B8 0001                          MOV     AX,01           ;THE FIRST ITERM
 0E81  89 47 02                         MOV     [BX+2],AX       ;[EXP_N+2]:[EXP_N]
                                                                ;INTEGER:FRAGMENT
 0E84  B8 0000                          MOV     AX,0            ;CLEAR AX FOR SUBTRACT
 0E87  2B 04                            SUB     AX,[SI]         ;1-D_BIN+...
 0E89  89 07                            MOV     [BX],AX         ;RESULT IN [EXP_N] LOOKS LIKE 1.b
 0E8B  B8 0000                          MOV     AX,0
 0E8E  8B 57 02                         MOV     DX,[BX+2]       ;INTEGER PART IN DX
 0E91  1B D0                            SBB     DX,AX           ;SUBTRACT THE CARRY
 0E93  89 57 02                         MOV     [BX+2],DX       ;INTEGRER PART

 0E96  8B 04                            MOV     AX,[SI]         ;FETCH b (BIN)
 0E98  BA 0000                          MOV     DX,0            ;CLEAR DX FOR DIV REMAINDER
 0E9B  B9 0001                          MOV     CX,1            ; ^2
 0E9E  E8 0F92 R                        CALL    DATA_POWER      ; DATA POWER
 0EA1  B9 0002                          MOV     CX,CM_2         ; 2!
 0EA4  F7 F1                            DIV     CX              ; DATA POWER / 2!
 0EA6  03 07                            ADD     AX,[BX]         ; +D_BIN^2/2!
 0EA8  89 07                            MOV     [BX],AX         ;THE RESULT NOW IN [EXP_N]
 0EAA  8B 47 02                         MOV     AX,[BX+2]       ;INTEGER PART IN AX
 0EAD  15 0000                          ADC     AX,0            ;ADD THE CARRY
 0EB0  89 47 02                         MOV     [BX+2],AX       ;NEW INTEGER PART

 0EB3  8B 04                            MOV     AX,[SI]         ;FETCH b (BIN)
 0EB5  BA 0000                          MOV     DX,0            ;CLEAR DX FOR DIV REMAINDER
 0EB8  B9 0002                          MOV     CX,2            ; ^3
 0EBB  E8 0F92 R                        CALL    DATA_POWER      ; DATA POWER
```

```
0EBE  B9 0006          MOV     CX,CM_3         ; 3!
0EC1  F7 F1            DIV     CX              ; DATA POWER / 3!
0EC3  8B 17            MOV     DX,[BX]
0EC5  2B D0            SUB     DX,AX           ; -D_BIN^2/3!
0EC7  89 17            MOV     [BX],DX         ;THE RESULT NOW IN [EXP_N]
0EC9  8B 57 02         MOV     DX,[BX+2]       ;INTEGER PART IN DX
0ECC  B8 0000          MOV     AX,0            ;ADD THE CARRY
0ECF  1B D0            SBB     DX,AX           ;SUB BORROW
0ED1  89 57 02         MOV     [BX+2],DX       ;NEW INTEGER PART

0ED4  8B 04            MOV     AX,[SI]         ;FETCH b (BIN)
0ED6  BA 0000          MOV     DX,0            ;CLEAR DX FOR DIV REMAINDER
0ED9  B9 0003          MOV     CX,3            ; ^4
0EDC  E8 0F92 R        CALL    DATA_POWER      ; DATA POWER
0EDF  B9 0018          MOV     CX,CM_4         ; 4!
0EE2  F7 F1            DIV     CX              ; DATA POWER / 4!
0EE4  03 07            ADD     AX,[BX]         ; +D_BIN^2/4!
0EE6  89 07            MOV     [BX],AX         ;THE RESULT NOW IN [EXP_N]
0EE8  8B 47 02         MOV     AX,[BX+2]
0EEB  15 0000          ADC     AX,0            ;ADD THE CARRY
0EEE  89 47 02         MOV     [BX+2],AX       ;NEW INTEGER PART

0EF1  8B 04            MOV     AX,[SI]         ;FETCH b (BIN)
0EF3  BA 0000          MOV     DX,0            ;CLEAR DX FOR DIV REMAINDER
0EF6  B9 0004          MOV     CX,4            ; ^5
0EF9  E8 0F92 R        CALL    DATA_POWER      ; DATA POWER
0EFC  B9 0078          MOV     CX,CM_5         ; 5!
0EFF  F7 F1            DIV     CX              ; DATA POWER / 5!
0F01  8B 17            MOV     DX,[BX]
0F03  2B D0            SUB     DX,AX           ; -D_BIN^5/5!
0F05  89 17            MOV     [BX],DX         ;THE RESULT NOW IN [EXP_N]
0F07  8B 57 02         MOV     DX,[BX+2]       ;INTEGER PART ADDRESS
0F0A  B8 0000          MOV     AX,0
0F0D  1B D0            SBB     DX,AX           ;ADD THE CARRY
0F0F  89 57 02         MOV     [BX+2],DX       ;NEW INTEGER PART

0F12  8B 04            MOV     AX,[SI]         ;FETCH b (BIN)
0F14  BA 0000          MOV     DX,0            ;CLEAR DX FOR DIV REMAINDER
0F17  B9 0005          MOV     CX,5            ; ^6
0F1A  E8 0F92 R        CALL    DATA_POWER      ; DATA POWER
0F1D  B9 02D0          MOV     CX,CM_6         ; 6!
0F20  F7 F1            DIV     CX              ; DATA POWER / 6!
0F22  03 07            ADD     AX,[BX]         ; +D_BIN^2/6!
0F24  89 07            MOV     [BX],AX         ;THE RESULT NOW IN [EXP_N]
0F26  8B 47 02         MOV     AX,[BX+2]
0F29  15 0000          ADC     AX,0            ;ADD THE CARRY
0F2C  89 47 02         MOV     [BX+2],AX       ;NEW INTEGER PART

0F2F  8B 04            MOV     AX,[SI]         ;FETCH b (BIN)
0F31  BA 0000          MOV     DX,0            ;CLEAR DX FOR DIV REMAINDER
0F34  B9 0006          MOV     CX,6            ; ^7
0F37  E8 0F92 R        CALL    DATA_POWER      ; DATA POWER
0F3A  B9 13B0          MOV     CX,CM_7         ; 7!
0F3D  F7 F1            DIV     CX              ; DATA POWER / 7!
0F3F  8B 17            MOV     DX,[BX]
0F41  2B D0            SUB     DX,AX           ; -D_BIN^7 / 7!
0F43  89 17            MOV     [BX],DX
0F45  8B 57 02         MOV     DX,[BX+2]       ;INTEGER PART ADDRESS
0F48  B8 0000          MOV     AX,0
0F4B  1B D0            SBB     DX,AX           ;ADD THE CARRY
0F4D  89 57 02         MOV     [BX+2],DX       ;NEW INTEGER PART

0F50  8B 04            MOV     AX,[SI]         ;FETCH b (BIN)
0F52  BA 0000          MOV     DX,0            ;CLEAR DX FOR DIV REMAINDER
0F55  B9 0007          MOV     CX,7            ; ^8
0F58  E8 0F92 R        CALL    DATA_POWER      ; DATA POWER
0F5B  B9 9D80          MOV     CX,CM_8         ; 8!
```

```
0F5E  F7 F1                          DIV     CX              ; DATA POWER / 8!
0F60  03 07                          ADD     AX,[BX]         ; +D_BIN^2/8!
0F62  89 07                          MOV     [BX],AX         ;THE RESULT NOW IN [EXP_N]
0F64  8B 47 02                       MOV     AX,[BX+2]
0F67  15 0000                        ADC     AX,0            ;ADD THE CARRY
0F6A  89 47 02                       MOV     [BX+2],AX       ;NEW INTEGER PART

                  ;--- SO FAR THE RESULT OF EXP b WAS STORED IN [EXP_N+2].[EXP_N] ------;
                  ;    NEXT STEP WAS TO CALCULATE (EXP a) * (EXP b)                     ;
                  ; 1. EXP a(a=0 --  -10) ARE DEFINEDED BY DATA AREA (2 X DD OR 8 BYTE).;
                  ; 2. EXP a * EXP b (FRAGMENT). RASON: EXPa<1, EXPb<1 (a,b<0).         ;
                  ; 3. THE RESULT WAS STORED IN [EXP_N] OF [EXP_N+4 ] [EXP_N+2].[EXP_N] ;
                  ;---------------------------------------------------------------------;
0F6D  8B 5C 02                       MOV     BX,[SI+2]       ;BX WAS BASE ADDRESS FOR LOOK UP
                                                             ;EXP_N
0F70  BE CEB0                        MOV     SI,0CEB0H       ;LEA SI,EXP_N0+100H THE FIRST
                                                             ;ADDRESS OF LOOK UP TABLE
0F73  8B C3                          MOV     AX,BX           ;PROPARE BASE * No. OF BYTE
0F75  B9 0002                        MOV     CX,2
0F78  F7 E1                          MUL     CX              ;BASE * No. OF BYTE
0F7A  8B D8                          MOV     BX,AX
0F7C  8B 00                          MOV     AX,[BX][SI]     ;POINT TO THE TABLE

                                     ;-----MULTIPLY THE FRAGMENT PART OF EXP b-------
0F7E  8D 3E 209C R                   LEA     DI,EXP_N        ;INTEGER ADDRESS OF EXPb
0F82  BA 0000                        MOV     DX,0
0F85  8B 0D                          MOV     CX,[DI]
0F87  F7 E1                          MUL     CX              ;NOW THE RESULT WAS IN DX:AX
0F89  89 15                          MOV     [DI],DX
0F8B  B8 0000                        MOV     AX,0
0F8E  89 45 02                       MOV     [DI+2],AX

0F91  C3                             RET

                  EXP_a_PLUS_b_NEGA          ENDP
                  ;
                  ;
0F92              DATA_POWER         PROC    NEAR            ;SUBROUTINE No. 33 FOR THE RTDP.
                                     ;----------------------------------------------;
                                     ; THIS SUBROUTINE WAS TO CALCULATE THE POWER   ;
                                     ;       OF A DATA                              ;
                                     ; 1. BEROR CALLING N --> CX, DATA ADDRESS -->SI ;
                                     ; 2. THE RESULT WAS REMAINED IN AX             ;
                                     ;----------------------------------------------;

0F92  51          AM:                PUSH    CX
0F93  8B 0C                          MOV     CX,[SI]
0F95  F7 E1                          MUL     CX
0F97  8B C2                          MOV     AX,DX
0F99  BA 0000                        MOV     DX,0
0F9C  59                             POP     CX
0F9D  E2 F3                          LOOP    AM

0F9F  C3                             RET

                  DATA_POWER         ENDP
                  ;
                  ;
0FA0              SIGMA_Z1           PROC    NEAR            ;SUBROUTINE No. 34 FOR THE RTDP.
                                     ;----------------------------------------------;
                                     ;THIS PROGAM WAS TO CALCULATE SISGMA Z1.       ;
                                     ;Z1 WAS THE DATA COLLECTED BY THE SENSORS.     ;
                                     ; 1. BEFOR CALCULATE THE DATA SHOULD BE SENT TO ;
                                     ;     Z1, Z2, Z3 IN RAM                        ;
                                     ; 2. THE RESULT SHOULD BE SENT TO SIG_Z1       ;
                                     ; 3. THE RESULT WILL BE POSITIVE.              ;
                                     ;----------------------------------------------;
```

```
0FA0  8D 36 22E2 R                  LEA     SI,Z1           ;POINT TO FIRST DATA
0FA4  8B 04                         MOV     AX,[SI]
0FA6  03 44 02                      ADD     AX,[SI+2]       ;ADD THE 1st AND THE 2nd DATA
0FA9  03 44 04                      ADD     AX,[SI+4]       ;ADD THE 1st THROUGH THE 3rd DATA
0FAC  8D 3E 22E8 R                  LEA     DI,SIG_Z1       ;POINT TO RESULT ADDRESS SIGMA Z1
0FB0  89 05                         MOV     [DI],AX

0FB2  C3                            RET

                        SIGMA_Z1    ENDP
                        ;
                        ;
0FB3                    SIGMA_X1Z1  PROC    NEAR            ;SUBRUTINE No. 35 FOR THE RTDP.
                                    ;-----------------------------------------------------;
                                    ;THIS PROGRAM WAS TO CALCULATE SIGMA X1Z1 X1 ARE THE  ;
                                    ;LOCATIONS OF THE SENSORS.                            ;
                                    ; SIGMA X1Z1 = -0.D70AH Z1 + 0.E148H Z3               ;
                                    ; 1. THE RESULT WAS SENT TO [SIG_X1Z1]                ;
                                    ; 2. DATA IN [SIG_X1Z1] WAS FRAGMENT PART AND THE DATA;
                                    ;    IN [SIG_X1Z1+2] WAS INTEGER PART.                ;
                                    ; 3. SYMBLE_SIGX1Y1 PLUS (0) OR MINUS (0).            ;
                                    ;-----------------------------------------------------;
0FB3  8D 36 22E2 R                  LEA     SI,Z1           ;POINT TO Z1
0FB7  8B 04                         MOV     AX,[SI]
0FB9  BA 0000                       MOV     DX,00H
0FBC  B9 D70A                       MOV     CX,0D70AH       ;THIS VALUE WAS FRAGMENT TOO
0FBF  F7 E1                         MUL     CX
0FC1  8D 3E 22F0 R                  LEA     DI,SIG_X1Z1     ;POINT TO RESULT DISTINATION
0FC5  89 05                         MOV     [DI],AX
0FC7  89 55 02                      MOV     [DI+2],DX

0FCA  8D 36 22E6 R                  LEA     SI,Z3           ;POINT TO Z3
0FCE  8B 04                         MOV     AX,[SI]
0FD0  BA 0000                       MOV     DX,00H          ;CLEAR DX FOR MULTIPLICATION.
0FD3  B9 E148                       MOV     CX,0E148H       ;THIS VALUE WAS FRAGMENT
0FD6  F7 E1                         MUL     CX

0FD8  2B 05                         SUB     AX,[DI]
0FDA  1B 55 02                      SBB     DX,[DI+2]       ;SUBTRACT WITH BORROW
0FDD  89 05                         MOV     [DI],AX         ;THE RESULT NOW WAS IN [SIG_X1Z1]
0FDF  89 55 02                      MOV     [DI+2],DX

0FE2  7C 03                         JL      NEGA            ;IF NGATIVE
0FE4  EB 21 90                      JMP     POSI            ;IF POSITIVE

0FE7  F7 1D             NEGA:       NEG     WORD PTR[DI]
0FE9  B8 0000                       MOV     AX,0
0FEC  3B 05                         CMP     AX,WORD PTR[DI] ;IF NEG [DI]=0 CARRY
0FEE  74 06                         JZ      NKD             ;NEG [0000H]=FFFFH+1=0000H+CARRY
0FF0  F7 55 02                      NOT     WORD PTR[DI+2]
0FF3  EB 07 90                      JMP     NKD_1
0FF6  F7 55 02          NKD:        NOT     WORD PTR[DI+2]
0FF9  FF 45 02                      INC     WORD PTR[DI+2]
0FFC  8D 3E 22F6 R      NKD_1:      LEA     DI,SYMBLE_SIGX1Z1 ;SYMBLE OF PARAMETER C
1000  C7 05 0001                    MOV     WORD PTR[DI],1  ;NEGATIVE
1004  EB 09 90                      JMP     HIN

1007  8D 3E 22F6 R      POSI:       LEA     DI,SYMBLE_SIGX1Z1 ;PLUS
100B  C7 05 0000                    MOV     WORD PTR[DI],0

100F  C3                HIN:        RET

                        SIGMA_X1Z1  ENDP
                        ;
                        ;
1010                    SIGMA_X1_SQ_2_Z1 PROC NEAR          ;SUBRUTINE No. 36 FOR THE RTDP.
                                    ;-----------------------------------------------------;
```

```
                                     ;THIS PROGRAM WAS TO CALCULATE SIGMA Xi^2*Zi          ;
                                     ; 1. THE RESULT WILL BE PUT IN SIG_Xi_SQ_2_Zi          ;
                                     ; 2. DATA IN [SIG_Xi_SQ_2_Zi] WAS FRAGMENT PART AND    ;
                                     ;     THE DATA IN [SIG_Xi_SQ_2_Zi+2] WAS INTEGER PART.;
                                     ; 3. THE RESULT WILL BE POSITIVE, BECAUSE OF Xi^2.     ;
                                     ;-----------------------------------------------------;
1010  8D 36 22E2 R                   LEA     SI,Zi            ;POINT TO DATA IN Zi.
1014  BA 0000                        MOV     DX,00H
1017  8B 04                          MOV     AX,[SI]
1019  B9 B4A2                        MOV     CX,0B4A2H
101C  F7 E1                          MUL     CX
101E  8D 3E 22F8 R                   LEA     DI,SIG_Xi_SQ2_Zi ;POINT TO DISTINATION STORAGE.
1022  89 05                          MOV     [DI],AX          ;FRAGMENT PART IN AX
1024  89 55 02                       MOV     [DI+2],DX        ;INTEGER PART IN DX

1027  8D 36 22E6 R                   LEA     SI,Z3            ;POINT TO THE DATA IN Z3
102B  BA 0000                        MOV     DX,00H
102E  8B 04                          MOV     AX,[SI]
1030  B9 C63F                        MOV     CX,0C63FH
1033  F7 E1                          MUL     CX
1035  03 05                          ADD     AX,[DI]
1037  13 55 02                       ADC     DX,[DI+2]
103A  89 05                          MOV     [DI],AX          ;SEND THE RESULT (FRAGMENT)
                                                              ;TO DISTINATION
103C  89 55 02                       MOV     [DI+2],DX        ;SEND THE RESULT (INTEGER)
                                                              ;TO DISTINATION
103F  C3                             RET

