UNIQUE MASSFLOW ESTIMATION TECHNIQUE FOR HORIZONTAL CONVEYOR BELTS

by
Karl Claus Alexander DIERKS

July 1990

A thesis submitted to the Department of Electrical Engineering, University of Cape Town, in partial fulfillment of the requirements for a degree of M.Sc. in Electrical Engineering.

For the Lord is the great God,
the great King above all gods.
In his hands are the depths of the earth,
and the mountain peaks belong to him.
The sea is his, for he made it,
and his hands formed the dry land
Psalm 95:3-5

ACKNOWLEDGEMENTS

I would like to thank the following:

My supervisor, Professor Martin Braae for his expert advice and encouragement throughout my studies.

The Diamond Research Laboratories of De Beers Industrial Diamond Division (Pty) Ltd. for initiating and providing the funding for this project.

Hermann Potgieter, Geoff Cragg and all other members of the D.R.L. for their supervision and help during my stay in Johannesburg.

The Foundation of Research and Development for their financial assistance.

Ingrid, my dear wife, for your support, understanding and love during this time.

ABSTRACT

Massflows on conveyor belts need to be measured in various process streams in mineral extraction processes to ensure good plant management. Conventional weightometers are, however, too expensive to allow a widespread installation in these processes.

This study investigates more economic means to measure the massflow on horizontal conveyor belts. A deterioration of accuracy resulting from an instrument using this technique is expected, but the advantages of being able to instrument entire plants is very attractive.

The study uses a model whereby the kinetic energy of the belt system is determined during a perturbation to the system. From this the massflow can be determined.

The technique was optimized in a laboratory environment for different modes of operation. Both Variable Speed Drive and Solid State Relays were used to perturb the system. The resulting accuracies were compared against each other.

Hence a prototype instrument was proposed which was easy to install on existing conveyor belts. This unit was tested successfully in a mining environment.

TABLE OF CONTENTS

HEADING:	page
ACKNOWLEDGEMENTS	i
ABSTRACT	ii
LIST OF FIGURES	xi
LIST OF TABLES	xix
LIST OF PHOTOGRAPHS	xxii
NOMENCLATURE	xxiii
CHAPTER 1: INTRODUCTION	1
CHAPTER 2: A TECHNIQUE TO MEASURE MASS FLOW ON A HORIZONTAL CONVEYOR BELT	6
2.1 The Relation between Massflow and Inertia of a System	6
2.2 A Technique to measure Inertia of a System	14
2.3 The Techniques Analogy in the Translational Frame of Reference	19
2.4 Suggestions to make the Run-Down Test suitable for On-Line Measurements	22
2.4.1 Using the Energy Balance rather than the Power Balance	22
2.4.2 Determining the Load Force before each Rundown Test	25
2.4.3 Determining the Mass from Step-up Responses	27
2.4.4 Adapting the System so that the Conveyor Belt needs only be slowed down.	29
2.5 Incorporating all Suggestions into a Single Test Procedure	31

2.6 Shortcomings of the Test Procedure 35
2.6.1 Modelling the Load Characteristics with respect to Speed 35
2.6.2 Incorporating the Efficiency of the Motor 37
2.6.3 Measuring the Speed of the Motor 41

CHAPTER 3: BASIC STATISTICS AND CALIBRATION 42

3.1 Introduction 42
3.2 The Static Characteristics of an Instrument 45
3.2.1 Static Calibration 45
3.2.2 Bias and Imprecision 47
3.2.3 Removing Bias 49
3.2.4 Quantifying Imprecision 53
3.3 The Dynamic Characteristics of an Instrument 63
3.4 Concluding Remarks 67

CHAPTER 4: THE SOFTWARE USED TO MEASURE AND CALIBRATE THE MASSFLOW ON A HORIZONTAL CONVEYOR BELT 69

4.1 General Remarks about Software 69
4.1.1 Basic Aims of Software 69
4.1.2 Hardware Requirements 69
4.1.3 General Software Development Policies 70
4.1.4 Documentation and Commenting 74
4.1.5 Data Formats used 74
4.2 The Software 77
4.2.1 Manual Data Acquisition and Correlation 79
4.2.2 Automatic Data Acquisition and Correlation 81
4.2.3 Automatic Data Acquisition 82

CHAPTER 5: VALIDATION AND CALIBRATION OF MASSFLOW ESTIMATION TECHNIQUE IN A LABORATORY ENVIRONMENT 84

5.1 Description of basic Plant 84
5.2 Specifications of the 380V Circuitry 88
5.2.1 Horizontal Conveyor Belt 88
5.2.2 Inclined Conveyor Belt 91
5.2.2 Vibrating Feeder 91
5.3 Instrumentation Circuitry 92
5.3.1 The Control Signals for the SSRs and the VSD 94
5.3.2 Speed 96
5.3.3 Power 97
5.3.4 Massflow 104
5.3.5 Other Variables to consider 111
A - Friction of the system 111
B - Sampling frequency 112
C - timing of Datalogging Routine 113
5.4 Results of Tests performed 116
5.4.1 Validating the Technique to use 116
5.4.2 On-line Calibration Results 122
5.4.3 Cross Validation Results 134
5.4.3 Effects of Filtering 137
5.5 Proposed Instrument 139

CHAPTER 6: VALIDATION OF THE PROTOTYPE IN AN INDUSTRIAL ENVIRONMENT 145

6.1 Description of Basic Plant 145
6.2 Specifications of 380V Circuitry 148
6.3 Instrumentation Circuitry 149
6.3.1 The Control Signal for the VSD 151
6.3.2 Speed 152
6.3.3 Power 153
6.3.4 Massflow 154
6.3.5 Time Constant of Plant 155

6.4 Results 156
6.4.1 Constant Speed Tests 156
6.4.2 Controlled Speed Results 164

CHAPTER 7: SUMMARY AND CONCLUSIONS 170

7.1 Summary 170
7.2 Conclusions 180
7.3 Recommendations for Future Work 182

REFERENCES 183

BIBLIOGRAPHY 185

APPENDIX A: INCORPORATING THE LOAD CHARACTERISTICS IN THE MASSFLOW ESTIMATION ALGORITHM 188

APPENDIX B: TABLE OF F-DISTRIBUTION NUMBERS 191

APPENDIX C: LISTINGS OF SOFTWARE 193

C.1 Index to Software Routines 193
C.2 Summary of Software Routines 197
C.2.1 General and Basic Menu Routines 197
C.2.2 Filehandling Routines 197
C.2.3 Mathematics Routines 199
C.2.4 Graphics Routines 201
C.2.5 Datalogging Routines 202
C.2.6 Specialized Massflow Meter Routines 203
C.2.7 Specialized System Identification