                             SIGMA_Xi_SQ_2_Zi  ENDP
                             ;
                             ;
1040                         A_PARAMETER  PROC    NEAR         ;SUBROUTINE No. 37 FOR THE RTDP.
                                  ;---------------------------------------------------------------;
                                  ;    THIS SUBROUTINE WAS TO CALCULATE THE PARAMETER A FOR CONCAVE;
                                  ; ARC DIRECTION REGRASSION.                                     ;
                                  ;    A = SIG_Zi + 0.DDCH * SIG_XiZi - 1.5A44H * SIG_Xi^2 Zi     ;
                                  ;  1. THE RESULT WILL BE SENT TO [A_PARA] FRAGMENT & [A_PARA+2] ;
                                  ;     INTEGER.                                                  ;
                                  ;  2. CHECK MSB OF [DI+2]:[DI], IF 1, CALCULATE COMPLIMENT.     ;
                                  ;  3. SEND 1 IN [SYMBLE_A_PARA] IF NEGTIVE, OR 0 IF POSITIVE.   ;
                                  ;---------------------------------------------------------------;
1040  8D 36 22E8 R                  LEA     SI,SIG_Zi        ;POINT TO SIGMA Zi
1044  8B 04                         MOV     AX,[SI]
1046  8D 3E 2314 R                  LEA     DI,A_PARA        ;POIN TO THE RESULT STORAGE ADDR
104A  89 45 02                      MOV     [DI+2],AX        ;THERE WAS ONLY INTEGER PART
104D  BA 0000                       MOV     DX,0             ;CLEAR THE FRAGMENT PART
1050  89 15                         MOV     [DI],DX

                                    ;---------THIS PART WAS TO CALCULATE THE FRAGMENT PART---;
1052  B8 0000                       MOV     AX,0
1055  8D 1E 22F6 R                  LEA     BX,SYMBLE_SIGXiZi
1059  3B 07                         CMP     AX,[BX]          ;SIG_XiZi PLUS OR MINUS
105B  74 03                         JZ      PLUS_1           ;IF POSITIVE
105D  EB 33 90                      JMP     MINUS_1          ;IF NEGATIVE

1060  8D 36 22F0 R           PLUS_1: LEA    SI,SIG_XiZi      ;POINT TO SIGMA XiZi FRAGMENT
1064  BA 0000                       MOV     DX,00H
1067  8B 04                         MOV     AX,[SI]          ;FRAGMENT PART
1069  B9 0DDC                       MOV     CX,0DDCH         ;0.0DDCH
106C  F7 E1                         MUL     CX               ;RESULT NOW IN DX:AX (FRAGMENT)
                                                             ;AX WAS OMMITED (IT IS TOO SMALL)
106E  03 15                         ADD     DX,[DI]          ;ADD DATA (FRAGMENT) IN A_PARA
1070  8B C2                         MOV     AX,DX            ;FRAGMENT PART WAS STILL IN AX
1072  8B 55 02                      MOV     DX,[DI+2]        ;INTEGER PART
1075  83 D2 00                      ADC     DX,00H           ;ADD THE CARRY
1078  89 05                         MOV     [DI],AX          ;SENT THE DATA TO [A_PARA]
107A  89 55 02                      MOV     [DI+2],DX        ;SENT THE DATA TO [A_PARA+2]
```

```
                                    ;---------THIS PART WAS TO CALCULATE THE INTGER PART-----;
107D  8D 36 22F2 R                  LEA     SI,SIG_XiZi+2    ;POINT TO SIGMA XiZi FRAGMENT
1081  8B 04                         MOV     AX,[SI]
1083  F7 E1                         MUL     CX               ;THE RESULT NOW IN
                                                             ;DX:AX INTEGER:FRAGMENT
1035  03 05                         ADD     AX,[DI]          ;ADD THE FRAGMENT PART
1087  13 55 02                      ADC     DX,[DI+2]        ;ADD THE INTEGER PART
108A  89 05                         MOV     [DI],AX          ;SENT THE FRAGMENT PART
108C  89 55 02                      MOV     [DI+2],DX        ;SENT THE INTEGER PART
108F  EB 33 90                      JMP     BIC

1092  8D 36 22F0 R      MINUS_1:    LEA     SI,SIG_XiZi
1096  8B 04                         MOV     AX,[SI]          ;FRAGMENT PART
1098  B9 0DDC                       MOV     CX,0DDCH         ;0.0DDCH
109B  F7 E1                         MUL     CX               ;RESULT NOW IN DX:AX (FRAGMENT)
                                                             ;AX WAS OMMITED (IT IS TOO SMALL)
109D  8B 1D                         MOV     BX,[DI]
109F  2B DA                         SUB     BX,DX            ;SUBTRACT FRAGMENT PART
10A1  89 1D                         MOV     [DI],BX
10A3  8B 5D 02                      MOV     BX,[DI+2]        ;INTEGER PART
10A6  BA 0000                       MOV     DX,00H
10A9  1B DA                         SBB     BX,DX            ;SUBTRACT INTEGER PART WITH CARRY
10AB  89 5D 02                      MOV     [DI+2],BX

10AE  8D 36 22F2 R                  LEA     SI,SIG_XiZi+2    ;POINT TO INTEGER PART
10B2  8B 04                         MOV     AX,[SI]
10B4  F7 E1                         MUL     CX
10B6  8B 1D                         MOV     BX,[DI]
10B8  2B D8                         SUB     BX,AX
10BA  89 1D                         MOV     [DI],BX
10BC  8B 5D 02                      MOV     BX,[DI+2]
10BF  1B DA                         SBB     BX,DX
10C1  89 5D 02                      MOV     [DI+2],BX

                                    ;NEXT PART 1:  (A_PARA) - SIG Xi^2Zi * 1
                                    ;     PART 2:  (A_PARA) - SIG Xi^2Zi * 0.5A44H
                                    ;---------------------------------------------
                                    ;PART 1.
10C4  8D 36 22F8 R      BIC:        LEA     SI,SIG_Xi_SQ2_Zi ;POINT TO SIGMA Xi^2 Zi
10C8  8B 05                         MOV     AX,[DI]          ;THE FRAGMENT OF A_PARA
10CA  2B 04                         SUB     AX,[SI]
10CC  8B 55 02                      MOV     DX,[DI+2]        ;THE INTEGER OF A_PARA
10CF  1B 54 02                      SBB     DX,[SI+2]        ;INTEGER-INTEGER WITH BORROW
10D2  89 05                         MOV     [DI],AX          ;SENT THE RESULT TO A_PARA
10D4  89 55 02                      MOV     [DI+2],DX

                                    ;PART 2.
                                    ;---------THIS PART WAS TO CALCULATE THE FRAGMENT PART---;
10D7  8D 36 22F8 R                  LEA     SI,SIG_Xi_SQ2_Zi ;POINT TO SIGMA Xi^2 Zi
10DB  BA 0000                       MOV     DX,00H
10DE  8B 04                         MOV     AX,[SI]          ;FRAGMENT PART
10E0  B9 5A44                       MOV     CX,5A44H         ;0.0DDCH
10E3  F7 E1                         MUL     CX               ;RESULT NOW IN DX:AX (FRAGMENT)
                                                             ;AX WAS OMMITED
10E5  8B 05                         MOV     AX,[DI]          ;A_PARAMETER NOW WAS IN AX
10E7  2B C2                         SUB     AX,DX            ;SUBTRACT THE DATA (FRAGMENT)
10E9  8B 55 02                      MOV     DX,[DI+2]        ;IN A_PARA INTEGER PART
10EC  83 DA 00                      SBB     DX,00H           ;SUBTRACT THE BORROW
10EF  89 05                         MOV     [DI],AX          ;SENT THE DATA TO [A_PARA]
10F1  89 55 02                      MOV     [DI+2],DX        ;SENT THE DATA TO [A_PARA+2]

                                    ;---------THIS PART WAS TO CALCULATE THE INTGER PART-----;
10F4  8D 36 22FA R                  LEA     SI,SIG_Xi_SQ2_Zi+2 ;POINT TO SIGMA Xi^2 Zi
10F8  8B 04                         MOV     AX,[SI]          ;FRAGMENT
10FA  F7 E1                         MUL     CX               ;RESULT NOW IN DX:AX
                                                             ;INTEGER:FRAGMENT
```

```
 10FC  52                              PUSH     DX               ;PROTECT DX
 10FD  8B 15                           MOV      DX,[DI]          ;FRAGMENT PART OF A_PARAMETER
 10FF  2B D0                           SUB      DX,AX            ;SUBTRACT THE FRAGMENT PART
 1101  8B C2                           MOV      AX,DX            ;FRAGMENT PART NOW IN AX
 1103  8B 4D 02                        MOV      CX,[DI+2]        ;INTEGER PART IN CX
 1106  5A                              POP      DX
 1107  1B CA                           SBB      CX,DX            ;SUBTRACT INTEGER WITH BORROW
 1109  8B D1                           MOV      DX,CX            ;THE INTEGER PART NOW IN DX
 110B  89 05                           MOV      [DI],AX          ;SENT THE FRAGMENT PART
 110D  89 55 02                        MOV      [DI+2],DX        ;SENT THE INTEGER PART

 1110  8B 45 02                        MOV      AX,[DI+2]
 1113  BA 8000                         MOV      DX,8000H
 1116  23 C2                           AND      AX,DX
 1118  3D 8000                         CMP      AX,8000H
 111B  74 03                           JZ       NEGATIVE         ;IF [DI+2]:[DI] NEGATIVE.
 111D  EB 21 90                        JMP      POSITIVE

 1120  F7 1D              NEGATIVE:    NEG      WORD PTR[DI]
 1122  B8 0000                         MOV      AX,0
 1125  3B 05                           CMP      AX,WORD PTR[DI]  ;IF NEG [DI]=0 CARRY
 1127  74 06                           JZ       MKD              ;NEG [0000H]=FFFFH+1=0000H+CARRY
 1129  F7 55 02                        NOT      WORD PTR[DI+2]
 112C  EB 07 90                        JMP      MKD_1
 112F  F7 55 02           MKD:         NOT      WORD PTR[DI+2]
 1132  FF 45 02                        INC      WORD PTR[DI+2]
 1135  8D 3E 231A R       MKD_1:       LEA      DI,SYMBLE_A_PARA ;SYMBLE OF PARAMETER C
 1139  C7 05 0001                      MOV      WORD PTR[DI],1   ;NEGATIVE
 113D  EB 09 90                        JMP      SHIN

 1140  8D 3E 231A R       POSITIVE:    LEA      DI,SYMBLE_A_PARA ;PLUS
 1144  C7 05 0000                      MOV      WORD PTR[DI],0

 1148  C3                 SHIN:        RET

                          A_PARAMETER  ENDP
                          ;
                          ;
 1149                     B_PARAMETER  PROC     NEAR          ;SUBROUTINE No. 38 FOR THE RTDP.
                           ;---------------------------------------------------------------;
                           ;  THIS SUBROUTINE WAS TO CALCULATE THE PARAMETER B FOR CONCAVE  ;
                           ; ARC DIRECTION REGRASSION.                                      ;
                           ; B=0.DDCH * SIG_Z1 + 0.AE91H * SIG_X1Z1 - 0.20CFH * SIG_X1^2 Z1 ;
                           ; 1. THE RESULT WILL BE SENT TO [B_PARA] FRAGMENT & [B_PARA+2]   ;
                           ;    INTEGER.                                                    ;
                           ; 2. CHECK MSB OF [DI+2]:[DI], IF 1, CALCULATE COMPLIMENT.       ;
                           ; 3. SEND 1 IN [SYMBLE_B_PARA] IF NEGTIVE, OR 0 IF POSITIVE.
                           ;---------------------------------------------------------------
                                       ;---------PART 1: 0.DDCH * SIG_Z1
 1149  8D 36 22E8 R                    LEA      SI,SIG_Z1        ;POINT TO SIGMA Z1
 114D  8B 04                           MOV      AX,[SI]
 114F  B9 0DDC                         MOV      CX,0DDCH         ;0.DDCH --> CX
 1152  F7 E1                           MUL      CX               ;PRODUCT NOW IN DX:AX
                                                                 ;INTEGER:FRAGMENT
 1154  8D 3E 231C R                    LEA      DI,B_PARA        ;POINT RESULT [illegible] B_PARA
 1158  89 05                           MOV      [DI],AX          ;SEND FRAGMENT
 115A  89 55 02                        MOV      [DI+2],DX        ;SEND INTEGER

                                       ;--------PART 2: +0.A9E1H * SIG X1Y1
 115D  8D 1E 22F6 R                    LEA      BX,SYMBLE_SIGX1Z1
 1161  B8 0000                         MOV      AX,0
 1164  3B 07                           CMP      AX,[BX]          ;CHECK THE SYMBLE OF SIG X1Z1:
 1166  74 03                           JZ       PLUS_3           ;0 PLUS, 1 MINUS IF POSITIVE
 1168  EB 2B 90                        JMP      MINUS_2          ;IF NEGATIVE

 116B  8D 36 22F0 R       PLUS_3:      LEA      SI,SIG_X1Z1      ;POINT TO SIGMA X1Z1
 116F  8B 04                           MOV      AX,[SI]
```

```
1171  B9 AE91                          MOV    CX,0AE91H        ;0.AE91H --> CX
1174  F7 E1                            MUL    CX               ;DX:AX ARE FRAGMENT AX WAS
                                                               ;OMMITED
1176  03 15                            ADD    DX,[DI]          ;ADD FRAGMENT
1178  89 15                            MOV    [DI],DX
117A  BA 0000                          MOV    DX,0
117D  13 55 02                         ADC    DX,[DI+2]
1180  89 55 02                         MOV    [DI+2],DX        ;SEND INTEGER

1183  8B 44 02                         MOV    AX,[SI+2]        ;INTEGER PART OF SIG X1Z1 IN AX
1186  F7 E1                            MUL    CX               ;DX:AX ARE INTEGER:FRAGMENT
1188  03 05                            ADD    AX,[DI]
118A  13 55 02                         ADC    DX,[DI+2]
118D  89 05                            MOV    [DI],AX
118F  89 55 02                         MOV    [DI+2],DX
1192  EB 30 90                         JMP    BID

1195  8D 36 22F0 R         MINUS_2:    LEA    SI,SIG_X1Z1      ;POINT TO SIGMA X1Z1
1199  8B 04                            MOV    AX,[SI]
119B  B9 AE91                          MOV    CX,0AE91H        ;0.AE91H --> CX
119E  F7 E1                            MUL    CX               ;DX:AX ARE FRAGMENT AX WAS
                                                               ;OMMITED.
11A0  8B 1D                            MOV    BX,[DI]
11A2  2B DA                            SUB    BX,DX
11A4  89 1D                            MOV    [DI],BX
11A6  BA 0000                          MOV    DX,0
11A9  8B 5D 02                         MOV    BX,[DI+2]
11AC  1B DA                            SBB    BX,DX
11AE  89 5D 02                         MOV    [DI+2],BX        ;SEND INTEGER

11B1  8B 44 02                         MOV    AX,[SI+2]        ;INTEGER PART OF SIG X1Z1 IN AX
11B4  F7 E1                            MUL    CX               ;DX:AX ARE INTEGER:FRAGMENT
11B6  8B 1D                            MOV    BX,[DI]
11B8  2B D8                            SUB    BX,AX
11BA  89 1D                            MOV    [DI],BX
11BC  8B 5D 02                         MOV    BX,[DI+2]
11BF  1B DA                            SBB    BX,DX
11C1  89 5D 02                         MOV    [DI+2],BX

                                       ;---------PART 3: -0.20CF * SIG X1^2 Z1
                                       ;---------CALCULATE FRAGMENT PART
11C4  8D 36 22F8 R         BID:        LEA    SI,SIG_X1_SQ2_Z1 ;POINT TO SIGMA X1^2 Z1
11C8  8B 04                            MOV    AX,[SI]
11CA  B9 20CF                          MOV    CX,20CFH         ;0.20CFH --> CX
11CD  F7 E1                            MUL    CX               ;DX:AX ARE FRAGMENT AX WAS
                                                               ;OMMITED.
11CF  8D 3E 231C R                     LEA    DI,B_PARA        ;POINT TO RESULT DISTINATION B_PARA

11D3  8B 1D                            MOV    BX,[DI]          ;FRAGMENT --> BX
11D5  2B DA                            SUB    BX,DX            ;SUBTRACT FRAGMENT
11D7  89 1D                            MOV    [DI],BX
11D9  8B 5D 02                         MOV    BX,[DI+2]        ;ADD INTEGER
11DC  BA 0000                          MOV    DX,00H           ;SEND FRAGMENT
11DF  1B DA                            SBB    BX,DX
11E1  89 5D 02                         MOV    [DI+2],BX        ;SEND INTEGER

                                       ;---------CALCULATE INTEGER PART
11E4  8B 44 02                         MOV    AX,[SI+2]        ;INTEGER OF SIG X1^2 Z1 IN AX
11E7  F7 E1                            MUL    CX               ;DX:AX ARE INTEGER:FRAGMENT
11E9  8B 1D                            MOV    BX,[DI]
11EB  2B D8                            SUB    BX,AX
11ED  89 1D                            MOV    [DI],BX
11EF  8B 5D 02                         MOV    BX,[DI+2]
11F2  1B DA                            SBB    BX,DX
11F4  89 5D 02                         MOV    [DI+2],BX

11F7  8B 45 02                         MOV    AX,[DI+2]
11FA  BA 8000                          MOV    DX,8000H
```

```
 11FD  23 C2                              AND      AX,DX
 11FF  3D 8000                            CMP      AX,8000H
 1202  74 03                              JZ       NEGATIVE_1      ;IF [DI+2]:[DI] NEGATIVE.
 1204  EB 21 90                           JMP      POSITIVE_1

 1207  F7 1D               NEGATIVE_1:    NEG      WORD PTR[DI]
 1209  B8 0000                            MOV      AX,0
 120C  3B 05                              CMP      AX,WORD PTR[DI] ;IF NEG [DI]=0 CARRY
 120E  74 06                              JZ       MKD_2           ;NEG [0000H]=FFFFH+1=0000H+CARRY
 1210  F7 55 02                           NOT      WORD PTR[DI+2]
 1213  EB 07 90                           JMP      MKD_3
 1216  F7 55 02            MKD_2:         NOT      WORD PTR[DI+2]
 1219  FF 45 02                           INC      WORD PTR[DI+2]
 121C  8D 3E 2322 R        MKD_3:         LEA      DI,SYMBLE_B_PARA ;SYMBLE OF PARAMETER C
 1220  C7 05 0001                         MOV      WORD PTR[DI],1  ;NEGATIVE
 1224  EB 09 90                           JMP      SHIN_1

 1227  8D 3E 2322 R        POSITIVE_1:    LEA      DI,SYMBLE_B_PARA ;PLUS
 122B  C7 05 0000                         MOV      WORD PTR[DI],0

 122F  C3                  SHIN_1:        RET

                           B_PARAMETER    ENDP
                           ;
                           ;
 1230                      C_PARAMETER    PROC     NEAR            ;SUBROUTINE No. 39 FOR THE RTDP.
                              ;-------------------------------------------------------------;
                              ;   THIS SUBROUTINE WAS TO CALCULATE THE PARAMETER C FOR CONCAVE ;
                              ;ARC DIRECTION REGRASSION.                                    ;
                              ; C=-1.5A44H * SIG_Z1 -0.20CFH * SIG_X1Z1 +2.BECCH * SIG_X1^2 Z1 ;
                              ; 1. THE RESULT WILL BE SENT TO [C_PARA] FRAGMENT & [C_PARA+2] ;
                              ;    INTEGER.                                                 ;
                              ; 2. THE SYMBLE OF SIG_X1Z1 DETERMIN + OR - OF .20CF*SIG_X1Z1 ;
                              ; 3. CHECK MSB OF [DI+2]:[DI], IF 1, CALCULATE COMPLIMENT.    ;
                              ; 4. SEND 1 IN [SYMBLE_C_PARA] IF NEGTIVE, OR 0 IF POSITIVE.  ;
                              ;-------------------------------------------------------------;
                                          ;PART 1. +2.BECCH * SIG_X1^2 Z1
                                          ;---------(1) CALCULATE 2.0*SIG_X1^2 Z1
 1230  8D 36 22F8 R                       LEA      SI,SIG_X1_SQ2_Z1  ;POINT TO SIG X1^2 Z1
 1234  8D 3E 2324 R                       LEA      DI,C_PARA       ;POINT TO RESULT ADDRESS
 1238  8B 04                              MOV      AX,[SI]         ;FRAGMENT PART OF SIG X1^2Z1
 123A  B9 0002                            MOV      CX,2
 123D  F7 E1                              MUL      CX              ;THE RESULT NOW IN DX:AX
 123F  03 05                              ADD      AX,[DI]
 1241  13 55 02                           ADC      DX,[DI+2]
 1244  89 05                              MOV      [DI],AX
 1246  89 55 02                           MOV      [DI+2],DX

 1249  8B 44 02                           MOV      AX,[SI+2]       ;INTEGER PART OF SIG X1^2 Z1
 124C  F7 E1                              MUL      CX
 124E  03 45 02                           ADD      AX,[DI+2]
 1251  89 45 02                           MOV      [DI+2],AX

                                          ;---------2. CALCULATE 0.BECC * SIG X1^2 Z1
 1254  8B 04                              MOV      AX,[SI]         ;FRAGMENT PART IN AX
 1256  B9 BECC                            MOV      CX,0BECCH
 1259  F7 E1                              MUL      CX              ;DX:AX ARE FRAGMENT AX WAS
                                                                   ;OMMITED.
 125B  03 15                              ADD      DX,[DI]
 125D  89 15                              MOV      [DI],DX
 125F  BA 0000                            MOV      DX,0
 1262  13 55 02                           ADC      DX,[DI+2]       ;INTEGER PART ADD THE CARRY
 1265  89 55 02                           MOV      [DI+2],DX

 1268  8B 44 02                           MOV      AX,[SI+2]       ;INTEGERPART OF SIGMA X1^2 Z1
 126B  F7 E1                              MUL      CX              ;DX:AX INTEGER:FRAGMENT
 126D  03 05                              ADD      AX,[DI]
```

```
126F  13 55 02                          ADC     DX,[DI+2]
1272  89 05                             MOV     [DI],AX
1274  89 55 02                          MOV     [DI+2],DX

                                        ;PART 2: -(1.0 + 0.5A44H) * SIG_Zi--------
1277  8D 36 22E8 R                      LEA     SI,SIG_Zi       ;SIG Zi INTEGER ONLY
127B  8B 04                             MOV     AX,[SI]
127D  B9 5A44                           MOV     CX,5A44H        ;0.5A44 --> CX
1280  F7 E1                             MUL     CX              ;PRODUCT NOW IN DX:AX
1282  8B 1D                             MOV     BX,[DI]         ;FRAGMENT PART IN BX
1284  2B D8                             SUB     BX,AX
1286  89 1D                             MOV     [DI],BX
1288  8B 5D 02                          MOV     BX,[DI+2]       ;INTEGER PART IN BX
128B  1B DA                             SBB     BX,DX
128D  89 5D 02                          MOV     [DI+2],BX
1290  8B 55 02                          MOV     DX,[DI+2]
1293  2B 14                             SUB     DX,[SI]
1295  89 55 02                          MOV     [DI+2],DX

                                        ;PART 3:-(0.20CFH * SIG XiZi)
                                        ;---------(1) CALCULATE FRAGMENT PART OF SIG XiZi
1298  8D 1E 22F6 R                      LEA     BX,SYMBLE_SIGXiZi
129C  B8 0000                           MOV     AX,0
129F  3B 07                             CMP     AX,[BX]
12A1  74 03                             JZ      PLUS_2
12A3  EB 33 90                          JMP     MINUS_3

12A6  8D 36 22F0 R        PLUS_2:       LEA     SI,SIG_XiZi     ;POINT TO SIG_XiZi
12AA  8B 04                             MOV     AX,[SI]         ;FRAGMENT PART
12AC  B9 20CF                           MOV     CX,20CFH
12AF  F7 E1                             MUL     CX              ;PRODUCT DX:AX WAS FRAGMENT
                                                                ;AX OMMITED
12B1  8B 1D                             MOV     BX,[DI]         ;FRAGMENT PART NOW IN BX
12B3  2B DA                             SUB     BX,DX
12B5  89 1D                             MOV     [DI],BX
12B7  8B 5D 02                          MOV     BX,[DI+2]       ;INTEGERPART NOW IN BX
12BA  BA 0000                           MOV     DX,0
12BD  1B DA                             SBB     BX,DX
12BF  89 5D 02                          MOV     [DI+2],BX

                                        ;---------(2) CALCULATE INTEGER PART OF SIG XiZi
12C2  8B 44 02                          MOV     AX,[SI+2]       ;INTEGER PART
12C5  F7 E1                             MUL     CX              ;PRODUCT DX:AX = INTEGER:FRAGMENT
12C7  8B 1D                             MOV     BX,[DI]         ;FRAGMENT PART NOW IN BX
12C9  2B D8                             SUB     BX,AX
12CB  89 1D                             MOV     [DI],BX
12CD  8B 5D 02                          MOV     BX,[DI+2]       ;INTEGER PART NOW IN BX
12D0  1B DA                             SBB     BX,DX
12D2  89 5D 02                          MOV     [DI+2],BX
12D5  EB 28 90                          JMP     HEI

12D8  8D 36 22F0 R        MINUS_3:      LEA     SI,SIG_XiZi     ;POINT TO SIG_XiZi
12DC  8B 04                             MOV     AX,[SI]         ;FRAGMENT PART
12D[illegible]  [illegible] 20CF        MOV     CX,20CFH
12[illegible]                           MUL     CX              ;PRODUCT DX:AX WAS FRAGMENT
12[illegible]                           ADD     DX,[DI]         ;AX OMMITED
12E5                                    MOV     [DI],DX
12E7                                    MOV     DX,0
12EA                                    ADC     DX,[DI+2]
12ED                                    MOV     [DI+2],DX
                                        ;---------INTEGER PART
12F0  8[illegible]                      MOV     AX,[SI+2]
12F3  F7 E1                             MUL     CX
12F5  03 05                             ADD     AX,[DI]
12F7  13 55 02                          ADC     DX,[DI+2]
12FA  89 05                             MOV     [DI],AX
12FC  89 55 02                          MOV     [DI+2],DX
```

```
12FF  8B 45 02        HEI:         MOV    AX,[DI+2]
1302  BA 8000                      MOV    DX,8000H
1305  23 C2                        AND    AX,DX
1307  3D 8000                      CMP    AX,8000H
130A  74 03                        JZ     NEGATIVE_2        ;IF [DI+2]:[DI] NEGATIVE.
130C  EB 21 90                     JMP    POSITIVE_2        ;JL IF LESS
130F  F7 [illegible]  NEGATIVE_2:  NEG    WORD PTR[DI]
1311  B8 [illegible]               MOV    AX,0
1314  3B [illegible]               CMP    AX,WORD PTR[DI] ;IF NEG [DI]=0 CARRY
1316  74 06                        JZ     MKD_4             ;NEG [0000H]=FFFFH+1=0000H+CARRY
1318  F7 55 02                     NOT    WORD PTR[DI+2]
131B  EB 07 90                     JMP    MKD_5
131E  F7 55 02        MKD_4:       NOT    WORD PTR[DI+2]
1321  FF 45 02                     INC    WORD PTR[DI+2]
1324  8D 3E 232A R    MKD_5:       LEA    DI,SYMBLE_C_PARA ;SYMBLE OF PARAMETER C
1328  C7 05 0001                   MOV    WORD PTR[DI],1  ;NEGATIVE
132C  EB 09 90                     JMP    SHIN_2

[illegible] [illegible] 3E 23?A R  POSITIVE_2: LEA  DI,SYMBLE_C_PARA ;PLUS
[illegible] [illegible] 05 00[illegible]       MOV  WORD PTR[DI],0

[illegible]           SHIN_2:      RET

                      C_PARAMETER  ENDP
                      ;
                      ;
1338                  Z_MEAN_VALUE PROC    NEAR              ;SUBROUTINE No. 40 FOR THE RTDP.
                          ;-------------------------------------------------------------;
                          ;  THIS SUBROUTINE WAS TO CALCULATE THE MEAN VALUE OF Z1       ;
                          ;   Z_MEAN = A_PARA + 0.8F6H * B_PARA + 0.6882H * C_PARA       ;
                          ; 1. THE SYMBLES OF A B & C MUST BE CONSIDERD, THESE SYMBLES   ;
                          ;    WILL DETERMIN THE CALCULATION WAS ADD OR SUB.             ;
                          ; 2. THE RESULT WILL BE SENT TO [Z_MEAN] FRAGMENT & [Z_MEAN+2] ;
                          ;    INTEGER.                                                  ;
                          ; 3. CHECK [Z_MEAN], IF IT WAS EQUAL OR ABOVE 0.1000H (0.5),   ;
                          ;    [Z_MEAN+2]+1, OTHERWISE [Z_MEAN] WAS OMITTED.             ;
                          ;-------------------------------------------------------------;
                                   ;PART 1. A_PARA
1338  8D 1E 231A R                 LEA    BX,SYMBLE_A_PARA ;POINT TO A'S SYMBLE
133C  8B 07                        MOV    AX,[BX]
133E  3D 0000                      CMP    AX,0
1341  74 03                        JZ     PLUS_4
1343  EB 16 90                     JMP    MINUS_4

1346  8D 36 2314 R    PLUS_4:      LEA    SI,A_PARA         ;POINT TO A PARAMETER
134A  8D 3E 230C R                 LEA    DI,Z_MEAN         ;POINT TO MEAN VALUE OF Z1
134E  8B 04                        MOV    AX,[SI]           ;SEND A PARA TO Z_MEAN
1350  89 05                        MOV    [DI],AX           ;(FRAGMENT PART)
1352  8B 44 02                     MOV    AX,[SI+2]         ;SEND A PARAMETER INTEGER PART
1355  89 45 02                     MOV    [DI+2],AX
1358  EB 19 90                     JMP    PART_2

135B  8D 36 2314 R    MINUS_4:     LEA    SI,A_PARA
135F  8D 3E 230C R                 LEA    DI,Z_MEAN
1363  B8 0000                      MOV    AX,00H
1366  BA 0000                      MOV    DX,00H
1369  2B 04                        SUB    AX,[SI]           ;SUBTRACT FRAGMENT PART
136B  1B 54 02                     SBB    DX,[SI+2]         ;SUBTRACT INTEGER PART
136E  89 05                        MOV    [DI],AX
1370  89 55 02                     MOV    [DI+2],DX

                                   ;PART 2. 0.8F6H * B_PARA
1373  8D 1E 2322 R    PART_2:      LEA    BX,SYMBLE_B_PARA ;POINT TO A'S SYMBLE
1377  8B 07                        MOV    AX,[BX]
1379  3D 0000                      CMP    AX,0
137C  74 03                        JZ     PLUS_5
```

```
137E  EB 2B 90                       JMP     MINUS_5

1381  8D 36 231C R     PLUS_5:       LEA     SI,B_PARA       ;POINT TO B PARAMETER
1385  8B 04                          MOV     AX,[SI]         ;FRAGMENT PART IN AX
1387  B9 08F6                        MOV     CX,8F6H
138A  F7 E1                          MUL     CX              ;DX:AX ARE FRAGMENT AX WAS
                                                             ;OMMITED.
138C  03 15                          ADD     DX,[DI]
138E  89 15                          MOV     [DI],DX         ;FRAGMENT OF RESULT
1390  BA 0000                        MOV     DX,0
1393  13 55 02                       ADC     DX,[DI+2]       ;INTEGER PART ADD THE CARRY
1396  89 55 02                       MOV     [DI+2],DX       ;SEND INTEGER RESULT

1399  8B 44 02                       MOV     AX,[SI+2]       ;INTEGER PART OF B_PARA IN AX
139C  F7 E1                          MUL     CX              ;DX:AX ARE INTEGER:FRAGMENT
139E  03 05                          ADD     AX,[DI]
13A0  13 55 02                       ADC     DX,[DI+2]
13A3  89 05                          MOV     [DI],AX
13A5  89 55 02                       MOV     [DI+2],DX
13A8  EB 30 90                       JMP     PART_3

13AB  8D 36 231C R     MINUS_5:      LEA     SI,B_PARA       ;POINT TO B PARAMETER
13AF  8B 04                          MOV     AX,[SI]         ;FRAGMENT PART IN AX
13B1  B9 08F6                        MOV     CX,8F6H
13B4  F7 E1                          MUL     CX              ;DX:AX ARE FRAGMENT AX WAS
                                                             ;OMMITED.
13B6  8B 1D                          MOV     BX,[DI]         ;FRAGMENT SEND TO BX
13B8  2B DA                          SUB     BX,DX
13BA  89 1D                          MOV     [DI],BX         ;FRAGMENT OF RESULT
13BC  BA 0000                        MOV     DX,0
13BF  8B 5D 02                       MOV     BX,[DI+2]
13C2  1B DA                          SBB     BX,DX           ;INTEGER PART SUB WITH THE BORROW
13C4  89 5D 02                       MOV     [DI+2],BX       ;SEND INTEGER RESULT

13C7  8B 44 02                       MOV     AX,[SI+2]       ;INTEGER PART OF B_PARA IN AX
13CA  F7 E1                          MUL     CX              ;DX:AX ARE INTEGER:FRAGMENT
13CC  8B 1D                          MOV     BX,[DI]
13CE  2B D8                          SUB     BX,AX
13D0  89 1D                          MOV     [DI],BX
13D2  8B 5D 02                       MOV     BX,[DI+2]
13D5  1B DA                          SBB     BX,DX
13D7  89 5D 02                       MOV     [DI+2],BX

                                     ;PART 3. 0.6882* C_PARA
13DA  8D 1E 232A R     PART_3:       LEA     BX,SYMBLE_C_PARA ;POINT TO C'S SYMBLE
13DE  8B 07                          MOV     AX,[BX]
13E0  3D 0000                        CMP     AX,0
13E3  74 03                          JZ      PLUS_6
13E5  EB 2B 90                       JMP     MINUS_6

13E8  8D 36 2324 R     PLUS_6:       LEA     SI,C_PARA       ;POINT TO C PARAMETER
13EC  8B 04                          MOV     AX,[SI]         ;FRAGMENT PART OF C PARA IN AX
13EE  B9 6882                        MOV     CX,6882H
13F1  F7 E1                          MUL     CX              ;DX:AX ARE FRAGMENT AX WAS
                                                             ;OMMITED.
13F3  03 15                          ADD     DX,[DI]
13F5  89 15                          MOV     [DI],DX
13F7  BA 0000                        MOV     DX,0
13FA  13 55 02                       ADC     DX,[DI+2]
13FD  89 55 02                       MOV     [DI+2],DX

1400  8B 44 02                       MOV     AX,[SI+2]       ;INTEGER PART OF C_PARA IN AX
1403  F7 E1                          MUL     CX              ;DX:CX ARE INTEGER:FRAGMENT
1405  03 05                          ADD     AX,[DI]
1407  13 55 02                       ADC     DX,[DI+2]
140A  89 05                          MOV     [DI],AX
140C  89 55 02                       MOV     [DI+2],DX
```

```
 140F  EB 30 90                           JMP     POSF

 1412  8D 36 2324 R       MINUS_6:        LEA     SI,C_PARA        ;POINT TO C PARAMETER
 1416  8B 04                              MOV     AX,[SI]          ;FRAGMENT PART OF C PARA IN AX
 1418  B9 6882                            MOV     CX,6882H
 141B  F7 E1                              MUL     CX               ;DX:AX ARE FRAGMENT AX WAS
                                                                   ;OMMITED.
 141D  8B 1D                              MOV     BX,[DI]          ;THE FRAGMENT IN BX
 141F  2B DA                              SUB     BX,DX
 1421  89 1D                              MOV     [DI],BX
 1423  BA 0000                            MOV     DX,0
 1426  8B 5D 02                           MOV     BX,[DI+2]
 1429  1B DA                              SBB     BX,DX
 142B  89 5D 02                           MOV     [DI+2],BX

 142E  8B 44 02                           MOV     AX,[SI+2]        ;INTEGER PART OF C_PARA IN AX
 1431  F7 E1                              MUL     CX               ;DX:CX ARE INTEGER:FRAGMENT
 1433  8B 1D                              MOV     BX,[DI]
 1435  2B D8                              SUB     BX,AX
 1437  89 1D                              MOV     [DI],BX
 1439  8B 5D 02                           MOV     BX,[DI+2]
 143C  1B DA                              SBB     BX,DX
 143E  89 5D 02                           MOV     [DI+2],BX