Routines 204
C.2.8 Data Routines 206
C.2.9 Main Programs 209
C.3 Listings of Software 210
C.3.1 Types.pas 210
C.3.2 Filehand.pas 214
C.3.3 Math.pas 226
C.3.4 Plotgraf.pas 236
C.3.5 Plotgrad.pas 245
C.3.6 DT2801_4.pas 247
C.3.7 Detmf.pas 249
C.3.8 Detmfd.pas 254
C.3.9 Datalog.pas 256
C.3.10 Dlog.pas 262
C.3.11 Dlogd.pas 266
C.3.12 Graf.pas 268
C.3.13 Grafd.pas 271
C.3.14 Mfcm.pas 273
C.3.15 Mfcmd.pas 281
C.3.16 Mfca.pas 286
C.3.17 Mfcad.pas 291
C.3.18 Mf.pas 294
C.3.19 Mfd.pas 298
C.3.20 Estimate.pas 301
C.3.21 Model.pas 308
C.3.22 Modeld.pas 315
C.3.23 Detfric.pas 319
C.3.24 Detfricd.pas 323
C.3.25 Dconv.pas 326
C.3.26 Dconvd.pas 329
C.3.27 Fit.pas 330
C.3.28 Fitd.pas 332

APPENDIX D: EXPLANATION OF MENU STRUCTURES AND FUNCTIONS OF INDIVIDUAL MENUS 333

D.1 DLOG.EXE 333
D.1.1 Basic Menu Structure 333
D.1.2 Description of Available Options 333
D.1.3 Copies of the Menus used 335
D.2: MFCM.EXE 336
D.2.1 Basic Menu Structure 336
D.2.2 Description of Available Options 336
D.2.3 Copies of the Menus used 340
D.3: GRAF.EXE 346
D.3.1 Basic Menu Structure 346
D.3.2 Description of Available Options 346
D.3.3 Copies of the Menus used 347
D.4: MFCA.EXE 348
D.4.1 Basic Menu Structure 348
D.4.2 Description of Available Options 348
D.4.3 Copies of the Menus used 352
D.5: MF.EXE 354
D.5.1 Basic Menu Structure 354
D.5.2 Description of Available Options 354
E.5.3 Copies of the Menus used 357

APPENDIX E: DATA FORMATS USED IN SOFTWARE 359

E.1 Data A: Real Time Response Data Format 359
E.2 Data B: Correlation Data Format 360
E.3 Data C: Massflow Data Format 361

APPENDIX F: CIRCUITS USED FOR LABORATORY TEST RIG 362

APPENDIX G: CALIBRATION RESULTS OF LABORATORY TEST RIG 366

G.1 Calibration and Regression Results for speed 366
G.2 Calibration and Regression Results for Power 367
G.3 Calibration and Regression Results for Reference Massflow 369
G.4 Load Friction Model 371
G.5 Calculation of Time Constant of Test Rig 373

APPENDIX H: RESULTS FROM LABORATORY TESTS 374

H.1 Speed and Power Responses from Tests 374
H.1.1 Various Speed Steps applied with VSD 374
H.1.1 Various Initial Speed Setpoints using VSD to Perturb the system 378
H.2 Correlation Graphs and Regression Results 380
H.2.1 Constant Initial Speed; Different Steps 380
H.2.2 Differing Initial Speed; Total Step 384
H.2.3 SSR: Stopping the HCB 387
H.2.4 SSR: Slowing HCB down 389
H.2.5 Effect of Filtering the Samples 391

APPENDIX I: CIRCUITS USED FOR MINE TEST RIG 394

APPENDIX J: CALIBRATION RESULTS FOR MINE TEST RIG 398

APPENDIX K: TEST RESULTS FROM MINE TEST RIG 401

K.1 Real time Pulse Responses 401
K.2 Correlation Curves For Different Filter Time Constants 404
K.3 Time Plots and Correlation Graphs for Different Speed Setpoint 406

K.4 Cross Validation Results 410
K.4.1 Initial Speed Setpoint = 75% 410
K.4.1 Initial Speed Setpoint = 50% 411
K.4.1 Initial Speed Setpoint = 25% 412
K.5 Results from Controlled Speed Tests 413
K.5.1 Time Plots for Different Filter Time Constants 413
K.5.2 Correlation Graphs for Different Filter Time Constant 414

List of Figures

Chapter 2:

2.1 Schematic of a horizontal conveyor belt 6
2.2 Block diagram of a conveyor belt 7
2.3 Force balance of a conveyor belt 10
2.4 Model of a conveyor belt with the its mass reduced to a point mass 10
2.5 Point mass moved to driving roller 11
2.6 A mass moving along a horizontal track 14
2.7 Data logging strategy to determine the load force before each test 25
2.8 A typical test response of power and speed 32
2.1 Flow diagram describing a proposed test cycle 33
2.9 Efficiency characteristics of an induction motor 38
2.10 Load characteristics of an induction motor 38
2.11 Efficiency load characteristics of an induction motor 39

Chapter 3:

3.1 A typical random scatter 54
3.2 The Gaussian Bell Curve 55
3.3 A typical best fit line and the associated 95% boundary limits 60
3.4 Accounting for the dynamics of a measuring device 63
3.5 Incorporating the dynamics of a process 65
3.6 Typical correlation graph (Best fit line and error bounds) 68

Chapter 4:

4.1 Hierarchical structure of programs and subroutines 72
4.2 Heading of a program to comment on the program 74
4.3 Basic function of software 77
4.4 Required function structure of whole process 78
4.5 Structure of functions in DLOG.EXE 79
4.6 Structure of functions in MFCM.EXE 80
4.7 Structure of functions in MFCA.EXE 82
4.8 Structure of functions in MF.EXE 83

Chapter 5:

5.1 Schematic diagram of laboratory test rig 85
5.2 Circuit diagram for 380 V network of the horizontal conveyor belt 89
5.3 Circuit diagram for instrumentation of the horizontal conveyor belt 93
5.4 Block circuit diagram to add currents 95
5.5 Measuring power using PT using the two Watt meter method 98
5.6 Measuring power using PT using the three Watt meter method 98
5.7 Input and output current wave shapes of a current transformer 100
5.8 Power factor load characteristics of an induction motor 102
5.9 Correlation graph illustrating the effect of Hysteresis due to the reference massflow determination technique 109
5.10 Confirming the timing of the data logging routine with a 250 msec pulse 115
5.11 Typical speed pulse response for a 75% step in speed 117
5.12 Typical power pulse response for a 75% step in speed 118
5.13 Typical speed pulse response for a 50% step in

speed 123
5.14 Typical power pulse response for a 50% step in speed 123
5.15 Imprecision versus percentage step applied 125
5.16 Typical speed pulse response for an initial speed setpoint of 50% of full speed 126
5.17 Typical power pulse response for an initial speed setpoint of 50% of full speed 126
5.18 Correlation of Y-intercept versus initial setpoint 128
5.19 Correlation of slope versus initial setpoint 128
5.20 Typical speed pulse response for the system being perturbed with SSRs 129
5.21 Typical power pulse response for the system being perturbed with SSRs 130
5.22 Typical speed pulse response for the system being only slowed down with the SSRs 131
5.23 Typical power pulse response for the system being only slowed down with the SSRs 132
5.24 Time graph of reference and estimated massflow 135
5.25 Block diagram to illustrate functional layout of proposed instrument 140
5.26 A typical test response of power and speed 142

Chapter 6:

6.1 Schematic diagram of test rig in an industrial environment 146
6.2 Circuit diagram for the 380 V network of the test rig used on a mine 149
6.3 Circuit diagram for instrumentation to be used on a mine 150
6.4 Speed pulse response for v_0 = 50% 157
6.5 Power Pulse Response for v_0 = 100% 157
6.6 MF time plot for τ_t = 60s 158
6.7 MF time plot for τ_t = 600s 158
6.8 MF time plot for τ_t = 6000s 159
6.9 Time plot for Test 1 when speed is controlled;