 1441  C3                 POSF:           RET

                          Z_MEAN_VALUE ENDP
                          ;
                          ;
 1442                     SENSOR_MOVING  PROC    NEAR             ;SUBROUTINE No. 41 FOR THE RTDP.
                                          ;---------------------------------------------;
                                          ;THIS PROCEDURE WAS TO MOVE SENSOR & Z_MEAN_i;
                                          ;DATA FOR UP LOADING TO AN IBM-PC COMPUTER   ;
                                          ;---------------------------------------------;
 1442  B9 000C                            MOV     CX,12            ;NINE SENSORS+3 MEAN VALUES.
 1445  8D 3E 2425 R                       LEA     DI,S1            ;DISTINATION IN TABLE.
 1449  89 3E 20A8 R                       MOV     [DI_MEM],DI      ;STORE IN [DI_MEM].
 144D  8D 36 2330 R                       LEA     SI,SENSOR_1      ;THE DATA TO BE CONVERTED.
 1451  8D 1E 20AA R       ZPP:            LEA     BX,S1_TEMP       ;FINAL RESULT TEMPERARY STORAGE.
 1455  51                                 PUSH    CX
 1456  E8 0B38 R                          CALL    BIN_BCD_INTEGER
 1459  59                                 POP     CX
 145A  51                                 PUSH    CX
 145B  83 F9 03                           CMP     CX,3
 145E  7E 03                              JLE     FFI_2            ;IF Z_MEAN_1 -- 3.
 1460  EB 04 90                           JMP     FFI_3            ;IF SENSORS.
 1463  83 C6 06           FFI_2:          ADD     SI,6             ;POINT TO NEXT Z_MEAN_i.
 1466  46                 FFI_3:          INC     SI               ;POINT TO NEXT SENSOR.
 1467  46                                 INC     SI
 1468  56                                 PUSH    SI
 1469  FC                                 CLD
 146A  8B 3E 20A8 R                       MOV     DI,[DI_MEM]      ;DISTINATION ADDRSESS MEMORY.
 146E  8D 36 20AB R                       LEA     SI,S1_TEMP+1
 1472  B9 0004                            MOV     CX,4
 1475  F3/ A4             REP             MOVSB
 1477  5E                                 POP     SI
 1478  59                                 POP     CX
 1479  83 F9 04                           CMP     CX,4             ;WAS SENSORS DATA CONVERTION
                                                                   ;FINISHED?
 147C  74 05                              JE      FFI              ;FINISHED.
 147E  7F 12                              JG      FFI_8            ;NO, GO ON NEXT SENSORS CONVERT.
 1480  EB 1E 90                           JMP     FFI_4
 1483  8D 36 238C R       FFI:            LEA     SI,Z_MEAN_1      ;YES Z_MEAN CONCONVERT.
 1487  8D 3E 2517 R                       LEA     DI,ZZ1+1
 148B  89 3E 20A8 R                       MOV     [DI_MEM],DI
 148F  EB 22 90                           JMP     FFI_1
 1492  8B 3E 20A8 R       FFI_8:          MOV     DI,[DI_MEM]
```

```
1496  83 C7 10                          ADD     DI,16
1499  89 3E 20A8 R                      MOV     [DI_MEM],DI
149D  EB 14 90                          JMP     FFI_1
14A0  83 F9 03          FFI_4:          CMP     CX,3
14A3  7E 03                             JLE     FFI_5
14A5  EB 0C 90                          JMP     FFI_1
14A8  8B 3E 20A8 R      FFI_5:          MOV     DI,[DI_MEM]
14AC  83 C7 60                          ADD     DI,32*3
14AF  89 3E 20A8 R                      MOV     [DI_MEM],DI

14B3  E2 9C             FFI_1:          LOOP    ZPP

14B5  C3                                RET

                        SENSOR_MOVING           ENDP
                        ;
                        ;
14B6                    ABC_MOVING      PROC    NEAR            ;SUBROUTINE No. 42 FOR THE RTDP.
                                        ;------------------------------------------;
                                        ; MOVING A, B & C PARAMETERS TO OUTPUT AREA ;
                                        ;------------------------------------------;
14B6  8D 36 2344 R                      LEA     SI,A_PARA_1     ;THE DATA TO BE CONVERTED
14BA  8D 3E 24E5 R                      LEA     DI,A1           ;DISTINATION IN TABLE
14BE  89 3E 20A8 R                      MOV     [DI_MEM],DI     ;STORE IN [DI_MEM]
14C2  B9 0009                           MOV     CX,9            ;9 PARAMETERS (A1---C3)
14C5  8D 1E 20AA R      PPPZ:           LEA     BX,S1_TEMP      ;THE FINAL RESULT TEMPERARY
                                                                ;STORSGE
14C9  8D 3E 2252 R                      LEA     DI,D_BCD_FRACTION ;DATA IN [D_BCD_FRACTION] NEED
                                                                ; TO * 0.000015258

14CD  51                                PUSH    CX
14CE  56                                PUSH    SI
14CF  E8 0A1A R                         CALL    BIN_BCD
14D2  5E                                POP     SI              ;POINT TO DATA TO BE CONVERTED
14D3  8B 44 06                          MOV     AX,[SI+6]       ;FETCH THE SIGN WORD
14D6  3D 0001                           CMP     AX,01
14D9  74 03                             JZ      MINUS_SIGN
14DB  EB 0A 90                          JMP     POSI_SIGN
14DE  8B 3E 20A8 R      MINUS_SIGN:     MOV     DI,[DI_MEM]
14E2  B0 2D                             MOV     AL,'-'
14E4  EB 07 90                          JMP     S_SIGN
14E7  8B 3E 20A8 R      POSI_SIGN:      MOV     DI,[DI_MEM]
14EB  B0 2B                             MOV     AL,'+'
14ED  88 05             S_SIGN:         MOV     [DI],AL
14EF  83 C6 18                          ADD     SI,18H          ;POINT TO NEXT PARAMETER
14F2  56                                PUSH    SI
14F3  E8 157A R                         CALL    M_F
14F6  83 C7 60                          ADD     DI,32*3
14F9  89 3E 20A8 R                      MOV     [DI_MEM],DI     ;NEXT DISTINATION ADDRESS
                                                                ;IN [DI_MEM]

14FD  5E                                POP     SI
14FE  59                                POP     CX
14FF  83 F9 07                          CMP     CX,7
1502  74 03                             JZ      NDJ
1504  EB 0D 90                          JMP     NDJ_1
1507  8D 36 234C R      NDJ:            LEA     SI,B_PARA_1
150B  8D 3E 24F6 R                      LEA     DI,B11
150F  89 3E 20A8 R                      MOV     [DI_MEM],DI
1513  83 F9 04          NDJ_1:          CMP     CX,4
1516  74 03                             JZ      NDJ_2
1518  EB 12 90                          JMP     NDJ_3
151B  8D 36 2354 R      NDJ_2:          LEA     SI,C_PARA_1
151F  8D 3E 2505 R                      LEA     DI,C11
1523  89 3E 20A8 R                      MOV     [DI_MEM],DI
1527  EB 03 90                          JMP     NDJ_3
152A  EB 99             PPP:            JMP     PPPZ
152C  E2 FC             NDJ_3:          LOOP    PPP
```

```
 152E  C3                                    RET

                         ABC_MOVING    ENDP
                         ;
                         ;
 152F                    Z_MEAN_Ln_MOV  PROC   NEAR          ;SUBROUTINE No. 43 FOR THE RTDP.
                                       ;--------------------------------;
                                       ; 1. CONVERT Ln Z_MEAN TO ASCII ;
                                       ; 2. MOV ASCII TO UP LOAD AREA  ;
                                       ;--------------------------------;
 152F  B9 000C                         MOV    CX,12         ;12 WORDS
 1532  8D 36 23A4 R                    LEA    SI,Z_MEAN_1_L ;POINT TO DATA
 1536  BA 3030                         MOV    DX,3030H
 1539  8B 04             ZSS:          MOV    AX,[SI]
 153B  0B C2                           OR     AX,DX
 153D  89 04                           MOV    [SI],AX
 153F  46                              INC    SI            ;POINT TO NEXT DATA
 1540  46                              INC    SI
 1541  E2 F6                           LOOP   ZSS
 1543  8D 3E 260E R                    LEA    DI,LnZ1+8
 1547  8D 36 23A4 R                    LEA    SI,Z_MEAN_1_L ;POINT TO DATA
 154B  E8 1565 R                       CALL   F_UP

 154E  8D 3E 261F R                    LEA    DI,LnZ2+8
 1552  8D 36 23AC R                    LEA    SI,Z_MEAN_2_L ;POINT TO DATA
 1556  E8 1565 R                       CALL   F_UP

 1559  8D 3E 262E R                    LEA    DI,LnZ3+8
 155D  8D 36 23B4 R                    LEA    SI,Z_MEAN_3_L ;POINT TO DATA
 1561  E8 1565 R                       CALL   F_UP

 1564  C3                              RET

                         Z_MEAN_Ln_MOV        ENDP
                         ;
                         ;
 1565                    F_UP          PROC   NEAR          ;SUBROUTINE No. 44 FOR THE RTDP.

 1565  B9 0008                         MOV    CX,8
 1568  83 F9 04          ZSS_1:        CMP    CX,4
 156B  74 03                           JE     F_U_1
 156D  EB 02 90                        JMP    F_U_2
 1570  4F                F_U_1:        DEC    DI
 1571  8A 04             F_U_2:        MOV    AL,[SI]
 1573  88 05                           MOV    [DI],AL
 1575  46                              INC    SI
 1576  4F                              DEC    DI
 1577  E2 EF                           LOOP   ZSS_1

 1579  C3                              RET

                         F_UP          ENDP
                         ;
                         ;
 157A                    M_F           PROC   NEAR          ;SUBROUTINE No. 45 FOR THE RTDP.

 157A  8B 3E 20A8 R                    MOV    DI,[DI_MEM]
 157E  A1 2242 R                       MOV    AX,[D_BCD]    ;FRAGMENT PART
 1581  0D 3030                         OR     AX,3030H      ;BCD-ASCII
 1584  A3 20AA R                       MOV    [S1_TEMP],AX
 1587  88 45 09                        MOV    [DI+9],AL
 158A  88 65 08                        MOV    [DI+8],AH
 158D  A1 2244 R                       MOV    AX,[D_BCD+2]
 1590  0D 3030                         OR     AX,3030H
 1593  A3 20AC R                       MOV    [S1_TEMP+2],AX
 1596  88 45 07                        MOV    [DI+7],AL
 1599  88 65 06                        MOV    [DI+6],AH
```

```
159C  8D 36 20AE R                    LEA     SI,S1_TEMP+4    ;INTEGRER PART
15A0  8B 04                           MOV     AX,[SI]         ;BCD--ASCII
15A2  0D 3030                         OR      AX,3030H
15A5  89 04                           MOV     [SI],AX
15A7  88 45 04                        MOV     [DI+4],AL
15AA  88 65 03                        MOV     [DI+3],AH
15AD  8B 44 02                        MOV     AX,[SI+2]
15B0  0D 3030                         OR      AX,3030H
15B3  89 44 02                        MOV     [SI+2],AX
15B6  88 45 02                        MOV     [DI+2],AL
15B9  88 65 01                        MOV     [DI+1],AH

15BC  C3                              RET
                        M_F           ENDP
                        ;
                        ;
                        ;------------------------------------------------------------;
                        ;             PART 3. MINI-OPERATION SYSTEM                  ;
                        ;------------------------------------------------------------;
15BD                    KEY_SCANNING PROC     NEAR
                                      ;------------------------------------------------------;
                                      ; This program was for keyboad and liquid crystal display. ;
                                      ; When the key was pressed the coresponding letter will be ;
                                      ; displaied on LCD.                                        ;
                                      ;------------------------------------------------------;
15BD  B0 90             COUN:         MOV     AL,CWIOO        ;8255-1 I/O A IN, I/O B, C OUT
15BF  E6 03                           OUT     IOCTR,AL

15C1  B0 00                           MOV     AL,00H
15C3  A2 2027 R                       MOV     [COUTER_A],AL   ;ROL TIMES FOR IOA
15C6  A2 2028 R                       MOV     [COUTER_B],AL   ;ROL TIMES FOR IOB
15C9  B8 0000                         MOV     AX,0000H        ;CLEAR KEY NUMBER
15CC  A3 202D R                       MOV     [KEY_NUMBER],AX

15CF  B0 FE                           MOV     AL,11111110B
15D1  A2 2029 R                       MOV     [ALM],AL        ;KEEP THE FIRST KEY SCANNING
15D4  A2 202A R                       MOV     [BLM],AL        ;STATUS

15D7  A0 202A R         PORT_B:       MOV     AL,[BLM]        ;3 MSB OF PORT B ARE LCD CONTROL
15DA  24 7F                           AND     AL,01111111B    ;MASK 1 MSB
15DC  E6 01                           OUT     IOB,AL          ;LINE FIRST LINE CONNNECTERD TO B=0

15DE  E4 00             PORT_A:       IN      AL,IOA          ;PORT A SCANNING
15E0  3A 06 2029 R                    CMP     AL,[ALM]        ;COLUNM CONNECTED TO A PRESSED?
15E4  75 23                           JNZ     RCL_A           ; NO ROTAT PORT A
15E6  8B 3E 202D R                    MOV     DI,[KEY_NUMBER] ;KEY NUMBER INC ONE
15EA  47                              INC     DI
15EB  89 3E 202D R                    MOV     [KEY_NUMBER],DI
15EF  E8 1644 R                       CALL    LOOK_UP_1       ;LOOK UP TABLE SUBROUTINE
15F2  B9 0014                         MOV     CX,14H          ;BOUNCING AVOID
15F5  E8 1EE8 R                       CALL    DN1MS
15F8  E4 00             POR_A:        IN      AL,IOA          ;WAS THE KEY RELEASED?
15FA  3A 06 2029 R                    CMP     AL,[ALM]
15FE  74 F8                           JZ      POR_A           ;NO, TEST AGAIN.
1600  B9 0014                         MOV     CX,0014H        ;YES PROVENT FROM BOUNCING
1603  E8 1EE8 R                       CALL    DN1MS
1606  EB 0A 90                        JMP     RCL_A1          ;A INSTRUCTION INC DI HAS EXECUTED

1609  8B 3E 202D R      RCL_A:        MOV     DI,[KEY_NUMBER]
160D  47                              INC     DI
160E  89 3E 202D R                    MOV     [KEY_NUMBER],DI

1612  A0 2029 R         RCL_A1:       MOV     AL,[ALM]        ;ROTATE [ALM] FOR NEXT COLUMN
1615  B1 01                           MOV     CL,01H
1617  D2 C0                           ROL     AL,CL
1619  A2 2029 R                       MOV     [ALM],AL
```

```
161C  FE 06 2027 R                      INC     [COUTER_A]      ;COLUMN NUMBER IN [COUNTER_A]
1620  A0 2027 R                         MOV     AL,[COUTER_A]
1623  3C 08                             CMP     AL,08H          ;COLUMN NO. 8?
1625  75 B7                             JNZ     PORT_A          ;NO, SCANNING CONTINOUSLLY

1627  B0 00                             MOV     AL,00H          ;YES,
1629  A2 2027 R                         MOV     [COUTER_A],AL   ;CLEAR COLUMN NOMBER STORAGE

162C  A0 202A R                         MOV     AL,[BLM]        ;POART A SCANNING FINISHED
162F  B1 01                             MOV     CL,01H          ;TURN TO PORT B SCANNING
1631  D2 C0                             ROL     AL,CL
1633  A2 202A R                         MOV     [BLM],AL
1636  FE 06 2028 R                      INC     [COUTER_B]      ;LINE NUMBER INC ONE
163A  A0 2028 R                         MOV     AL,[COUTER_B]
163D  3C 05                             CMP     AL,05H          ;LINE 5 FINISHED ?
163F  75 96                             JNZ     PORT_B          ;NO, LINE SCANNING CONTINUOSLLY,

1641  E9 00C0 R                         JMP     K_80

                        KEY_SCANNING            ENDP
                        ;
                        ;
1644                    LOOK_UP_1       PROC    NEAR

1644  A1 202B R                         MOV     AX,[PRESS_T]    ;HOW MANY TIMES HAS PRESSED?
1647  40                                INC     AX
1648  A3 202B R                         MOV     [PRESS_T],AX
164B  3C 11                             CMP     AL,17           ;OUT OF LCD RANG?
164D  74 4F                             JZ      CLR_LXX         ;YES, CLEAR LCD

164F  BB D1F6                           MOV     BX,0D1F6H       ;POINT ASCII LIST. FIRST ADDRESS-1
1652  8B 3E 202D R                      MOV     DI,[KEY_NUMBER] ;KEY NUMBER
1656  8A 01                             MOV     AL,[BX][DI]     ;LOOK UP ASCII LIST TO FIND OUT THE
1658  3C 21                             CMP     AL,'!'          ;LETTER BEING PRESSED, LRTTER IN AL
165A  74 2A                             JZ      COMMAND         ;IT WAS F1
165C  3C 23                             CMP     AL,'#'
165E  74 38                             JZ      F_3             ;IT WAS F3
1660  3C 24                             CMP     AL,'$'
1662  74 2E                             JZ      F_2             ;IT WAS F2

1664  8D 1E 2012 R                      LEA     BX,KEY_DIS      ;POINT TO KEY DISPLAY BUFFER
1668  88 07                             MOV     [BX],AL         ;SEND LETTER TO DISPLAY BUFFER
166A  8D 1E 2013 R                      LEA     BX,STRING       ;STRING EQU 1CF1H
166E  A1 202B R                         MOV     AX,[PRESS_T]
1671  3C 01                             CMP     AL,01H          ;FIRST PRESS?
1673  74 06                             JZ      PRE             ;YES,

1675  48                                DEC     AX              ;NO FIRST PRESS

1676  8B C8                             MOV     CX,AX           ;POINT TO THE POSITION WHERE
1678  43                PPE:            INC     BX              ;THE LETTER SHOULD BE DISPLAYED
1679  E2 FD                             LOOP    PPE

167B  A0 2012 R         PRE:            MOV     AL,[KEY_DIS]    ;FIND THE LETTER IN DIS BUFFER
167E  88 07                             MOV     [BX],AL         ;SEND LETTER TO STRING AREA

1680  E8 1740 R                         CALL    DIS_KEY         ;PRESS_TIME AS POSITION OF DIS
1683  EB 1C 90                          JMP     LK_END

1686  E8 16A2 R         COMMAND:        CALL    STRING_C

1689  B8 0000                           MOV     AX,0000H        ;AFTER EXCUSE COMMAND CLEAR PRESS
168C  A3 202B R                         MOV     [PRESS_T],AX    ;TIME

168F  EB 10 90                          JMP     LK_END

1692  E8 00F4 R         F_2:            CALL    WORKING         ;INTERRUPT & COUNTER INICIALIZATION
```

```
1695  EB 0A 90                        JMP     LK_END

1698  E8 1BBF R          F_3:         CALL    END_TEST        ;F3 WAS THE COMMAND END TEST.
169B  EB 04 90                        JMP     LK_END

169E  E8 1A78 R          CLR_LXX:     CALL    CLEAR_LCD

16A1  C3                 LK_END:      RET

                         LOOK_UP_1    ENDP
                         ;
                         ;
16A2                     STRING_C     PROC    NEAR
                                      ;--------------------------------------------------------;
                                      ; COMMAND AND REAL TIME CLOCK PROGRAM ARE ALSO INCLUDED.  ;
                                      ; 1. CHECK PRESS KEY TIMES, IF 5(TYPE) 6(CLOCK) 10 CLOCK ;
                                      ;    SET 9(SET 11-11-88 & 12 37 00) 14(INTERVAL TIME)    ;
                                      ;    3(00 -- 99 MINUTSET)                                ;
                                      ; 2. IF THE COMMAND WAS VALID THEN CALL FUCTION SUBROUTINE;
                                      ;    IF IT WAS NOT VALID THEN CALL BAD COMMAND !         ;
                                      ;--------------------------------------------------------;
16A2  A1 202B R                       MOV     AX,[PRESS_T]
16A5  3D 0005                         CMP     AX,5
16A8  74 70                           JZ      TY_PE           ;TYPE THE LIST FILE
16AA  3D 0006                         CMP     AX,6
16AD  74 52                           JZ      CL_CK           ;CLOCK READ
16AF  3D 000A                         CMP     AX,10
16B2  74 34                           JZ      CL_SE           ;CLOCK SET
16B4  3D 0009                         CMP     AX,9
16B7  74 2C                           JZ      SET             ;SET YEAR MONTH --- SECOND
16B9  3D 000E                         CMP     AX,14
16BC  74 0E                           JZ      INTER           ;INTERVAL TIME SET
16BE  3D 0003                         CMP     AX,3            ;SET INTERVAL 00 --- 99 MINUTS
16C1  74 06                           JZ      INT_TT

16C3  E8 1AB2 R                       CALL    BAD_COMMAND     ;IT WAS NOT VALID PRESS TIMES
16C6  EB 77 90                        JMP     PPP2            ;JMP TO RET

16C9  E9 19C9 R          INT_TT:      JMP     INT_TIME        ;GO TO INTERVAL TIME SET

16CC  A1 202B R          INTER:       MOV     AX,[PRESS_T]    ;PRESS TIMES -1 BECAUSE
16CF  48                              DEC     AX              ;THERE WAS A RETURN KEY AT
16D0  8B C8                           MOV     CX,AX           ;THE END OF THE COMMAND
16D2  BF D241                         MOV     DI,0D241H       ;POINT TO COMMAND ASCII STRING
16D5  8D 36 2013 R                    LEA     SI,STRING       ;POINT TO STRING TO BE ENTERED
16D9  FC                              CLD                     ;STRING COMPAR
16DA  A6                 REP_4:       CMPS    BYTE PTR[DI],BYTE PTR[SI]
16DB  75 5F                           JNE     PPP1            ;INVALID THEN BAD COMMAND !
16DD  E2 FB                           LOOP    REP_4
16DF  E8 19C3 R                       CALL    IN_TE_T         ;VALID INTERVAL TIME SET
16E2  EB 5B 90                        JMP     PPP2            ;END INTERVAL TIME SET RET

16E5  E9 1924 R          SET:         JMP     C_SET           ;SET THE REAL TIME CLOCK

16E8  A1 202B R          CL_SE:       MOV     AX,[PRESS_T]    ;PRESS TIMES
16EB  48                              DEC     AX          ;-1 BECAUSE THERE WAS A RETURN KEY AT
16EC  8B C8                           MOV     CX,AX       ;THE END OF THE COMMAND
16EE  BF D234                         MOV     DI,0D234H   ;POINT TO COMMAND ASCII STRING
16F1  8D 36 2013 R                    LEA     SI,STRING   ;POINT TO STRING TO BE ENTERED
                                                          ;STRING COMPAR
16F5  FC                              CLD                 ;CLEAR DF FLAG DI SI ARE INCREASED
16F6  A6                 REP_1:       CMPS    BYTE PTR [DI],BYTE PTR [SI]
16F7  75 43                           JNE     PPP1        ;INVALID THEN BAD COMMAND !
16F9  E2 FB                           LOOP    REP_1
16FB  E8 191E R                       CALL    CLOCK_SET       ;VALID CLOCK SET
16FE  EB 3F 90                        JMP     PPP2            ;END CLOCK SET RET
```

```
1701  A1 202B R           CL_CK:      MOV    AX,[PRESS_T]       ;PRESS TIMES
1704  48                              DEC    AX          ;-1 BECAUSE THERE WAS A RETURN KEY AT
1705  8B C8                           MOV    CX,AX              ;THE END OF THE COMMAND
1707  BF D22F                         MOV    DI,0D22FH          ;POINT TO COMMAND ASCII STRING
170A  8D 36 2013 R                    LEA    SI,STRING          ;POINT TO STRING TO BE ENTERED
170E  FC                              CLD                       ;STRING COMPAR
170F  A6                  REP_2:      CMPS   BYTE PTR [DI],BYTE PTR [SI]
1710  75 2A                           JNE    PPP1               ;INVALID THEN BAD COMMAND !
1712  E2 FB                           LOOP   REP_2
1714  E8 183E R                       CALL   CLOCK_DIS          ;VALID CLOCK DISPLAY
1717  EB 26 90                        JMP    PPP2               ;END CLOCK DISPLAY RET

171A  A1 202B R           TY_PE:      MOV    AX,[PRESS_T]       ;PRESS TIMES
171D  48                              DEC    AX          ;-1 BECAUSE THERE WAS A RETURN KEY AT
171E  8B C8                           MOV    CX,AX              ;THE END OF THE COMMAND
1720  BF D23D                         MOV    DI,0D23DH          ;POINT TO COMMAND ASCII STRING
1723  8D 36 2013 R                    LEA    SI,STRING          ;POINT TO STRING TO BE ENTERED
1727  FC                              CLD                       ;STRING COMPAR
1728  A6                  REP_3:      CMPS   BYTE PTR [DI],BYTE PTR [SI]
1729  75 11                           JNE    PPP1               ;INVALID THEN BAD COMMAND !
172B  E2 FB                           LOOP   REP_3

172D  E8 17FB R                       CALL   RS_232             ;VALID RS 232 INCIALISATION
1730  BF 0400                         MOV    DI,0400H       ;THE BIGINNING ADDRESS OF LIST FILE
1733  B9 02C0                         MOV    CX,22*32           ;THE NUMBER OF BYTE TO BE PRINT
1736  E8 1818 R                       CALL   TEST_TX            ;FILE TRANSMISSION VIA RS 232
1739  EB 04 90                        JMP    PPP2

173C  E8 1AB2 R           PPP1:       CALL   BAD_COMMAND
173F  C3                  PPP2:       RET

                          STRING_C    ENDP
                          ;
                          ;
1740                      DIS_KEY     PROC   NEAR
                                      ;----------------------------------------------;
                                      ;DISPLAY THE LETTER WHICH WAS PRESSED AT KEYBOAD;
                                      ;----------------------------------------------;
1740  B0 80                           MOV    AL,80H
1742  E6 01                           OUT    IOB,AL
1744  B0 0F                           MOV    AL,0FH             ;TURN LCD ON
1746  E6 02                           OUT    IOC,AL
1748  B0 00                           MOV    AL,00H
174A  E6 01                           OUT    IOB,AL

174C  8D 1E 2012 R                    LEA    BX,KEY_DIS         ;POINT TO DISPLAY BUFFER
1750  8B 16 202B R                    MOV    DX,[PRESS_T]       ;SET POSITION POINT
                                                                ;POSITION=[PRESS_TIMES-1]
1754  89 16 2010 R                    MOV    [POINT_S1],DX      ;PROTECT POINT
1758  E8 1764 R                       CALL   PO_SI              ;PROCEDURE FOR SELECT LCD
                                                                ;ADDRESS
175B  8B 07                           MOV    AX,[BX]            ;THE LETTER TO BE DISPLAY NOW
                                                                ;IN AX
175D  E8 1A29 R                       CALL   DATA_D             ;DISPLAY THE LETTER
1760  E8 1A48 R                       CALL   CHECK              ;CHECK THE BUSY STATUS

1763  C3                              RET

                          DIS_KEY     ENDP
                          ;
                          ;
1764                      PO_SI       PROC   NEAR               ;LCD address selecting

1764  B0 90                           MOV    AL,CWIOO
1766  E6 03                           OUT    IOCTR,AL

1768  8B 16 2010 R        POS:        MOV    DX,[POINT_S1]      ;THE POSITION WHERE THE LETTER
```

```
176C  83 FA 01      POS_1:     CMP    DX,0001H       ; DISPLAIED
176F  75 16                    JNZ    DX_EQU8
1771  B0 00                    MOV    AL,00H         ;SET LCD ADDRESS 00H
1773  E6 01                    OUT    IOB,AL
1775  B0 80                    MOV    AL,80H
1777  E6 01                    OUT    IOB,AL
1779  B0 80                    MOV    AL,80H         ;SET ADDRESS DB7=1
177B  E6 02                    OUT    IOC,AL
177D  B0 00                    MOV    AL,00H
177F  E6 01                    OUT    IOB,AL
1781  E8 1A48 R                CALL   CHECK          ;CHECK HAND SHAKE SIGNAL
1784  EB 24 90                 JMP    POS_BACK

1787  83 FA 09      DX_EQU8:   CMP    DX,0009H
178A  75 16                    JNZ    DX_EQU16
178C  B0 00                    MOV    AL,00H         ;SET LCD ADDRESS 40H
178E  E6 01                    OUT    IOB,AL
1790  B0 80                    MOV    AL,80H
1792  E6 01                    OUT    IOB,AL
1794  B0 C0                    MOV    AL,0C0H        ;SET ADDRESS INSTRUCTION DB7=1
1796  E6 02                    OUT    IOC,AL
1798  B0 00                    MOV    AL,00H
179A  E6 01                    OUT    IOB,AL
179C  E8 1A48 R                CALL   CHECK          ;CHECK HAND SHAKE SIGNAL
179F  EB 09 90                 JMP    POS_BACK

17A2  83 FA 10      DX_EQU16:  CMP    DX,0016
17A5  75 03                    JNZ    POS_BACK
17A7  BA 0000                  MOV    DX,0000H

17AA  C3            POS_BACK:  RET

                    PO_SI      ENDP
                    ;
                    ;
17AB                POSITION   PROC   NEAR           ;LCD ADDRESS SELECTION

17AB  B0 90                    MOV    AL,CWIOO
17AD  E6 03                    OUT    IOCTR,AL

17AF  8B 16 200E R             MOV    DX,[POINT_S]   ;RESUME POINT
17B3  83 FA 00      POSI_1:    CMP    DX,0000H
17B6  75 16                    JNZ    DX_EQU7
17B8  B0 00                    MOV    AL,00H         ;SET LCD ADDRESS 00H
17BA  E6 01                    OUT    IOB,AL
17BC  B0 80                    MOV    AL,80H
17BE  E6 01                    OUT    IOB,AL
17C0  B0 80                    MOV    AL,80H         ;SET ADDRESS INSTRUCTION DB7=1
17C2  E6 02                    OUT    IOC,AL
17C4  B0 00                    MOV    AL,00H
17C6  E6 01                    OUT    IOB,AL
17C8  E8 1A48 R                CALL   CHECK          ;CHECK HAND SHAKE SIGNALS
17CB  EB 28 90                 JMP    DX_INC

17CE  83 FA 08      DX_EQU7:   CMP    DX,0008H
17D1  75 18                    JNZ    DX_EQU47
17D3  B0 00                    MOV    AL,00H         ;SET LCD ADDRESS 40H
17D5  E6 01                    OUT    IOB,AL
17D7  B0 80                    MOV    AL,80H
17D9  E6 01                    OUT    IOB,AL
17DB  B0 C0                    MOV    AL,0C0H        ;SET ADDRESS INSTRUCTION DB7=1
17DD  E6 02                    OUT    IOC,AL
17DF  B2 40                    MOV    DL,40H
17E1  B0 00                    MOV    AL,00H
17E3  E6 01                    OUT    IOB,AL
17E5  E8 1A48 R                CALL   CHECK          ;CHECK HAND SHAKE SIGNAL
17E8  EB 0B 90                 JMP    DX_INC
```

```
17EB  83 FA 48          DX_EQU47:   CMP     DX,0048H
17EE  75 05                         JNZ     DX_INC
17F0  BA 0000                       MOV     DX,0000H

17F3  EB BE                         JMP     POSI_1

17F5  42                DX_INC:     INC     DX
17F6  89 16 200E R                  MOV     [POINT_S],DX

17FA  C3                PRD_BACK:   RET

                        POSITION    ENDP
                        ;
                        ;
17FB                    RS_232      PROC    NEAR
                                    ;-----------------------------------------------;
                                    ; 1. 8253 TIMER COUNTER AS BAUD RATE GENERATER. ;
                                    ; 2. THE PROGRAMMBEL UART 8251 AS RS-232        ;
                                    ;-----------------------------------------------;
                                    ;PART 1: INICIALIZATION OF TIMER UPD8253

17FB  B0 36                         MOV     AL,00110110B     ;CONTROL WORDS
17FD  E6 23                         OUT     23H,AL
17FF  B0 04                         MOV     AL,04H           ;DATA WILL BE SEND INTO COUNTER
1801  E6 20                         OUT     20H,AL           ;LSB=04H FIRST, MSB=01H SECOND.
1803  B0 01                         MOV     AL,01H
1805  E6 20                         OUT     20H,AL

                                    ;PART 2: UPD8251 INICIALIZATION
1807  BA 0011                       MOV     DX,UART_CTRL     ;RESET
180A  B0 00                         MOV     AL,00H
180C  EE                            OUT     DX,AL
180D  EE                            OUT     DX,AL
180E  EE                            OUT     DX,AL
180F  B0 50                         MOV     AL,01010000B ;COMMAND:INTERNAL RESET,
1811  EE                            OUT     DX,AL        ;CLEAR ERROR BITS.
1812  B0 CE                         MOV     AL,11001110B ;MODE:2 STOP BIT, NO PARRITY BIT
                                                         ;8 DATA BITS, BAUD RATE FACTER=16
1814  EE                            OUT     DX,AL
1815  B0 33                         MOV     AL,00110011B ;COMMAND: RTS~ ACTIV, CLEAR ERROR
                                                         ;DTR~ OUTPUT LOW, TRANSMITTER WAS
                                                         ;ENABLED.
1817  C3                            RET

                        RS_232      ENDP
                        ;
                        ;
1818                    TEST_TX     PROC    NEAR
                                    ;-----------------------------------------------------;
                                    ; 1. THE BEGINNING ADDRSS OF THE PRINT FILE WAS IN DI;
                                    ; 2. THE NUMBER OF BYTES TO BE PRINTED WAS IN CX      ;
                                    ;-----------------------------------------------------;
1818  B0 9B                         MOV     AL,CWIII
181A  E6 53                         OUT     IOCTR_2,AL   ;IOB_2 SERVE AS HANDSHAKING INPUT
181C  51                BEGIN:      PUSH    CX
181D  B0 33                         MOV     AL,00110011B
181F  E6 11                         OUT     UART_CTRL,AL
1821  8A 05                         MOV     AL,[DI]
1823  E6 10                         OUT     UART_DATA,AL

1825  E4 51             RTS:        IN      AL,IOB_2     ;TEST THE BUZY STATUE, IF IT WAS 0
1827  24 01                         AND     AL,01H       ;CONTINUE TO TEST.
1829  3C 01                         CMP     AL,01H
182B  75 F8                         JNZ     RTS

182D  E4 51             RETEST:     IN      AL,IOB_2     ;TEST THE BUSY STATUS
```

```
182F  24 01                    AND     AL,01H        ;MASK REST OF THE BITS
1831  3C 01                    CMP     AL,01H
1833  74 F8                    JZ      RETEST

1835  47                       INC     DI            ;POINT TO NEXT LETTER
1836  59                       POP     CX
1837  E2 E3                    LOOP    BEGIN

1839  B0 99                    MOV     AL,CWIOI
183B  E6 53                    OUT     IOCTR_2,AL    ;IOB_2 SERVE AS COUNTERS & TIMER
                                                     ;CONTROL AFTER TYPING.
183D  C3                       RET

                TEST_TX        ENDP
                ;
                ;
183E            CLOCK_DIS PROC         NEAR

183E  B9 0004                  MOV     CX,04H        ;DISPLAY CLOCK FOR FOUR TIMES
1841  51        RT:            PUSH    CX
1842  B0 40                   MOV     AL,01000000B  ;WRITE CONTROL WORDS FOR REAL TIME CLOCK

1844  BB 27F8                  MOV     BX,RTCTRL
1847  88 07                    MOV     [BX],AL

1849  8D 1E 202F R             LEA     BX,YEAR_RAM   ;1D0DH MOVE DATA IN RTCLOCK TO RAM
184D  BF 27FF                  MOV     DI,YEAR       ;27FFH
1850  8A 05                    MOV     AL,[DI]
1852  E8 1A95 R                CALL    BCD_ASCII_CL
1855  88 67 08                 MOV     [BX]+8,AH
1858  88 47 09                 MOV     [BX]+9,AL

185B  BF 27FE                  MOV     DI,MONTH      ;27FEH
185E  8A 05                    MOV     AL,[DI]
1860  E8 1A95 R                CALL    BCD_ASCII_CL
1863  88 27                    MOV     [BX],AH
1865  88 47 01                 MOV     [BX]+1,AL

1868  BF 27FD                  MOV     DI,DATE       ;27FDH
186B  8A 05                    MOV     AL,[DI]
186D  E8 1A95 R                CALL    BCD_ASCII_CL
1870  88 67 03                 MOV     [BX]+3,AH
1873  88 47 04                 MOV     [BX]+4,AL

                               ;MOV    DI,DAY        ;27FCH
                               ;MOV    AL,[DI]
                               ;CALL   BCD_ASCII_CL
                               ;MOV    [BX]+12,AH
                               ;MOV    [BX]+13,AL

1876  8D 1E 203F R             LEA     BX,HOUR_RAM   ;1D1DH

187A  BF 27FB                  MOV     DI,HR         ;27FBH
187D  8A 05                    MOV     AL,[DI]
187F  E8 1A95 R                CALL    BCD_ASCII_CL
1882  88 67 04                 MOV     [BX]+4,AH
1885  88 47 05                 MOV     [BX]+5,AL

1888  BF 27FA                  MOV     DI,MIN        ;27FAH
188B  8A 05                    MOV     AL,[DI]
188D  E8 1A95 R                CALL    BCD_ASCII_CL
1890  88 67 07                 MOV     [BX]+7,AH
1893  88 47 08                 MOV     [BX]+8,AL