τ_f = 600s 164
6.10 Time plot for Test 1 when speed is controlled;
τ_f = 6000s 166

APPENDIX D:

D.1 Menu structure of DLOG.EXE 333
D.2 Main menu of DLOG.EXE 335
D.3 Change Format menu of DLOG.EXE 335
D.4 Menu structure of MFCM.EXE 336
D.5 Flowchart of to illustrate the working of the Manual Data Acquisition procedure 338
D.6 Main menu of MFCM.EXE 340
D.7 Data Processing menu 341
D.8 Graphical Display menu in Data Processing option 341
D.9 Change Scale menu in Data Processing option 342
D.10 Menu for Manual Massflow Data Acquisition using real time responses 342
D.11 Menu to define filenames in Manual Data Acquisition 343
D.12 Menu to define variables in Manual Data Acquisition 343
D.13 Menu to display correlation results 344
D.14 Menu to change scales of correlation graph 344
D.15 Transfer Files menu 345
D.16 Menu structure of GRAF.EXE 346
D.17 Main menu of GRAF.EXE 347
D.18 Menu to define files to display 347
D.19 Menu structure of MFCA.EXE 348
D.20 Flow diagram for massflow test if reference massflow is obtained from mass in the hopper bin 350
D.21 Flow diagram for massflow test if reference massflow is obtained from a massflow meter 351
D.22 Main menu of MFCA.EXE 352
D.23 Change Fixed Variables menu in MFCA.EXE 352
D.24 Menu to calibrate massflow constants 353
D.25 Menu structure of MF.EXE 354
D.26 Flow diagram for massflow correlation of Massflow 356
D.27 Main menu of MF.EXE 357

D.28 Change Fixed Variables menu 357
D.29 Statistical Data display menu 358
D.30 Menu to calibrate massflow constants 358

APPENDIX F:

F.1 Circuit diagram for the 380 V network of the inclined conveyor belt 362
F.2 Circuit for vibrating feeder of horizontal conveyor belt 363
F.3 Circuit diagram of op-amp adding circuit 363
F.4 Anti-aliasing filter and A/D protection 364
F.5 Schematic circuit diagram of load cell amplifier 364
F.6 Circuit diagram of power transducer 365

APPENDIX G:

G.1 Calibration of speed for laboratory test rig 366
G.2 Calibration graph for power transducer when system is operating with VSD 367
G.3 Calibration graph for power transducer when system is operating in Direct Mode 368
G.4 Calibration graph of mass in hopper bin 369
G.5 Confirming calibration for massflow on horizontal conveyor belt 370
G.6 Modelling load friction with a first order model 371
G.7 Modelling load friction with a second order model 372
G.8 Typical pulse response of laboratory test rig 373

APPENDIX H:

H.1 Speed response of system; Initial Speed = 100%; Step = 100% 374
H.2 Power response of system; Initial Speed = 100%; Step = 100% 374

H.3 Speed response of system; Initial Speed = 100%;
Step = 75% 375
H.4 Power response of system; Initial Speed = 100%;
Step = 75% 375
H.5 Speed response of system; Initial Speed = 100%;
Step = 50% 376
H.6 Power response of system; Initial Speed = 100%;
Step = 50% 376
H.7 Speed response of system; Initial Speed = 100%;
Step = 25% 377
H.8 Power response of system; Initial Speed = 100%;
Step = 25% 377
H.9 Speed response of system; Initial Speed = 100%;
Step = 100% 378
H.10 Power response of system; Initial Speed = 100%;
Step = 100% 378
H.11 Speed response of system; Initial Speed = 50%;
Step = 50% 379
H.12 Power response of system; Initial Speed = 50%;
Step = 50% 379
H.13 Correlation graph; Initial speed setpoint = 100%;
Step = 100% 380
H.14 Correlation graph; Initial speed setpoint = 100%;
Step = 75% 381
H.15 Correlation graph; Initial speed setpoint = 100%;
Step = 50% 382
H.16 Correlation graph; Initial speed setpoint = 100%;
Step = 25% 383
H.17 Correlation graph; Initial speed setpoint = 100%;
Step = 100% 384
H.18 Correlation graph; Initial speed setpoint = 75%;
Step = 75% 385
H.19 Correlation graph; Initial speed setpoint = 50%;
Step = 50% 386
H.20 Correlation graph for step -down response:
Perturbation with SSR 387
H.21 Correlation graph for step -up response:
Perturbation with SSR 387

H.22 Correlation graph for step-down response:
Perturbation with SSR; System only slowed down 389
H.23 Correlation graph for step-up response:
Perturbation with SSR; System only slowed down 389
H.24 Time plot of estimated and reference massflow for
τ_f = 6s 391
H.25 Correlation graph of estimated versus reference
massflow for τ_f = 6s 391
H.26 Time plot of estimated and reference massflow for
τ_f = 60s 392
H.27 Correlation graph of estimated versus reference
massflow for τ_f = 60s 392
H.28 Time plot of estimated and reference massflow for
τ_f = 600s 393
H.29 Correlation graph of estimated versus reference
massflow for τ_f = 600s 393

APPENDIX I:

I.1 Detailed circuit diagram of 380V network 394
I.2 Detailed circuit diagram of instrumentation circuit 395
I.3 Circuit diagram of Sample and Hold circuit
(Linear 1 Databook [10]) 395
I.4 Timing diagram for Sample and Hold circuit 396
I.5 Circuit diagram for trigger signal 396
I.6 Timing diagram for trigger signal circuit 397

APPENDIX J:

J.1 Correlation graph for speed of conveyor belt 398
J.2 Typical pulse response of test plant on a mine 400

APPENDIX K:

K.1 Speed pulse response for v_0 = 100% 401
K.2 Power pulse response for v_0 = 100% 401
K.3 Speed pulse response for v_0 = 75% 402
K.4 Power pulse response for v_0 = 75% 402
K.5 Speed pulse response for v_0 = 25% 403
K.6 Power pulse response for v_0 = 25% 403
K.7 Correlation curve for τ_f = 60 s 404
K.8 Correlation curve for τ_f = 600 s 404
K.9 Correlation curve for τ_f = 6000 s 405
K.10 Time plot for test done at v_0 = 100% 406
K.11 Correlation graph for test done at v_0 = 100% 406
K.12 Time plot for test done at v_0 = 75% 407
K.13 Correlation graph for test done at v_0 = 75% 407
K.14 Time plot for test done at v_0 = 50% 408
K.15 Correlation graph for test done at v_0 = 50% 408
K.16 Time plot for test done at v_0 = 25% 409
K.17 Correlation graph for test done at v_0 = 25% 409
K.18 Time plot for Test 2 when speed is controlled; τ_f = 600s 413
K.19 Time plot for Test 2 when speed is controlled; τ_f = 6000s 413
K.20 Calibration graph for Test 1 when speed is controlled; τ_f = 600s 414
K.21 Calibration graph for Test 2 when speed is controlled; τ_f = 600s 414
K.22 Calibration graph for Test 1 when speed is controlled; τ_f = 6000s 415
K.23 Calibration graph for Test 2 when speed is controlled; τ_f = 6000s 415