1896  BF 27F9                  MOV     DI,SEC        ;27F9H
1899  8A 05                    MOV     AL,[DI]
189B  E8 1A95 R                CALL    BCD_ASCII_CL
189E  88 67 0A                 MOV     [BX]+10,AH
```

```
18A1  88 47 0B                    MOV     [BX]+11,AL

18A4  B0 80            C00:       MOV     AL,80H
18A6  E6 01                       OUT     IOB,AL
18A8  B0 0C                       MOV     AL,0CH
18AA  E6 02                       OUT     IOC,AL
18AC  B0 00                       MOV     AL,00H
18AE  E6 01                       OUT     IOB,AL

18B0  8D 1E 202F R     C0:        LEA     BX,YEAR_RAM      ;DIS YEAR--DAY
18B4  BA 0000                     MOV     DX,POINT
18B7  89 16 200E R                MOV     [POINT_S],DX

18BB  E8 17AB R        C1:        CALL    POSITION
18BE  8B 07                       MOV     AX,[BX]
18C0  E8 1A29 R                   CALL    DATA_D
18C3  E8 1A48 R                   CALL    CHECK
18C6  43                          INC     BX
18C7  B0 10                       MOV     AL,10H
18C9  FE 06 2008 R                INC     [FLAG1]
18CD  3A 06 2008 R                CMP     AL,[FLAG1]
18D1  75 E8                       JNZ     C1
18D3  B9 03E8                     MOV     CX,1000
18D6  E8 1EE8 R                   CALL    DN1MS

18D9  B0 00                       MOV     AL,00H           ;DIS HOUR--SECND
18DB  A2 2008 R                   MOV     [FLAG1],AL
18DE  BA 0000                     MOV     DX,POINT
18E1  89 16 200E R                MOV     [POINT_S],DX

18E5  8D 1E 203F R                LEA     BX,HOUR_RAM
18E9  E8 17AB R        C2:        CALL    POSITION
18EC  8B 07                       MOV     AX,[BX]
18EE  E8 1A29 R                   CALL    DATA_D
18F1  E8 1A48 R                   CALL    CHECK
18F4  43                          INC     BX
18F5  B0 10                       MOV     AL,10H
18F7  FE 06 2008 R                INC     [FLAG1]
18FB  3A 06 2008 R                CMP     AL,[FLAG1]
18FF  75 E8                       JNZ     C2

1901  B0 00                       MOV     AL,00H
1903  A2 2008 R                   MOV     [FLAG1],AL

1906  B0 00                       MOV     AL,00000000B     ;START CLOCK
1908  BB 27F8                     MOV     BX,RTCTRL
190B  88 07                       MOV     [BX],AL

190D  B9 03E8                     MOV     CX,1000
1910  E8 1EE8 R                   CALL    DN1MS

1913  59                          POP     CX               ;DIS TIME FOR 4 TIMES
1914  49                          DEC     CX
1915  74 03                       JZ      CED
1917  E9 1841 R                   JMP     RT

191A  E8 1A78 R        CED:       CALL    CLEAR_LCD
191D  C3                          RET

                       CLOCK_DIS ENDP
                       ;
                       ;
191E                   CLOCK_SET PROC     NEAR

191E  E8 1A78 R                   CALL    CLEAR_LCD
1921  E9 15BD R                   JMP     COUN             ;TO KEY SCANNING
```

```
1924  B0 FF              C_SET:    MOV     AL,0FFH
1926  3A 06 200C R                 CMP     AL,[FLAG6]
192A  74 4D                        JZ      C_SET1          ;IF FLAG6=FFH YEAR HAS BEEN SET

192C  B0 80                        MOV     AL,10000000B    ;SET CLOCK
192E  BB 27F8                      MOV     BX,RTCTRL       ;REAL TIME CLOCK CONTROL WORDS
1931  88 07                        MOV     [BX],AL         ;REGISTER

1933  8D 3E 2013 R                 LEA     DI,STRING
1937  8A 25                        MOV     AH,[DI]
1939  8A 45 01                     MOV     AL,[DI]+1

                         ;         MOV     BX,X_MONTH      ;TO TYPE 0551H
                         ;         MOV     [BX],AH
                         ;         MOV     [BX]+1,AL

193C  E8 1AA6 R                    CALL    ASCII_BCD_CL

193F  BB 27FE                      MOV     BX,MONTH
1942  88 07                        MOV     [BX],AL

1944  8A 65 03                     MOV     AH,[DI]+3
1947  8A 45 04                     MOV     AL,[DI]+4

                         ;         MOV     BX,X_DATE       ;TO TYPE 0546H
                         ;         MOV     [BX],AH
                         ;         MOV     [BX]+1,AL

194A  E8 1AA6 R                    CALL    ASCII_BCD_CL

194D  BB 27FD                      MOV     BX,DATE
1950  88 07                        MOV     [BX],AL

1952  8A 65 06                     MOV     AH,[DI]+6
1955  8A 45 07                     MOV     AL,[DI]+7

                         ;         MOV     BX,X_YEAR       ;TO TIPE 055AH
                         ;         MOV     [BX],AH
                         ;         MOV     [BX]+1,AL

1958  E8 1AA6 R                    CALL    ASCII_BCD_CL

195B  BB 27FF                      MOV     BX,YEAR
195E  88 07                        MOV     [BX],AL

1960  B0 00                        MOV     AL,00H        ;SET FT=0 (FT BIT WAS AT 7th IN DAY)
1962  BB 27FC                      MOV     BX,DAY
1965  88 07                        MOV     [BX],AL

1967  B0 FF                        MOV     AL,0FFH
1969  A2 200C R                    MOV     [FLAG6],AL      ;IF 00H SET YEAR, IF 0FFH SET HOUR

196C  B0 00                        MOV     AL,00000000B    ;START CLOCK
196E  BB 27F8                      MOV     BX,RTCTRL
1971  88 07                        MOV     [BX],AL

1973  E8 1A78 R                    CALL    CLEAR_LCD
1976  E9 15BD R                    JMP     COUN            ;GO BACK TO KEY SCANNING FIND HOUR

1979  B0 80              C_SET1:   MOV     AL,10000000B    ;SET CLOCK
197B  BB 27F8                      MOV     BX,RTCTRL
197E  88 07                        MOV     [BX],AL

1980  8D 3E 2013 R                 LEA     DI,STRING
1984  8A 25                        MOV     AH,[DI]
1986  8A 45 01                     MOV     AL,[DI]+1
```

```
                        ;           MOV     BX,X_TIMEE        ;START TIME 058DH
                        ;           MOV     [BX],AH
                        ;           MOV     [BX]+1,AL

 1989  E8 1AA6 R                    CALL    ASCII_BCD_CL

 198C  BB 27FB                      MOV     BX,HR
 198F  88 07                        MOV     [BX],AL

 1991  8A 65 03                     MOV     AH,[DI]+3
 1994  8A 45 04                     MOV     AL,[DI]+4

                        ;           MOV     BX,X_TIMEE+3      ;0590H
                        ;           MOV     [BX],AH
                        ;           MOV     [BX]+1,AL

 1997  E8 1AA6 R                    CALL    ASCII_BCD_CL

 199A  BB 27FA                      MOV     BX,MIN
 199D  88 07                        MOV     [BX],AL

 199F  8D 1E 2050 R                 LEA     BX,T_RAM1
 19A3  88 07                        MOV     [BX],AL            ;STORE OLD TIME

 19A5  8A 65 06                     MOV     AH,[DI]+6
 19A8  8A 45 07                     MOV     AL,[DI]+7

                        ;           MOV     BX,X_TIMEE+6       ;0593H
                        ;           MOV     [BX],AH
                        ;           MOV     [BX]+1,AL

 19AB  E8 1AA6 R                    CALL    ASCII_BCD_CL

 19AE  BB 27F9                      MOV     BX,SEC
 19B1  88 07                        MOV     [BX],AL

 19B3  B0 00                        MOV     AL,00H
 19B5  A2 200C R                    MOV     [FLAG6],AL

 19B8  B0 00                        MOV     AL,00000000B       ;START CLOCK
 19BA  BB 27F8                      MOV     BX,RTCTRL
 19BD  88 07                        MOV     [BX],AL

 19BF  E8 1A78 R                    CALL    CLEAR_LCD
 19C2  C3                           RET

                        CLOCK_SET ENDP
                        ;
                        ;
 19C3                   IN_TE_T     PROC    NEAR

 19C3  E8 1A78 R                    CALL    CLEAR_LCD
 19C6  E9 15BD R                    JMP     COUN              ;TO KEY SCANNING

 19C9  8D 3E 2013 R     INT_TIME:   LEA     DI,STRING         ;SENT 01--60 MINUTS TO RAM
 19CD  8A 25                        MOV     AH,[DI]
 19CF  8A 45 01                     MOV     AL,[DI]+1
 19D2  E8 1AA6 R                    CALL    ASCII_BCD_CL
 19D5  8D 1E 204F R                 LEA     BX,I_T_V          ;STOER INTERVAL
 19D9  88 07                        MOV     [BX],AL
 19DB  E8 1A78 R                    CALL    CLEAR_LCD
 19DE  B0 FF                        MOV     AL,0FFH           ;IF INTERVAL TIME HAS BEEN SET
 19E0  A2 200D R                    MOV     [FLAG7],AL        ;SET THE FLAG7

 19E3  C3                           RET

                        IN_TE_T     ENDP
```

```
                                ;
                                ;
19E4                            TIME_CHECK    PROC    NEAR
19E4  B0 40                          MOV     AL,01000000B     ;WRITE CONTROL WORDS FOR REAL TIME CLOCK

19E6  BB 27F8                        MOV     BX,RTCTRL
19E9  88 07                          MOV     [BX],AL

19EB  8D 1E 203F R                   LEA     BX,HOUR_RAM      ;1D1DH
19EF  BF 27FB                        MOV     DI,HR            ;27FBH
19F2  8A 05                          MOV     AL,[DI]
19F4  E8 1A95 R                      CALL    BCD_ASCII_CL
19F7  88 67 04                       MOV     [BX]+4,AH
19FA  88 47 05                       MOV     [BX]+5,AL

19FD  BF 27FA                        MOV     DI,MIN           ;27FAH
1A00  8A 05                          MOV     AL,[DI]
1A02  53                             PUSH    BX
1A03  8D 1E 2051 R                   LEA     BX,T_RAM2
1A07  88 07                          MOV     [BX],AL          ;CURRENT TIME-->[T_RAM2]
1A09  E8 1A95 R                      CALL    BCD_ASCII_CL
1A0C  5B                             POP     BX
1A0D  88 67 07                       MOV     [BX]+7,AH
1A10  88 47 08                       MOV     [BX]+8,AL

1A13  BF 27F9                        MOV     DI,SEC           ;27F9H
1A16  8A 05                          MOV     AL,[DI]
1A18  E8 1A95 R                      CALL    BCD_ASCII_CL
1A1B  88 67 0A                       MOV     [BX]+10,AH
1A1E  88 47 0B                       MOV     [BX]+11,AL

1A21  B0 00                          MOV     AL,00000000B
1A23  BB 27F8                        MOV     BX,RTCTRL
1A26  88 07                          MOV     [BX],AL

1A28  C3                             RET

                                TIME_CHECK      ENDP
                                ;
                                ;
1A29                            DATA_DIS  PROC    NEAR        ;THIS WAS DATA DISPLAY PROCEDURE.

1A29  50                        DATA_D:   PUSH    AX          ;PROTECT DATA STORED IN AL.
1A2A  B0 90                               MOV     AL,CWIOO    ;INITI PIO as IOO.
1A2C  E6 03                               OUT     IOCTR,AL

1A2E  B0 00                               MOV     AL,00H
1A30  E6 01                               OUT     IOB,AL
1A32  90                                  NOP

1A33  B0 20                               MOV     AL,20H      ;WHEN WRITE DATA TO DATA RAM RS=1
1A35  E6 01                               OUT     IOB,AL
1A37  90                                  NOP
1A38  B0 A0                               MOV     AL,0A0H     ;WRITE DATA RS=1 (PB5,I[illegible]
1A3A  E6 01                               OUT     IOB,AL
1A3C  58                                  POP     AX
1A3D  E6 02                               OUT     IOC,AL      ;DISPLAY THE DATA STORED IN AL.
1A3F  B0 20                               MOV     AL,20H
1A41  E6 01                               OUT     IOB,AL
1A43  B0 00                               MOV     AL,00H
1A45  E6 01                               OUT     IOB,AL

1A47  C3                                  RET

                                DATA_DIS  ENDP
                                ;
```

```
1A48                          CHECK     PROC      NEAR         ;THIS WAS THE LCD BUSY STATUS CHECK
                                                               ;PROCEDURE.

1A48  B0 99                             MOV       AL,CWIOI
1A4A  E6 03                             OUT       IOCTR,AL

1A4C  B0 40                   CHE:      MOV       AL,40H
1A4E  E6 01                             OUT       IOB,AL
1A50  90                                NOP
1A51  B0 C0                             MOV       AL,0C0H
1A53  E6 01                             OUT       IOB,AL
1A55  E4 02                             IN        AL,IOC            ;CHECK BUSY STATUS.
1A57  E4 02                             IN        AL,IOC
1A59  24 80                             AND       AL,80H
1A5B  75 EF                             JNZ       CHE
1A5D  E4 02                             IN        AL,IOC
1A5F  24 80                             AND       AL,80H
1A61  75 E9                             JNZ       CHE
1A63  E4 02                             IN        AL,IOC
1A65  E4 02                             IN        AL,IOC
1A67  24 80                             AND       AL,80H
1A69  75 E1                             JNZ       CHE

1A6B  B0 40                             MOV       AL,40H
1A6D  E6 01                             OUT       IOB,AL
1A6F  B0 00                             MOV       AL,00H
1A71  E6 01                             OUT       IOB,AL

1A73  B0 90                             MOV       AL,CWIOO
1A75  E6 03                             OUT       IOCTR,AL

1A77  C3                                RET

                              CHECK     ENDP
                              ;
                              ;
1A78                          CLEAR_LCD PROC      NEAR

1A78  B0 00                             MOV       AL,00H            ;CLEAR LCD
1A7A  E6 01                             OUT       IOB,AL
1A7C  B0 80                             MOV       AL,80H
1A7E  E6 01                             OUT       IOB,AL
1A80  B0 01                             MOV       AL,01H
1A82  E6 02                             OUT       IOC,AL
1A84  B0 00                             MOV       AL,00H
1A86  E6 01                             OUT       IOB,AL

1A88  B9 00A0                           MOV       CX,160
1A8B  E8 1EE8 R                         CALL      DN1MS

1A8E  B8 0000                           MOV       AX,0000H
1A91  A3 202B R                         MOV       [PRESS_T],AX      ;CLEAR PRESS TIMES

1A94  C3                                RET

                              CLEAR_LCD ENDP
                              ;
                              ;
1A95                          BCD_ASCII_CL PROC      NEAR

1A95  B4 00                             MOV       AH,00H
1A97  B1 04                             MOV       CL,04H
1A99  8A D0                             MOV       DL,AL
1A9B  D2 CA                             ROR       DL,CL
1A9D  0A E2                             OR        AH,DL
1A9F  25 0F0F                           AND       AX,0F0FH
1AA2  0D 3030                           OR        AX,3030H
```

```
1AA5  C3                            RET
                         BCD_ASCII_CL ENDP
                         ;
                         ;
1AA6                     ASCII_BCD_CL PROC      NEAR

1AA6  25 0F0F                      AND       AX,0F0FH
1AA9  8A D4                        MOV       DL,AH
1AAB  B1 04                        MOV       CL,04H
1AAD  D2 C2                        ROL       DL,CL
1AAF  0A C2                        OR        AL,DL

1AB1  C3                           RET

                         ASCII_BCD_CL ENDP
                         ;
                         ;
1AB2                     BAD_COMMAND PROC     NEAR
                                   ;-----------------------------------------------;
                                   ;  IF THE COMMAND ENTERED WAS WRONG, BAD COMMAND! ;
                                   ;  WOULD BE DISPLAYED ON THE LCD                  ;
                                   ;-----------------------------------------------;
1AB2  B0 80              B00:      MOV       AL,80H
1AB4  E6 01                        OUT       IOB,AL
1AB6  B0 0C                        MOV       AL,0CH
1AB8  E6 02                        OUT       IOC,AL
1ABA  B0 00                        MOV       AL,00H
1ABC  E6 01                        OUT       IOB,AL

1ABE  BB D21F            B0:       MOV       BX,0D21FH
1AC1  BA 0000            DIS_DIS:  MOV       DX,POINT
1AC4  89 16 200E R                 MOV       [POINT_S],DX

1AC8  E8 17AB R          B1:       CALL      POSITION
1ACB  8B 07                        MOV       AX,[BX]
1ACD  E8 1A29 R                    CALL      DATA_D
1AD0  E8 1A48 R                    CALL      CHECK
1AD3  43                           INC       BX
1AD4  B0 10                        MOV       AL,10H
1AD6  FE 06 2008 R                 INC       [FLAG1]
1ADA  3A 06 2008 R                 CMP       AL,[FLAG1]
1ADE  75 E8                        JNZ       B1
1AE0  B9 03E8                      MOV       CX,1000
1AE3  E8 1EE8 R                    CALL      DN1MS
1AE6  E8 1A78 R                    CALL      CLEAR_LCD
1AE9  B0 00                        MOV       AL,00H
1AEB  A2 2008 R                    MOV       [FLAG1],AL

1AEE  C3                           RET

                         BAD_COMMAND ENDP
                         ;
                         ;
1AEF                     DIS_TITLE     PROC      NEAR          ;DISPLAY TITLE & FORMS.