6.4 Table of cross validation results for v_e = 100% 163
6.5 Table of correlation coefficients for different filter time constants 166
6.6 Cross validation of MF results when speed is controlled 167
6.7 Table of calibration constants from correlation results 169
6.8 Table of calibration constants from cross validation tests 169

APPENDIX G:

G.1 Regression results from speed calibration 366
G.2 Regression results for calibration of PT when system is running with the VSD 367
G.3 Regression results for calibration of PT when system is running in Direct Mode 368
G.4 Regression results of calibration of mass of bin 369
G.5 Regression results from massflow confirmation 370
G.6 Regression results of modelling the load friction with a first order model 371
G.7 Regression results of modelling the load friction with a second order model 372

APPENDIX H:

H.1 Regression results; Initial speed setpoint = 100%; Step = 100% 380
H.2 Regression results; Initial speed setpoint = 100%; Step = 75% 381
H.3 Regression results; Initial speed setpoint = 100%; Step = 50% 382
H.4 Regression results; Initial speed setpoint = 100%; Step = 25% 383
H.5 Regression results; Initial speed setpoint = 100%; Step = 100% 384

H.6 Regression results; Initial speed setpoint = 75%; Step = 75% 385
H.7 Regression results; Initial speed setpoint = 50%; Step = 50% 386
H.8 Regression results; Perturbation with SSR 388
H.9 Regression results; Perturbation with SSR; System only slowed down 390

APPENDIX J:

J.1 Regression results for speed calibration 398
J.2 Calibration results for nuclear weightometer 399

APPENDIX K:

K.1 Table of cross validation results for v_0 = 75% 410
K.2 Table of cross validation results for v_0 = 50% 411
K.3 Table of cross validation results for v_0 = 25% 412

List of Photographs

CHAPTER 5:

5.1 Photograph of the test plant at the DRL 86
5.2 Photograph of the control panel of the test rig 86

CHAPTER 6:

6.1 Photo of the test rig on a Mine 147
6.2 Photo of the control panel of the test rig on a mine 147
6.3 Photo to illustrate mounting of tacho 152

NOMENCLATURE

Symbol	Meaning
a	acceleration
b	Y-intercept of best fitted line
b_i	bias
CB	conveyor belt
CT	current transformer
C_1	constant to simplify writing of error equation
C_2	same as C_1
DLOG	datalogging package
DRL	Diamond Research Laboratories
E	energy
E	in the statistical sense: error as modelled by elliptical boundary
e_i	error of imprecision
F	F-number
F_{acc}	acceleration force
F_d	driving force
F_l	load force
f(x)	Probability Density Function
F(x)	cummulative Probability Density Function
GRAF	graphical display package
HB	hopper bin
HCB	horizontal conveyor belt
ICB	inclined conveyor belt
J	inertia

l_{belt} length of conveyor belt
l_h = l_{belt}
l_i length of inclined conveyor belt
l_{ch1} length of belt modeling Chute 1
l_{ch2} length of belt modeling Chute 2
LSE Least Squares Estimation

M mass on conveyor belt
m slope of best fit line through correlation data
M_{belt} total mass on all belts in the system
M_{ch1} mass in Chute 1
M_{ch2} mass in Chute 2
M_{dn} mass as determined from step-down response
MF massflow
MF massflow estimation package
MFCA massflow correlation package with automatic data acquisition
MFCM massflow correlation package with manual data acquisition
MFM massflow meter
M_h = M
M_{hopper} mass in hopper bin
M_i mass on inclined conveyor belt
M_{tot} total mass in hopper bin and on belts
M_{up} mass as determined from step-up response

n no of readings

P_{belt} Power delivered to belt
PC Personal Computer
P_d driving power of belt
PDF Probability Density Function
pf power factor
P_l load power of belt
P_{shaft} power on shaft of motor = P_{belt}
P_{supp} power from supply delivered to motor
PT power transducer

$P(a<x<b)$	probability of x lying between a and b
r	radius of driving drum of belt
RMS	Root Mean Square
SD	Standard Deviation
SSR	solid state relay
T	torque
t	time
T_{acc}	acceleration torque
T_l	load torque
T_d	driving torque
v	speed of conveyor belt
v_{ch1}	speed of belt modelling Chute 1
v_{ch2}	speed of belt modelling Chute 2
v_h	speed of horizontal conveyor belt
v_i	speed of inclined conveyor belt
VSD	Variable Speed Drive
v_0	speed in unperturbed mode
v_1	speed in perturbed mode
x	distance covered by conveyor belt
x	in the statistical sense: true or reference value to be measured
$\bar{x}$	mean of true measurement data
x_m	$= \bar{x}$
y	actual measurement data
$\bar{y}$	mean of measurement data
y_m	$= \bar{y}$

α angular acceleration

ε_{mot} efficiency of motor

μ mean value of a set of measurement data representing the same true value

σ Standard Deviation

σ_m Standard error of measurement values

τ_p time constant of plant

τ_s sampling period of PC

τ_f time constant of filter

ω angular velocity of motor

CHAPTER 1 :
INTRODUCTION

Conveyor belts (CB) are used to transport all sorts of material from one unit process to the next. In the Mining Industry ore is transported extensively using CBs.

As these CBs can be loaded to various degrees it is important for good plant management to know what the massflow (MF) is along these ore paths. For instance the MF needs to be known,

- to monitor the efficiency, the performance of a plant and thus its economic viability. Hence the MF at the input and at the output of the plant has to be known.

- to determine the total amount of material handled and the average MF. The MF signal would have to be integrated or totalized to determine such values.

- between the different unit processes to optimize a plant.

- to ensure the proper blending of chemicals in chemical processes by weighing the ingredients.

- to ensure the efficient operation of the equipment used on these plants. The efficiency of for instance a pulverizing mill very often depends on its loading, which means the MF needs to be

controlled to ensure its efficient operation.

Two main criteria determine the usefulness of a massflow determination technique or meter: *accuracy* and *cost*. These two oppose one another. The more accurate a certain instrument, the higher the cost, and vice versa. Thus a trade off is necessary. Looking at the spectrum of massflow meters (MFM) available on the market, one could get the impression that the market tends to be biased towards high accuracy/high cost instruments.

Various techniques are used to measure the MF in these high accuracy instruments. All of them though are similar in that the weight of the material is measured first. Then the MF can be determined by multiplying the weight reading with the speed of the belt.

The most commonly used MFM in mineral extraction plants is the nuclear weightometer. Considine[1] calls this instrument a radioactive transmission gauge of the absorption type and describes it as follows: Beta and gamma radiation is first passed through an ionization chamber and then passed through the moving material which is to be weighed. The attenuation of the signal is a measure of the material weight per unit area. Advantages of this instrument are high accuracy (suppliers quote up to 0.5%), and the ease with which it can be installed on existing CBs. Disadvantages obviously include the high cost of such an instrument.

Another common technique is batch weighing. Considine[1] describes this technique as follows: The material is discharged into a hopper, which fills until a certain definite weight is reached. The inflow is then shut off, the hopper and its contents are weighed, and then the contents is discharged. The obvious disadvantage of this technique is that the MF has to be shut off for the purpose

of weighing. This means that the technique is not continuous as production is cut for some time. A further disadvantage is that the CB has to be specially adapted to suit this technique. However the advantage of this technique is its high accuracy (Mitchell[12] quotes the accuracy to be about 0.1% of full scale).