1AEF  B0 80                            MOV       AL,80H
1AF1  E6 01                            OUT       IOB,AL
1AF3  B0 0C                            MOV       AL,0CH        ;TURN LCD ON
1AF5  E6 02                            OUT       IOC,AL
1AF7  B0 00                            MOV       AL,00H
1AF9  E6 01                            OUT       IOB,AL

1AFB  BB D186            D0:           MOV       BX,0D186H     ;POINT TO TITLE
1AFE  BA 0000                          MOV       DX,POINT      ;SET POSITION POINT
1B01  89 16 200E R                     MOV       [POINT_S],DX  ;PROTECT POINT
```

```
1B05  E8 17AB R          D1:         CALL    POSITION       ;SELECT LCD POSITION
1B08  8B 07                          MOV     AX,[BX]        ;LETTER TO BE DISPLAIED IN AX
1B0A  E8 1A29 R                      CALL    DATA_D
1B0D  E8 1A48 R                      CALL    CHECK
1B10  43                 DD2:        INC     BX             ;DISPLAY NEXT LETTER OF THE TITLE
1B11  B0 10                          MOV     AL,10H
1B13  FE 06 2008 R                   INC     [FLAG1]
1B17  3A 06 2008 R                   CMP     AL,[FLAG1]     ;DIS FULL?
1B1B  75 0B                          JNZ     D2             ;NO, GO ON.
1B1D  B9 03E8                        MOV     CX,1000
1B20  E8 1EE8 R                      CALL    DN1MS          ;YES DELAY 1 SECOND
1B23  B0 00                          MOV     AL,00H
1B25  A2 2008 R                      MOV     [FLAG1],AL     ;CLEAR POSITION COUNTER

1B28  BF D1F6            D2:         MOV     DI,0D1F6H
1B2B  3B DF                          CMP     BX,DI          ;TITLE FINISHED?
1B2D  75 D6                          JNZ     D1             ;NO
1B2F  E8 1A78 R                      CALL    CLEAR_LCD      ;YES

1B32  C3                             RET

                         DIS_TITLE   ENDP
                         ;
                         ;
1B33                     LCD_INI     PROC    NEAR

1B33  B0 00                          MOV     AL,00H
1B35  E6 01                          OUT     IOB,AL
1B37  B0 80                          MOV     AL,80H
1B39  E6 01                          OUT     IOB,AL
1B3B  B0 38                          MOV     AL,38H         ;THE LCD INITIALIZATION.
1B3D  E6 02                          OUT     IOC,AL         ;8 BIT
1B3F  B0 00                          MOV     AL,00H
1B41  E6 01                          OUT     IOB,AL
1B43  B9 000A                        MOV     CX,10          ;10 ms
1B46  E8 1EE8 R                      CALL    DN1MS

1B49  B0 80                          MOV     AL,80H
1B4B  E6 01                          OUT     IOB,AL
1B4D  B0 38                          MOV     AL,38H         ;SAME AS ABOUVE
1B4F  E6 02                          OUT     IOC,AL
1B51  B0 00                          MOV     AL,00H
1B53  E6 01                          OUT     IOB,AL
1B55  B9 0001                        MOV     CX,1           ;1ms
1B58  E8 1EE8 R                      CALL    DN1MS

1B5B  B0 80                          MOV     AL,80H
1B5D  E6 01                          OUT     IOB,AL
1B5F  B0 38                          MOV     AL,38H         ;REPEAT ABOUVE INSTRUCTION AGAIN
1B61  E6 02                          OUT     IOC,AL
1B63  B0 00                          MOV     AL,00H
1B65  E6 01                          OUT     IOB,AL

1B67  B0 80                          MOV     AL,80H
1B69  E6 01                          OUT     IOB,AL
1B6B  B0 38                          MOV     AL,38H         ;REAPEAT
1B6D  E6 02                          OUT     IOC,AL
1B6F  B0 00                          MOV     AL,00H
1B71  E6 01                          OUT     IOB,AL

1B73  B0 80                          MOV     AL,80H
1B75  E6 01                          OUT     IOB,AL
1B77  B0 01                          MOV     AL,01H         ;DIS CLEAR
1B79  E6 02                          OUT     IOC,AL
1B7B  B0 00                          MOV     AL,00H
1B7D  E6 01                          OUT     IOB,AL
```

```
1B7F  B0 80                         MOV     AL,80H
1B81  E6 01                         OUT     IOB,AL
1B83  B0 0F                         MOV     AL,0FH          ;ACT DIS,CURSOR BLINK LETTER
1B85  E6 02                         OUT     IOC,AL
1B87  B0 00                         MOV     AL,00H
1B89  E6 01                         OUT     IOB,AL

1B8B  B0 80                         MOV     AL,80H
1B8D  E6 01                         OUT     IOB,AL
1B8F  B0 06                         MOV     AL,06H          ;MODE SET INCREASE NO SHIFT
1B91  E6 02                         OUT     IOC,AL
1B93  B0 00                         MOV     AL,00H
1B95  E6 01                         OUT     IOB,AL

1B97  B9 03E8                       MOV     CX,1000         ;TO SEE LCD HAVING BEEN CLEARED
1B9A  E8 1EE8 R                     CALL    DN1MS           ;FOR 1 SECOND

1B9D  C3                            RET

                      LCD_INI       ENDP
                      ;
                      ;
1B9E                  BEEP_N        PROC    NEAR
                                    ;-------------------------------------;
                                    ;BEEP N TIMES FOR TROUBLE SHOUTING.   ;
                                    ;N MUST BE PUT IN CX BEFOR CALL       ;
                                    ;CAUTION! AFTER CALLING MODE IS 'III';
                                    ;-------------------------------------;
1B9E  B0 99                         MOV     AL,CWIOI
1BA0  E6 53                         OUT     IOCTR_2,AL
1BA2  51              NHVKJH:       PUSH    CX
1BA3  B0 00                         MOV     AL,00000000B
1BA5  E6 51                         OUT     IOB_2,AL
1BA7  B9 0032                       MOV     CX,50
1BAA  E8 1EE8 R                     CALL    DN1MS
1BAD  B0 04                         MOV     AL,00000100B    ;PB2 AS OUTPUT PORT FOR BEEP
1BAF  E6 51                         OUT     IOB_2,AL
1BB1  B9 0032                       MOV     CX,50
1BB4  E8 1EE8 R                     CALL    DN1MS
1BB7  59                            POP     CX
1BB8  E2 E8                         LOOP    NHVKJH
1BBA  B0 9B                         MOV     AL,CWIII
1BBC  E6 53                         OUT     IOCTR_2,AL

1BBE  C3                            RET

                      BEEP_N        ENDP
                      ;
                      ;
1BBF                  END_TEST      PROC    NEAR
                                    ;---------------------------------------------;
                                    ;COMMAND SUBROUTINE                           ;
                                    ; 1. UNABLE INTERRUPT                         ;
                                    ; 2. CLOSE THE TIMER AND THE COUNTERS GATES   ;
                                    ; 3. PRINT OUT THE LIST FILE.                 ;
                                    ; 4. LOAD FLAG2 WITH 0                        ;
                                    ;---------------------------------------------;
1BBF  FA                            CLI
1BC0  B0 99                         MOV     AL,CWIOI
1BC2  E6 53                         OUT     IOCTR_2,AL
1BC4  B0 04                         MOV     AL,00000100B  ;CLOSE THE TIMER AND
1BC6  E6 51                         OUT     IOB_2,AL      ;COUNTERS GATES
1BC8  B0 70                         MOV     AL,01110000B  ;SET OUT(INTERRUPT REQUEST) TO 0
1BCA  E6 23                         OUT     TIMER_M,AL    ;MODE 0 INTERRUPT ON TERMINAL COUNT

1BCC  8D 1E 2072 R                  LEA     BX,DATA_DISPLAY ;POINT TO DATA DISPLAY UNIT
```

```
 1BD0  B8 4554                     MOV    AX,'ET'
 1BD3  89 07                       MOV    [BX],AX
 1BD5  B8 5453                     MOV    AX,'TS'
 1BD8  89 47 02                    MOV    [BX+2],AX
 1BDB  B8 2020                     MOV    AX,'  '
 1BDE  89 47 04                    MOV    [BX+4],AX
 1BE1  B8 4520                     MOV    AX, 'E '
 1BE4  89 47 06                    MOV    [BX+6],AX
 1BE7  B8 444E                     MOV    AX,'DN'
 1BEA  89 47 08                    MOV    [BX+8],AX
 1BED  B8 2020                     MOV    AX,'  '
 1BF0  89 47 0A                    MOV    [BX+10],AX
 1BF3  89 47 0C                    MOV    [BX+12],AX
 1BF6  89 47 0E                    MOV    [BX+14],AX
 1BF9  E8 1EBC R                   CALL   DISPLAY_1      ;DIS 'TEST   END' WITHOUT 1S DELAY.

 1BFC  E8 17FB R                   CALL   RS_232         ;VALID RS 232 INCIALISATION
 1BFF  BF 0400                     MOV    DI,0400H       ;THE BIGINNING ADDRESS OF LIST FILE
 1C02  B9 02C0                     MOV    CX,22*32       ;THE NUMBER OF BYTE TO BE PRINT
 1C05  8D 36 2004 R                LEA    SI,SAMPLE_No_1 ;POINT TO SAMPLE NUMBER
 1C09  8B C1                       MOV    AX,CX
 1C0B  8B 1C                       MOV    BX,[SI]        ;SAMPLE NUMBER --> BX
 1C0D  F7 E3                       MUL    BX             ;DX:AX = AX * BX
 1C0F  8B C8                       MOV    CX,AX          ;THE NUMBER OF BYTE TO BE PRINTED
                                                         ;IN CX
 1C11  E8 1818 R                   CALL   TEST_TX        ;FILE TRANSMISSION VIA RS 232

 1C14  8D 1E 2009 R                LEA    BX,FLAG2       ;LOAD FLAG2 WITH 0H
 1C18  B0 00                       MOV    AL,00H
 1C1A  88 07                       MOV    [BX],AL

 1C1C  C3                          RET

                     END_TEST      ENDP
                     ;
                     .LIST
                     ;
 1C1D                READ_SENSORS PROC       NEAR
                                   ;------------------------------------------------------;
                                   ;THIS PROGRAM WAS AN INTERRUPT SERVICE ROUTINE.        ;
                                   ; 1. CLOSE THE GATES AND RESET INTERRUPT ERQUIRMENT.;
                                   ;    RESET TIMER.                                      ;
                                   ; 2. READ THE DATA IN COUNTERS 1--9.                   ;
                                   ; 3. RESET THE COUNTERS & CLARE THE SENSORS AREA.      ;
                                   ; 4. INC SAMPLE_No_1 = 10 ? IF YES FLLAG2 = 0.         ;
                                   ;------------------------------------------------------;
 1C1D  FA                          CLI
 1C1E  50                          PUSH   AX
 1C1F  53                          PUSH   BX
 1C20  51                          PUSH   CX
 1C21  52                          PUSH   DX
 1C22  56                          PUSH   SI
 1C23  57                          PUSH   DI
                     .LIST
                        ;PART 1: CLOSE THE GATES AND RESET INTERRUPT ERQUIRMENT & TIMER
 1C24  B0 99                       MOV    AL,CWIOI
 1C26  E6 53                       OUT    IOCTR_2,AL
 1C28  B0 84                       MOV    AL,10000100B        ;AFTER INTERRUPT CLOSE THE
                                                              ;SECOND GATE.
 1C2A  E6 51                       OUT    IOB_2,AL            ;COUNTERS GATES STILL OPEN

 1C2C  B0 70                       MOV    AL,01110000B        ;SET OUT(INTERRUPT REQUEST) 0
 1C2E  E6 23                       OUT    TIMER_M,AL          ;MODE 0 INTERRUPT ON COUNTING
                                                              ;TERMINATION
 1C30  B0 1F                       MOV    AL,1FH              ;IT USED TO BE 1FH FOR 1S.
 1C32  E6 21                       OUT    TIMER_1,AL          ;LOAD LSB
 1C34  B0 C3                       MOV    AL,0C3H             ;IT USED TO BE 0C3H FOR 1S.
```

```
1C36  E6 21                       OUT     TIMER_1,AL          ;LOAD MSB

1C38  B0 B6                       MOV     AL,10110110B        ;CHENNEL 2
1C3A  E6 23                       OUT     TIMER_M,AL          ;MODE 3 SQUARE WAVE GENERATOTR
1C3C  B0 64                       MOV     AL,64H              ;IT USED TO BE 32H FOR 1S.
1C3E  E6 22                       OUT     TIMER_2,AL          ;LOAD LSB
1C40  B0 00                       MOV     AL,00H
1C42  E6 22                       OUT     TIMER_2,AL          ;LOAD MSB

1C44  B9 0003                     MOV     CX,3
1C47  E8 1B9E R                   CALL    BEEP_N

                          ;PART 2: READ THE DATA IN COUNTERS 1--9
1C4A  8D 3E 2330 R                LEA     DI,SENSOR_1
1C4E  E4 40                       IN      AL,COUNTER_1        ;READ LSB
1C50  8A D0                       MOV     DL,AL
1C52  E4 40                       IN      AL,COUNTER_1        ;READ MSB
1C54  8A F0                       MOV     DH,AL               ;NOM THE DATA WAS IN DX
1C56  8B C2                       MOV     AX,DX
1C58  F7 D0                       NOT     AX
1C5A  89 05                       MOV     [DI],AX

1C5C  E4 41                       IN      AL,COUNTER_2        ;READ LSB
1C5E  8A D0                       MOV     DL,AL
1C60  E4 41                       IN      AL,COUNTER_2        ;READ MSB
1C62  8A F0                       MOV     DH,AL               ;NOM THE DATA WAS IN DX
1C64  8B C2                       MOV     AX,DX
1C66  F7 D0                       NOT     AX
1C68  89 45 02                    MOV     [DI+2],AX

1C6B  E4 42                       IN      AL,COUNTER_3        ;READ LSB
1C6D  8A D0                       MOV     DL,AL
1C6F  E4 42                       IN      AL,COUNTER_3        ;READ MSB
1C71  8A F0                       MOV     DH,AL               ;NOM THE DATA WAS IN DX
1C73  8B C2                       MOV     AX,DX
1C75  F7 D0                       NOT     AX
1C77  89 45 04                    MOV     [DI+4],AX

1C7A  E4 60                       IN      AL,COUNTER_4        ;READ LSB
1C7C  8A D0                       MOV     DL,AL
1C7E  E4 60                       IN      AL,COUNTER_4        ;READ MSB
1C80  8A F0                       MOV     DH,AL               ;NOM THE DATA WAS IN DX
1C82  8B C2                       MOV     AX,DX
1C84  F7 D0                       NOT     AX
1C86  89 45 06                    MOV     [DI+6],AX

1C89  E4 61                       IN      AL,COUNTER_5        ;READ LSB
1C8B  8A D0                       MOV     DL,AL
1C8D  E4 61                       IN      AL,COUNTER_5        ;READ MSB
1C8F  8A F0                       MOV     DH,AL               ;NOM THE DATA WAS IN DX
1C91  8B C2                       MOV     AX,DX
1C93  F7 D0                       NOT     AX
1C95  89 45 08                    MOV     [DI+8],AX

1C98  E4 62                       IN      AL,COUNTER_6        ;READ LSB
1C9A  8A D0                       MOV     DL,AL
1C9C  E4 62                       IN      AL,COUNTER_6        ;READ MSB
1C9E  8A F0                       MOV     DH,AL               ;NOM THE DATA WAS IN DX
1CA0  8B C2                       MOV     AX,DX
1CA2  F7 D0                       NOT     AX
1CA4  89 45 0A                    MOV     [DI+10],AX

1CA7  E4 70                       IN      AL,COUNTER_7        ;READ LSB
1CA9  8A D0                       MOV     DL,AL
1CAB  E4 70                       IN      AL,COUNTER_7        ;READ MSB
1CAD  8A F0                       MOV     DH,AL               ;NOM THE DATA WAS IN DX
1CAF  8B C2                       MOV     AX,DX
```

```
1CB1  F7 D0                          NOT     AX
1CB3  89 45 0C                       MOV     [DI+12],AX

1CB6  E4 71                          IN      AL,COUNTER_8      ;READ LSB
1CB8  8A D0                          MOV     DL,AL
1CBA  E4 71                          IN      AL,COUNTER_8      ;READ MSB
1CBC  8A F0                          MOV     DH,AL             ;NOW THE DATA WAS IN DX
1CBE  8B C2                          MOV     AX,DX
1CC0  F7 D0                          NOT     AX
1CC2  89 45 0E                       MOV     [DI+14],AX

1CC5  E4 72                          IN      AL,COUNTER_9      ;READ LSB
1CC7  8A D0                          MOV     DL,AL
1CC9  E4 72                          IN      AL,COUNTER_9      ;READ MSB
1CCB  8A F0                          MOV     DH,AL             ;NOW THE DATA WAS IN DX
1CCD  8B C2                          MOV     AX,DX
1CCF  F7 D0                          NOT     AX
1CD1  89 45 10                       MOV     [DI+16],AX

                                ;PART 3: RESET THE COUNTERS OPEN TIMER COUNTERS GATE
1CD4  E8 0182 R                      CALL    COUNTER_INI       ;RESET THE COUNTERS 1 -- 9
1CD7  E8 01DD R                      CALL    PULS_GENERATE     ;GENERATE PULS TO CLEAR COUNTERS
                                                               ;S & COUNTERS GATES ARE OPEND

                                ;PART 4: THE RTDP
1CDA  E8 01F6 R                      CALL    REAL_TIME_DATA_PROCESSING

                                ;PART 5: INC SAMPLE_No_1 = 10 ? IF YES, CLOSE THE GATE; RESET TIMER

                                     ;AND COUNTERS;
1CDD  8D 1E 2004 R                   LEA     BX,SAMPLE_No_1
1CE1  FF 07                          INC     WORD PTR[BX]
1CE3  83 3F 0A                       CMP     WORD PTR[BX],10
1CE6  75 12                          JNZ     ON_NEXT

1CE8  B0 99                          MOV     AL,CWIOI
1CEA  E6 53                          OUT     IOCTR_2,AL
1CEC  B0 04                          MOV     AL,00000100B
1CEE  E6 51                          OUT     IOB_2,AL

1CF0  B0 70                          MOV     AL,01110000B      ;SET OUT(INTERRUPT REQUEST) 0
1CF2  E6 23                          OUT     TIMER_M,AL        ;MODE 0 INTERRUPT ON COUNTING
                                                               ;TERMINATION
1CF4  E8 0182 R                      CALL    COUNTER_INI       ;RESET THE COUNTERS 1 -- 9
1CF7  E8 1BBF R                      CALL    END_TEST

                                     ;IF NO, CLEAR SENSORS
1CFA  8D 3E 2330 R      ON_NEXT:     LEA     DI,SENSOR_1       ;CLEAR THE SENSORS AREA.
1CFE  B9 000A                        MOV     CX,10
1D01  B8 0000                        MOV     AX,0
1D04  89 05             GPE_1:       MOV     [DI],AX
1D06  47                             INC     DI
1D07  47                             INC     DI
1D08  E2 FA                          LOOP    GPE_1

1D0A  5F                             POP     DI
1D0B  5E                             POP     SI
1D0C  5A                             POP     DX
1D0D  59                             POP     CX
1D0E  5B                             POP     BX
1D0F  58                             POP     AX