A third technique is that of continuous weigh feeders (Mitchell[12]) or scale-balanced belt-feeders (Considine[1]). This technique is describes by Considine as follows: Either a portion of or the whole CB is isolated and the mass on the CB including the material is weighed. From the variations in weight due to changes in MF the MF can be determined by integration. The advantage of this technique is the relative high accuracy that can be achieved (1-2% for single idler and 0.5% for multi idler techniques - Mitchell[12]). A further advantage is that any material can be used, because the technique relies on the weight of the material alone, and not on its radioactive absorption characteristics for instance. The integration of instantaneous loads also corrects for variations in belt speed. An Obvious disadvantage is the special adaptation necessary to install such an instrument on an existing CB.

No provision, however, seems to be made for a low cost MFM on the market. Such an instrument obviously would not have the same accuracy requirements as the more expensive ones.

In mineral extraction processes the MF along various ore paths need not be determined to the same high degree of accuracy. An example is that if the total MF into a plant is known to a high degree of accuracy, then the MF along individual and parallel CBs inside this plant need not be known to the same high degree of accuracy, as the proportional loading is of prime relevance.

A view shared by Considine[1] is that such an instrument should then be used in conjunction with more accurate instruments. The results from both types of instrument could be fed into a Mass smoothing package (Potgieter[13]), which acts as a filter, using a Kalman filtering technique to obtain a best fit for all the readings. Such packages also account for the different accuracies associated with the readings by weighting the individual readings appropriately.

The following question is thus posed: Is there a technique, which could be used to manufacture a low cost MFM, even at the expense of accuracy. Such an instrument could then achieve widespread installation within the whole mining industry. An additional requirement would be that such an instrument should be easy to install on existing CBs in the mines.

An initial investigation was done to test the feasibility of measuring MF by measuring the Inertia of a given system on-line (Dierks[2]). This study proved on an experimental basis that the inertia of an induction motor could be measured by perturbing the input to the motor. More specifically the frequency of the supply to the motor was perturbed. Following this initial investigation a provisional patent (Dierks[3]) was taken out on this technique by De Beers Industrial Diamond Division (Pty) Ltd. and a more detailed study of this method was initiated.

In this thesis first the theory to estimate the massflow on a conveyor belt and the basic statistics involved in the process of calibration will be described. This is followed by a description of the software developed to perform the estimation and calibration of the massflow. Then the technique is validated first in a laboratory type environment and then in an industrial environment. Lastly conclusions and recommendations are made.

CHAPTER 2:
A TECHNIQUE TO MEASURE MASS FLOW ON A HORIZONTAL CONVEYOR BELT

2.1 THE RELATION BETWEEN MASSFLOW AND INERTIA OF A SYSTEM

In the initial investigation done on the possibility of measuring the MF on a horizontal conveyor belt (HCB) (Dierks[2]) the basic assumption was made, that the inertia of the CB, and thus the inertia driven by the motor, is proportional to the MF on the HCB. To prove this assumption consider the HCB itself as shown in the figure below:

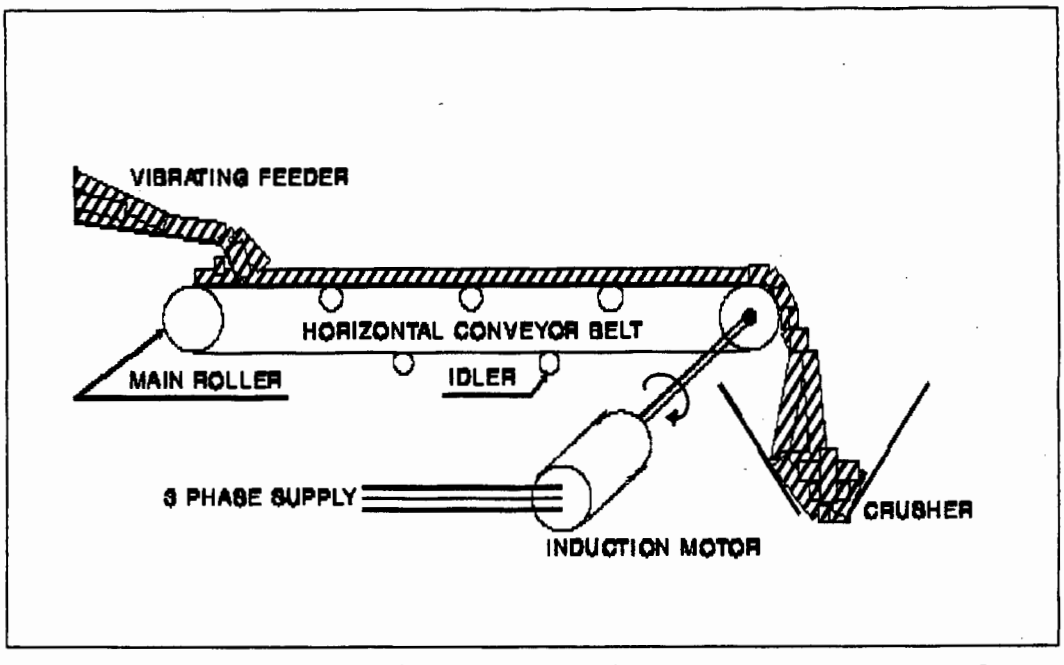


Figure 2.1: Schematic of a horizontal conveyor belt

The HCB consists of two main rollers at each end. The shaft of one roller is connected to the driving motor. This motor is normally a 3-phase induction motor. The actual belt runs over these two rollers but is also supported in between by idlers. The top part of the belt is supported by more idlers as this portion is carrying the weight of the material to be transported. It can be assumed that both the idlers and the bearings of the main rollers impose a load force on the motor. It can also be assumed that the material on the belt is not distributed uniformly.

The power on the shaft of the induction motor is the product of torque T and angular velocity ω . At the main roller a conversion between the rotational and the translational frame of reference has to be considered. Hence the power delivered to the actual belt can be broken into force F and speed v. The following block diagram illustrates the frames of reference:

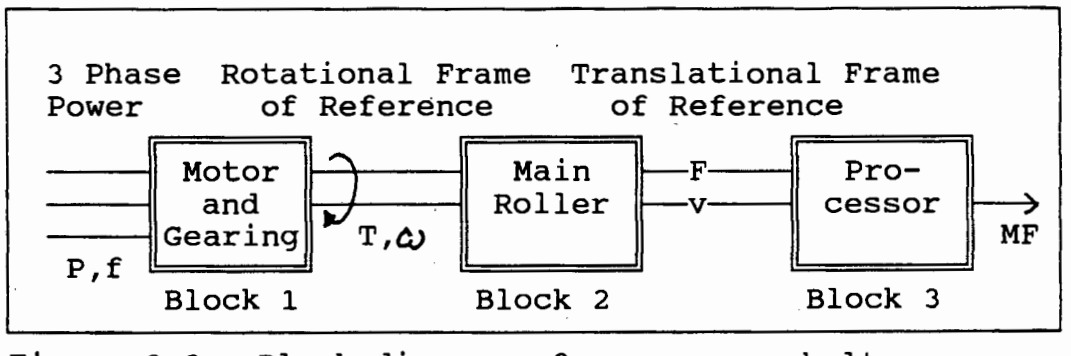


Figure 2.2: Block diagram of a conveyor belt

The model of the CB can thus be viewed in two different ways:

a) In the rotational frame of reference

b) In the translational frame of reference

Viewing the model in a particular frame of reference

means that all equations describing that model are written in terms of that frame of reference. In the above figure the translation between the rotational and translational frame of reference takes place in block 2 (Main Roller). Thus, if the model is viewed in the rotational frame of reference, block 2 is merged with block 3. Alternatively, if the model is viewed in the translational frame of reference, block 2 is merged with block 1.