1D10  FB                             STI

1D11  CF                             IRET

                        READ_SENSORS ENDP
```

```
                                ;
 1D12                   TEST_59      PROC    NEAR
                                     ;-----------------------------------------------;
                                     ;THIS PROGRAM WAS FOR TEST 8088 INTERRUPT FUNCTION;
                                     ; 1. BOTH IN 8259 AND IN SOFTWARE MODE          ;
                                     ; 2. THE ORG ADDRESSES ARE DEPENDS              ;
                                     ;-----------------------------------------------;
 1D12  FA                            CLI
 1D13  50                            PUSH    AX
 1D14  53                            PUSH    BX
 1D15  51                            PUSH    CX
 1D16  52                            PUSH    DX
                        .LIST
                                     ;PART 1: CLOSE THE GATS AND RESET INTERRUPT ERQUIRMENT
 1D17  B0 99                         MOV     AL,CWIOI
 1D19  E6 53                         OUT     IOCTR_2,AL
 1D1B  B0 04                         MOV     AL,00000100B     ;AFTER INTERRUPT CLOSE THE
 1D1D  E6 51                         OUT     IOB_2,AL         ;SECOND AND COUNTERS GATES

 1D1F  B0 70                         MOV     AL,01110000B     ;SET OUT(INTERRUPT REQUEST) 0
 1D21  E6 23                         OUT     TIMER_M,AL       ;MODE 0 INTERRUPT ON COUNTING
                                                              ;TERMINATION
 1D23  B0 1F                         MOV     AL,1FH
 1D25  E6 21                         OUT     TIMER_1,AL       ;LOAD LSB
 1D27  B0 C3                         MOV     AL,0C3H
 1D29  E6 21                         OUT     TIMER_1,AL       ;LOAD MSB

 1D2B  B0 B6                         MOV     AL,10110110B     ;CHENNEL 2
 1D2D  E6 23                         OUT     TIMER_M,AL       ;MODE SQUARE WAVE GENERATOTR
 1D2F  B0 32                         MOV     AL,32H
 1D31  E6 22                         OUT     TIMER_2,AL       ;LOAD LSB
 1D33  B0 00                         MOV     AL,00H
 1D35  E6 22                         OUT     TIMER_2,AL       ;LOAD MSB

                                     ;PART 2: READ THE DATA IN COUNTERS 1--9
 1D37  8D 3E 2330 R                  LEA     DI,SENSOR_1
 1D3B  E4 40                         IN      AL,COUNTER_1     ;READ LSB
 1D3D  8A D0                         MOV     DL,AL
 1D3F  E4 40                         IN      AL,COUNTER_1     ;READ MSB
 1D41  8A F0                         MOV     DH,AL            ;NOM THE DATA WAS IN DX
 1D43  8B C2                         MOV     AX,DX
 1D45  F7 D8                         NEG     AX
 1D47  89 05                         MOV     [DI],AX

 1D49  E4 41                         IN      AL,COUNTER_2     ;READ LSB
 1D4B  8A D0                         MOV     DL,AL
 1D4D  E4 41                         IN      AL,COUNTER_2     ;READ MSB
 1D4F  8A F0                         MOV     DH,AL            ;NOM THE DATA WAS IN DX
 1D51  8B C2                         MOV     AX,DX
 1D53  F7 D8                         NEG     AX
 1D55  89 45 02                      MOV     [DI+2],AX

 1D58  E4 42                         IN      AL,COUNTER_3     ;READ LSB
 1D5A  8A D0                         MOV     DL,AL
 1D5C  E4 42                         IN      AL,COUNTER_3     ;READ MSB
 1D5E  8A F0                         MOV     DH,AL            ;NOM THE DATA WAS IN DX
 1D60  8B C2                         MOV     AX,DX
 1D62  F7 D8                         NEG     AX
 1D64  89 45 04                      MOV     [DI+4],AX

 1D67  E4 60                         IN      AL,COUNTER_4     ;READ LSB
 1D69  8A D0                         MOV     DL,AL
 1D6B  E4 60                         IN      AL,COUNTER_4     ;READ MSB
 1D6D  8A F0                         MOV     DH,AL            ;NOM THE DATA WAS IN DX
 1D6F  8B C2                         MOV     AX,DX
 1D71  F7 D8                         NEG     AX
 1D73  89 45 06                      MOV     [DI+6],AX
```

```
1D76  E4 61                  IN      AL,COUNTER_5      ;READ LSB
1D78  8A D0                  MOV     DL,AL
1D7A  E4 61                  IN      AL,COUNTER_5      ;READ MSB
1D7C  8A F0                  MOV     DH,AL             ;NOM THE DATA WAS IN DX
1D7E  8B C2                  MOV     AX,DX
1D80  F7 D8                  NEG     AX
1D82  89 45 08               MOV     [DI+8],AX

1D85  E4 62                  IN      AL,COUNTER_6      ;READ LSB
1D87  8A D0                  MOV     DL,AL
1D89  E4 62                  IN      AL,COUNTER_6      ;READ MSB
1D8B  8A F0                  MOV     DH,AL             ;NOM THE DATA WAS IN DX
1D8D  8B C2                  MOV     AX,DX
1D8F  F7 D8                  NEG     AX
1D91  89 45 0A               MOV     [DI+10],AX

1D94  E4 70                  IN      AL,COUNTER_7      ;READ LSB
1D96  8A D0                  MOV     DL,AL
1D98  E4 70                  IN      AL,COUNTER_7      ;READ MSB
1D9A  8A F0                  MOV     DH,AL             ;NOM THE DATA WAS IN DX
1D9C  8B C2                  MOV     AX,DX
1D9E  F7 D8                  NEG     AX
1DA0  89 45 0C               MOV     [DI+12],AX

1DA3  E4 71                  IN      AL,COUNTER_8      ;READ LSB
1DA5  8A D0                  MOV     DL,AL
1DA7  E4 71                  IN      AL,COUNTER_8      ;READ MSB
1DA9  8A F0                  MOV     DH,AL             ;NOM THE DATA WAS IN DX
1DAB  8B C2                  MOV     AX,DX
1DAD  F7 D8                  NEG     AX
1DAF  89 45 0E               MOV     [DI+14],AX

1DB2  E4 72                  IN      AL,COUNTER_9      ;READ LSB
1DB4  8A D0                  MOV     DL,AL
1DB6  E4 72                  IN      AL,COUNTER_9      ;READ MSB
1DB8  8A F0                  MOV     DH,AL             ;NOM THE DATA WAS IN DX
1DBA  8B C2                  MOV     AX,DX
1DBC  F7 D8                  NEG     AX
1DBE  89 45 10               MOV     [DI+16],AX

                             ;PART 5: CONVERT CHENNELS 1 --9 DATA INTO ASCII
                                     ;AND DISPLAY IT ON LED
1DC1  8D 1E 2072 R           LEA     BX,DATA_DISPLAY   ;POINT TO THE ADDRESS OF
1DC5  B9 0008                MOV     CX,8              ;DISPLAYING PROPARE TO DISPLAY
1DC8  B8 2020                MOV     AX,'  '           ; |*  00032*  00456|
1DCB  89 07          CMZ:    MOV     [BX],AX
1DCD  43                     INC     BX
1DCE  43                     INC     BX
1DCF  E2 FA                  LOOP    CMZ               ;NOW THE DATA_DISPLAY EMPTY

1DD1  B0 31                  MOV     AL,'1'            ;STORE SENSORS NUMBER IN RAM
1DD3  A2 2082 R              MOV     [LEFT],AL         ;LEFT SIDE OF LCD
1DD6  B0 32                  MOV     AL,'2'
1DD8  A2 2083 R              MOV     [RIGHT],AL        ;RIGHT SIDE OF LCD

1DDB  8D 36 2330 R           LEA     SI,SENSOR_1       ;POINT TO SENSOR 1
1DDF  8D 1E 2072 R           LEA     BX,DATA_DISPLAY   ;POINT TO FIRST ADDRESS OF LCD

1DE3  B9 0005                MOV     CX,5              ;DISPLAY 9 + 1 PLACES
1DE6  51             UNN:    PUSH    CX
1DE7  56                     PUSH    SI                ;SAVE SENSOR_1
1DE8  53                     PUSH    BX                ;LCD FIRST ADDRESS
1DE9  83 C3 03               ADD     BX,3              ;POINT TO FIRST DATA POSITION
1DEC  E8 1E50 R              CALL    BIN_BCD_ASCII_D   ;NOW RESULT IN [DATA_DISPLAY+3]
1DEF  5B                     POP     BX                ;RESUME LCD FIRST ADDRESS
1DF0  5E                     POP     SI                ;RESUME SENSOR ADDRESS
```

```
1DF1  46                         INC     SI                  ;POINT TO NEXT SENSOR
1DF2  46                         INC     SI
1DF3  56                         PUSH    SI                  ;SAVE NEW SENSOR  ADDRESS
1DF4  53                         PUSH    BX                  ;SAVE LCD FIRST ADDRESS
1DF5  83 C3 0B                   ADD     BX,11               ;POINT TO SECOND DATA ADDRESS
1DF8  E8 1E50 R                  CALL    BIN_BCD_ASCII_D ;NOW RESULT IN [DATA_DISPLAY+11]
1DFB  5B                         POP     BX                 ;RESUME LCD 1st ADDR FOR DISPLAY
1DFC  5E                         POP     SI                  ;RESUME SI AND
1DFD  46                         INC     SI                  ;POINT TO NEXT SENSOR.
1DFE  46                         INC     SI
1DFF  56                         PUSH    SI
1E00  53                         PUSH    BX
1E01  E8 1E35 R                  CALL    SENSOR_NUMBER       ;CHANGE THE SENSORS NUMBER
1E04  E8 1E92 R                  CALL    DISPLAY             ;DISPLAY THE DATA ON LCD
1E07  5B                         POP     BX
1E08  5E                         POP     SI
1E09  59                         POP     CX
1E0A  E2 DA                      LOOP    UNN

1E0C  B0 99                      MOV     AL,CWIOI
1E0E  E6 53                      OUT     IOCTR_2,AL
1E10  B0 8C                      MOV     AL,10001100B        ;AFTER DISPLAY DATA OPEN THE
1E12  E6 51                      OUT     IOB_2,AL            ;TIMER AND SENSORS GATA.

1E14  E8 0182 R                  CALL    COUNTER_INI         ;RESET THE COUNTERS 1 -- 9

1E17  8D 3E 2330 R               LEA     DI,SENSOR_1         ;CLEAR THE SENSORS AREA.
1E1B  B9 000A                    MOV     CX,10
1E1E  B8 0000                    MOV     AX,00H
1E21  89 05              GPE:    MOV     [DI],AX
1E23  47                         INC     DI
1E24  47                         INC     DI
1E25  E2 FA                      LOOP    GPE

1E27  8D 1E 2009 R               LEA     BX,FLAG2            ;IF FLAG5 WAS SET TO 1,
                                                             ;SERVICES
1E2B  B0 01                      MOV     AL,1                ;HAVE BEEN DONF
1E2D  88 07                      MOV     [BX],AL

1E2F  5A                         POP     DX
1E30  59                         POP     CX
1E31  5B                         POP     BX
1E32  58                         POP     AX

1E33  FB                         STI

1E34  CF                         IRET

                         TEST_59 ENDP
                         ;
                         ;
1E35                     SENSOR_NUMBER      PROC      NEAR
                                 ;----------------------------------------;
                                 ;DISPLAY THE SENSORS NUMBER              ;
                                 ;THE FIRST DISPLAY ADDRESS MUST BE IN BX ;
                                 ;----------------------------------------;
1E35  A0 2082 R                  MOV     AL,[LEFT]           ;SENSORS 1, 3, 5, 7, 9
1E38  88 47 01                   MOV     [BX+1],AL
1E3B  40                         INC     AX
1E3C  40                         INC     AX
1E3D  A2 2082 R                  MOV     [LEFT],AL
1E40  A0 2083 R                  MOV     AL,[RIGHT]          ;SENSORS 2, 4, 6, 8
1E43  88 47 09                   MOV     [BX+9],AL
1E46  40                         INC     AX
1E47  40                         INC     AX
1E48  37                         AAA
1E49  0D 3030                    OR      AX,3030H
```

```
 1E4C  A2 2083 R                    MOV     [RIGHT],AL

 1E4F  C3                           RET

                    SENSOR_NUMBER           ENDP
                    ;
                    ;
 1E50               BIN_BCD_ASCII_D         PROC    NEAR
                                    ;-----------------------------------------------------;
                                    ; 1. BIN IN [SI] WAS A WORD TO BE CONVERDED           ;
                                    ; 2. RESULT IN [BX] -- [BX+4] ARE 5 DIGITS ASCII CODE ;
                                    ;-----------------------------------------------------;
 1E50  BA 0000                      MOV     DX,0
 1E53  8B 04                        MOV     AX,[SI]
 1E55  B9 2710                      MOV     CX,10000
 1E58  F7 F1                        DIV     CX
 1E5A  0C 30                        OR      AL,30H
 1E5C  88 07                        MOV     [BX],AL

 1E5E  8B C2                        MOV     AX,DX
 1E60  BA 0000                      MOV     DX,00H
 1E63  B9 03E8                      MOV     CX,1000
 1E66  F7 F1                        DIV     CX
 1E68  0C 30                        OR      AL,30H
 1E6A  88 47 01                     MOV     [BX+1],AL

 1E6D  8B C2                        MOV     AX,DX
 1E6F  BA 0000                      MOV     DX,00H
 1E72  B9 0064                      MOV     CX,100
 1E75  F7 F1                        DIV     CX
 1E77  0C 30                        OR      AL,30H
 1E79  88 47 02                     MOV     [BX+2],AL

 1E7C  8B C2                        MOV     AX,DX
 1E7E  BA 0000                      MOV     DX,00H
 1E81  B9 000A                      MOV     CX,10
 1E84  F7 F1                        DIV     CX
 1E86  0C 30                        OR      AL,30H
 1E88  88 47 03                     MOV     [BX+3],AL
 1E8B  80 CA 30                     OR      DL,30H
 1E8E  88 57 04                     MOV     [BX+4],DL

 1E91  C3                           RET

                    BIN_BCD_ASCII_D         ENDP
                    ;
                    ;
 1E92               DISPLAY     PROC        NEAR
                              ;-------------------------------------------------------------;
                              ;THIS PROCRDURE WAS TO DISPLAY 16 ASCII CHARACTERS ON LCD     ;
                              ;  1. THE CHARACTERS ADDRESS WAS IN BX                         ;
                              ;  2. AFTER DISPLAY THE MESAGE WILL KEEP ON LCD FOR 1 SECOND   ;
                              ;  3. PUSH CX WAS RECONMANDED BEFOR CALLING.                   ;
                              ;-------------------------------------------------------------;
 1E92  BA 0000                      MOV     DX,POINT
 1E95  89 16 200E R                 MOV     [POINT_S],DX

 1E99  E8 17AB R    CCCC1:          CALL    POSITION
 1E9C  8B 07                        MOV     AX,[BX]
 1E9E  E8 1A29 R                    CALL    DATA_D
 1EA1  E8 1A48 R                    CALL    CHECK
 1EA4  43                           INC     BX
 1EA5  B0 10                        MOV     AL,10H
 1EA7  FE 06 2008 R                 INC     [FLAG1]
 1EAB  3A 06 2008 R                 CMP     AL,[FLAG1]
 1EAF  75 E8                        JNZ     CCCC1
 1EB1  B9 03E8                      MOV     CX,1000
```

```
 1EB4  E8 1EE8 R                         CALL     DN1MS

 1EB7  B0 00                             MOV      AL,00H          ;DIS HOUR--SECND
 1EB9  A2 2008 R                         MOV      [FLAG1],AL

                          DISPLAY        ENDP
                          ;
                          ;
 1EBC                     DISPLAY_1      PROC     NEAR
                                  ;-------------------------------------------------------;
                                  ;THIS PROCRDURE WAS TO DISPLAY 16 ASCII CHARACTERS ON LCD  ;
                                  ;  1. THE CHARACTERS ADDRESS WAS IN BX                    ;
                                  ;  2. PUSH CX WAS RECONMANDED BEFOR CALLING.              ;
                                  ;-------------------------------------------------------;
 1EBC  BA 0000                           MOV      DX,POINT
 1EBF  89 16 200E R                      MOV      [POINT_S],DX

 1EC3  E8 17AB R          CCCC11:        CALL     POSITION
 1EC6  8B 07                             MOV      AX,[BX]
 1EC8  E8 1A29 R                         CALL     DATA_D
 1ECB  E8 1A48 R                         CALL     CHECK
 1ECE  43                                INC      BX
 1ECF  B0 10                             MOV      AL,10H
 1ED1  FE 06 2008 R                      INC      [FLAG1]
 1ED5  3A 06 2008 R                      CMP      AL,[FLAG1]
 1ED9  75 E8                             JNZ      CCCC11

 1EDB  B0 00                             MOV      AL,00H          ;DIS HOUR--SECND
 1EDD  A2 2008 R                         MOV      [FLAG1],AL

                          DISPLAY_1      ENDP
                          ;
                          ;
 1EE0                     ONE_MS         PROC     NEAR
                                  ;--------------------------------------------;
                                  ;  THIS PROGRAM EXECUTION TIME WAS ABOUT 1 mS ;
                                  ;--------------------------------------------;
 1EE0  B9 0106            D1MS:          MOV      CX,0106H
 1EE3  FE CE              D1MS1:         DEC      DH
 1EE5  E2 FC                             LOOP     D1MS1

 1EE7  C3                                RET

                          ONE_MS         ENDP
                          ;
                          ;
 1EE8                     N_MS           PROC     NEAR
                                  ;--------------------------------;
                                  ;  IF THE DELAY TIME WAS N*1ms,  ;
                                  ;  LET CX = N  AND CALL N_MS.    ;
                                  ;--------------------------------;
 1EE8  51                 DN1MS:         PUSH     CX
 1EE9  E8 1EE0 R                         CALL     D1MS
 1EEC  59                                POP      CX
 1EED  E2 F9                             LOOP     DN1MS

 1EEF  C3                                RET

                          N_MS           ENDP
                          ;
                          ;
 1FF0                                    ORG      1FF0H
                                  ;----------------------------;
                                  ;           RESET            ;
                                  ;----------------------------;
 1FF0                     RESET          PROC     NEAR
 1FF0  E9 0000 R                         JMP      START
```

```
                              RESET        ENDP
                              ;
                              ;
 1FF3                         GRAIN        ENDS

                              END          START
```