Thus, there should exist an analogy between the rotational and translational frame of reference. To prove this analogy consider the power balance for a system:

$$P_{shaft} = P_{supp} * \mathcal{E}_{mot} \qquad ...(2.1)$$

where

P_{shaft} - Power delivered from the motor to the shaft

P_{supp} - Power supplied to the motor

$\mathcal{E}_{mot}$ - efficiency of the motor

$$P_{belt} = P_{shaft} \qquad ...(2.2)$$

where

P_{belt} - Power delivered to actual belt

By the Principle of Power Invariance the power delivered in the rotational frame of reference must be equal to the power delivered in the translational reference frame, i.e. no losses are considered in the conversion between the rotational and translational roller. Losses are however involved in the motor, which is why the efficiency has to be taken into account in Eq.(2.1).

The power in the rotational frame of reference can be broken down as follows:

$$P_{shaft} = T_d * \omega \qquad ...(2.3)$$

where
T_d - Driving torque on shaft

Extending by (1/r * r) it becomes:

$$P_{shaft} = T_d * 1/r * r * \omega$$

where
r - radius of the driving roller

By definition $F = T / r$ and $v = \omega * r$ and thus follows:

$$P_{shaft} = F_d * v = P_{belt} \qquad ...(2.4)$$

where
F_d - Driving force on actual belt

Thus the following analogy can be determined by comparing Eq.(2.3) and Eq.(2.4) describing the rotational and the translational frame of reference respectively:

$$T_d \quad <=> \quad F_d \qquad ...(2.5)$$

$$\omega \quad <=> \quad v \qquad ...(2.6)$$

To derive the analogy for the MF on the CB, consider the mass M on the CB alone as the MF can then easily be determined by multiplying M with the speed of the belt.

When looking at the mass the CB is viewed in the translational motion space. The force balance of the CB is shown in the figure below:

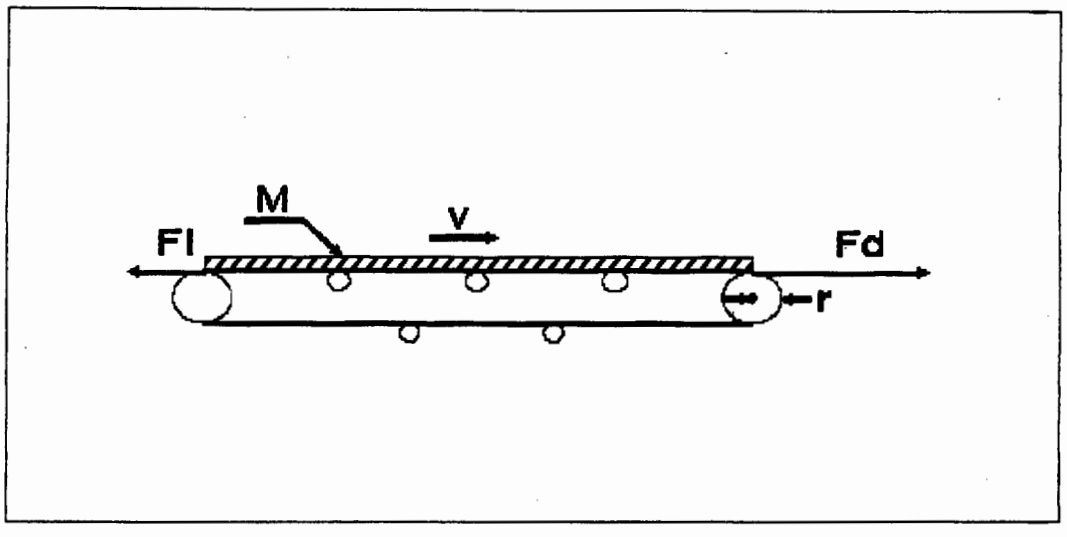


Figure 2.3: Force balance of a conveyor belt

Lets assume the mass, which is distributed along the CB, can be condensed into a point mass (i.e. a mass which is infinitely small and contains the whole weight of the mass on the belt). This mass is lying somewhere on the CB. The force balance in the schematic diagram above stays the same and thus the system is the same. The figure below shows the mass reduced to a point mass:

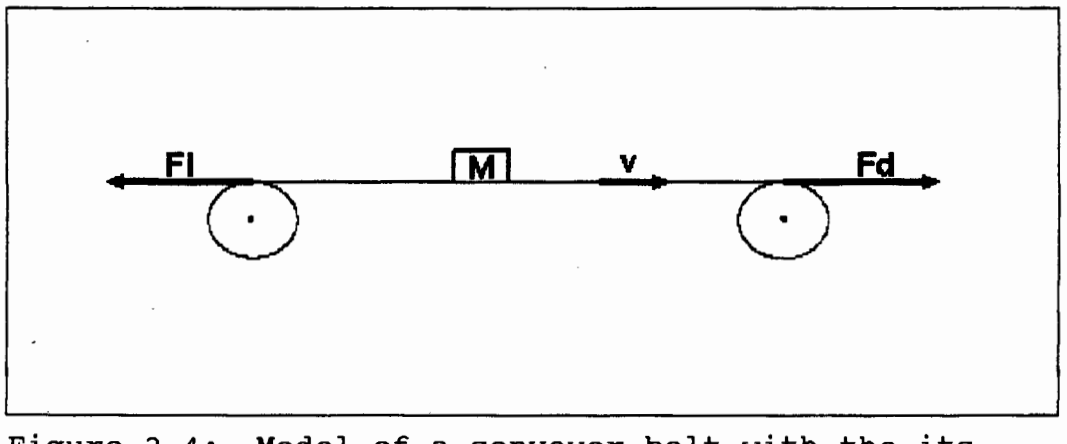


Figure 2.4: Model of a conveyor belt with the its mass reduced to a point mass

This point mass can then be moved to the right to sit

right on top of the driving roller, and the rest of the belt can be disregarded. Again if the force balance is maintained the system can be regarded as not having changed. This model is shown in the following figure:

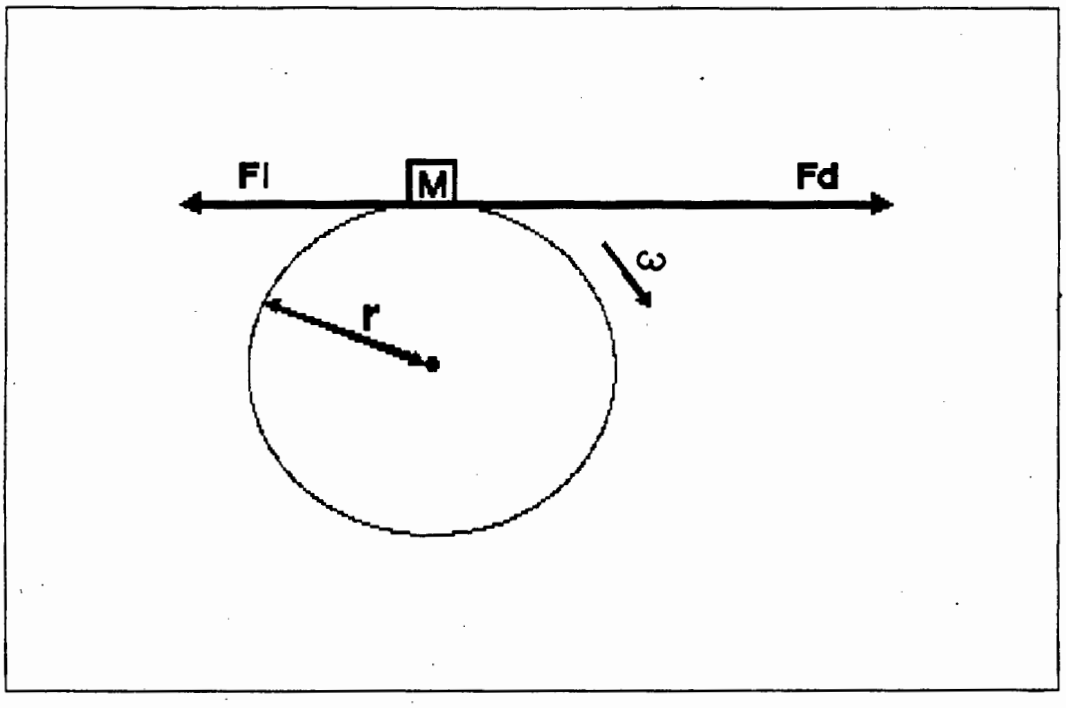


Figure 2.5: Point mass moved to driving roller

This mass M at a distance r from the centre of rotation can be described as an inertia by the following relationship (Halliday and Resnick[7]):

$$J = M * r^2 \qquad \ldots(2.7)$$

where
J - Inertia of the body

An analogy can be derived by considering the force of acceleration in the translational frame of reference:

$$F_{acc} = m * a \qquad ...(2.8)$$

where
- F_{acc} - Accelerating force
- M - Mass to be accelerated
- a - actual acceleration = dv/dt

Eq.(2.8) can therefore be rewritten as follows:

$$F_{acc} = M * dv/dt$$

$$= M * d/dt\ (v/r) * r$$

Multiplying by the radius yields:

$$F_{acc} * r = M * d/dt\ (v/r) * r^2$$

$$= T_{acc} = M * r^2 * d\omega/dt$$

Thus applying Eq.(2.5) and Eq.(2.6):

$$T_{acc} = (mr^2) * d\omega/dt \qquad ...(2.9)$$

Comparing Eq.(2.8) and Eq.(2.9), the analogy between the rotational and translational frame of reference can be extended as follows:

$$m <=> mr^2 \qquad ...(2.10)$$

The term mr^2 is defined as inertia J in the rotational reference frame (Eq.(2.7)).

Thus inertia of the system is proportional to the mass

on the CB and the MF can be determined using the following equation:

$$MF = M * v / l_{belt} \qquad ...(2.11)$$

where
- v - speed of belt
- l_{belt} - physical length of belt

The assumption used in the initial study (Dierks[2]) has hence been shown to be correct.

2.2 A TECHNIQUE TO MEASURE INERTIA OF A SYSTEM

In the previous section the relationship between the mass on a CB and the inertia of the system was derived. Hence if the inertia could be measured, the MF could be determined, too. Leonard [9] suggests the following technique to measure the inertia of a system:

A mass M is moving on a translational horizontal track in x-direction at a speed v. F_d is the driving force, whereas F_l is the load force opposing F_d. x is the distance covered within a certain period in time. The following figure illustrates this system:

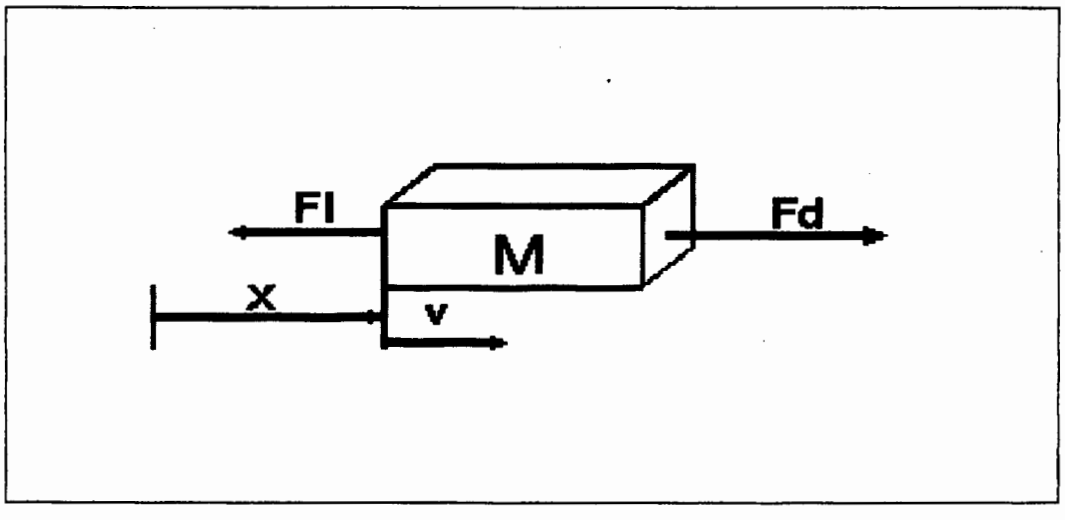


Figure 2.6: A mass moving along a horizontal track

According to Newton's law the Force Balance may be written down as:

$$F_d - F_l = d/dt\ (M*v) \qquad ...(2.12)$$

where
M*v is the mechanical momentum

As v=v(t) and M=M(t):

$$F_d - F_l = M\ dv/dt + v\ dM/dt \qquad ...(2.13)$$

If the mass on the conveyor is constant, i.e. dM/dt=0, Eq.(2.13) can be reduced as follows:

$$F_d - F_l = M\ dv/dt \qquad ...(2.14)$$

$$v = dx/dt; \Rightarrow dv/dt = d^2x/dt^2 = a \qquad ...(2.15)$$

Note that this system is similar to the system shown in figure 2.4 of the previous section. Hence the same analogies can be applied using Eq.(2.5), (2.6) and (2.10) to convert Eq.(2.14) into the rotational frame of reference:

$$T_d - T_l = J * \alpha \qquad ...(2.16)$$

where
- α - angular acceleration = $d\omega/dt = d^2\theta/dt^2$
- T_d - driving torque
- T_l - load torque

Note again that this equation represents the case of constant system inertia.

In order to measure acceleration and torque which is a difficult procedure, Leonard[9] suggests the following:

Convert the above equation to a power equation by

multiplying by ω:

$$T_d * \omega - T_l * \omega = J * d\omega/dt * \omega$$

$$P_d - P_l = J * d\omega/dt * \omega \qquad ...(2.17)$$

where
P_d - driving power
P_l - load power

The right hand side of Eq.(2.17) is equal to the power lost due to the change of kinetic energy stored in the system.

Leonard suggests that this power relation be used in order to measure the inertia of the system in what he calls a 'Run-down' test. Therefore the power supplied to the system and the angular velocity of the motor have to be measured. An advantage of the technique is that no details of the plant are required.

Before applying the actual Run-down test the relationship of $T_l = f(\omega)$ has to be established. This can be done under steady state conditions where

$$P_l = P_d \qquad ...(2.18)$$

as $d\omega/dt = 0$ - i.e. the system is running at constant speed

This implies that:

$$T_l = P_l/\omega \qquad ...(2.20)$$

If steady state power is measured for various speeds

the function of torque with respect to speed can be established.

Note that it is practical to measure the power supplied to the motor, i.e. P_{supp}, instead of the driving torque. This is due to the fact that measuring the torque would require a special adaptation of the motor pedestal to accommodate the load cells needed to measure torque. This power measurement has to be corrected for the efficiency of the motor to yield a measurement for the load power P_l (Eq.(2.1)).

Details of the **Run-down test:**

The system is running at an initial speed of ω_o. At $t=t_o$ the power to the motor is switched off and the plant is left undisturbed. As the plant decelerates due to the load torque of the system, the speed should be recorded as a function of time.

Due to P_d equalling zero if the power is switched off, J can be determined as follows:

$$J = \frac{-P_l}{d\omega/dt * \omega} = \frac{-T_l}{d\omega/dt} \qquad ...(2.21)$$

At any instance in time t_1 the following terms can be determined from the speed response:

a) the speed $\omega_1 = \omega(t_1)$ of the system from the speed response
b) $\alpha_1 = d\omega(t_1)/dt$ of the system also from the speed response
c) $T_1 = T_l(\omega_1)$ from the load torque characteristics determined earlier

These three terms can then be entered into Eq.(2.21)

to yield an estimate for the system's inertia.

In the initial study (Dierks[2]) a relationship between inertia and the angular acceleration was established experimentally. This relationship showed that the inertia of the system is inversely proportional to the deceleration of the system, which can be confirmed by Eq.(2.21). No mention was however made about the effect of load torque on this relationship.

2.3 THE TECHNIQUE'S ANALOGY IN THE TRANSLATIONAL FRAME OF REFERENCE

The aim of this study is to determine a technique in order to measure MF, which is simply the mass M on the belt multiplied by the speed v of the belt. As the inertia in a rotational frame of reference is analogous to mass in the translational frame of reference, the technique described in the previous section can be converted from the rotational into the translational frame of reference.

Applying these analogies Eq.(2.21) becomes:

$$M = \frac{-F_l}{dv/dt} \qquad \ldots(2.22)$$

The technique to estimate the mass is similar to that of measuring inertia of the system. Thus, the steady state characteristics between the load force F_l and speed v have to be determined first. This can be done by measuring the power delivered to the driving motor. Taking η_{mot} into account a reading for the driving power of the motor is obtained. Thus the load force can be determined by dividing this power reading by the speed of the belt.

Then a 'Run-down' test can be applied by switching the power to the CB off and recording the speed response. This response provides at any time $t = t_o$ the speed, deceleration and load force from the load force characteristics. Substituting these values into Eq.(2.22) will yield an estimate for the mass on the CB.

There are, however, a few shortcomings to be mentioned, which do not make it suitable for an application where the mass is measured continually in an on-line fashion:

a) Both measurements for power and speed are contaminated by noise in an industrial environment. This means that the instantaneous measurements of speed and power are not accurate. As the speed differential is needed for Eq.(2.21) the higher frequency components have to be accounted for, as differentiation only attenuates low frequency components.

b) The load characteristics are difficult to determine as they change continually. External influences, which cannot be modelled a long time before, play a large role.

c) The load characteristics cannot be assumed to be constant, because the friction of the system depends on the speed of the belt.

d) The technique proposed by Leonard[9] is designed for a one-off test, i.e. a test that is applied once. It needs to be decided how this technique can be adapted to serve as an on-line "mass estimator" in an environment where the mass has to be estimated continually.

e) In the Run-down test the system is stopped completely. This is not always desirable in practice, as this would mean a cut in production for the period that the CB is stopped.

f) The efficiency of the motor has to be known as the input power to the motor has to be adjusted

in order to get a value for the output power of the motor.

2.4 SUGGESTIONS TO MAKE THE RUN-DOWN TEST SUITABLE FOR ON-LINE MEASUREMENTS

2.4.1 Using the Energy Balance rather than the Power Balance

The energy equilibrium can be obtained by integrating the power within a defined period of $t_0 <= t <= t_1$:

$$E_{10} = \int_{t_o}^{t_1} P(t)\, dt \qquad ...(2.23)$$

The power balance as used in the previous section can be written down for the translational frame of reference by using the analogies in section 2.1:

$$P_d - P_1 = M * dv/dt * v \qquad ...(2.24)$$

For which the time integration will yield:

$$\int P_d\, dt - \int P_1\, dt = \int M * v * \frac{dv}{dt}\, dt$$

Assuming constant mass M and according to $P_1 = F_1\, v(t)$ this becomes:

$$\int P_d\, dt - F_1 \int v\, dt = M \int v\, dv$$

Hence the integrals reduce to:

$$\int P_d \, dt - F_1 * x = 1/2 \, M \, v^2 \quad ...(2.25)$$

where
x - distance covered by the CB in a certain period of time

Imposing time limits for any time period $t_0 <= t <= t_1$ on the above integral results in:

$$\int_{t_0}^{t_1} P_d \, dt - F_1 * x_{10} = 1/2 \, M \, (v_1^2 - v_0^2) \quad ...(2.26)$$

where
$x_{10} = x_1 - x_0$

The term on the right hand side of the equation designates the energy converted into kinetic energy, which only has an effect if there is a change in speed. During steady state this term equals zero.

As P_d equals zero in a Run-down test, only the speed response needs to be recorded. Solving Eq.(2.26) for M yields:

$$M = \frac{-2 * F_1 * x_{10}}{(v_1^2 - v_0^2)}$$

As v_1 equals zero it follows:

$$M = \frac{2 * F_1 * x_{10}}{v_0^2} \qquad ...(2.27)$$

The advantage of using the energy model rather than the power model is that the power and speed signal are effectively filtered by an ideal filter in that integration can be viewed as a filter with infinite time constant. If the noise on the signals can be assumed to be of a Gaussian nature (i.e. with zero mean), this noise is eliminated in this process of integration.

Instantaneous values for speed are not needed anymore, but averages. The speed differential is also not required anymore. Hence any high frequency components on the signal have less effect.

Also note that the dynamics of the speed response are not required, as only the initial and the final speed needs to be known. The speed signal will only be used to determine the distance x_{10} covered

2.4.2 Determining the load force before each Rundown Test

Assume the following data logging strategy: Power and speed are logged from t_0 to t_2. At t_1 where $t_0<=t_1<=t_2$ the signal to stop the motor is given. For t_0 = 0s, t_1 = 1.0s and t_2 = 2.5s, a response could look as follows:

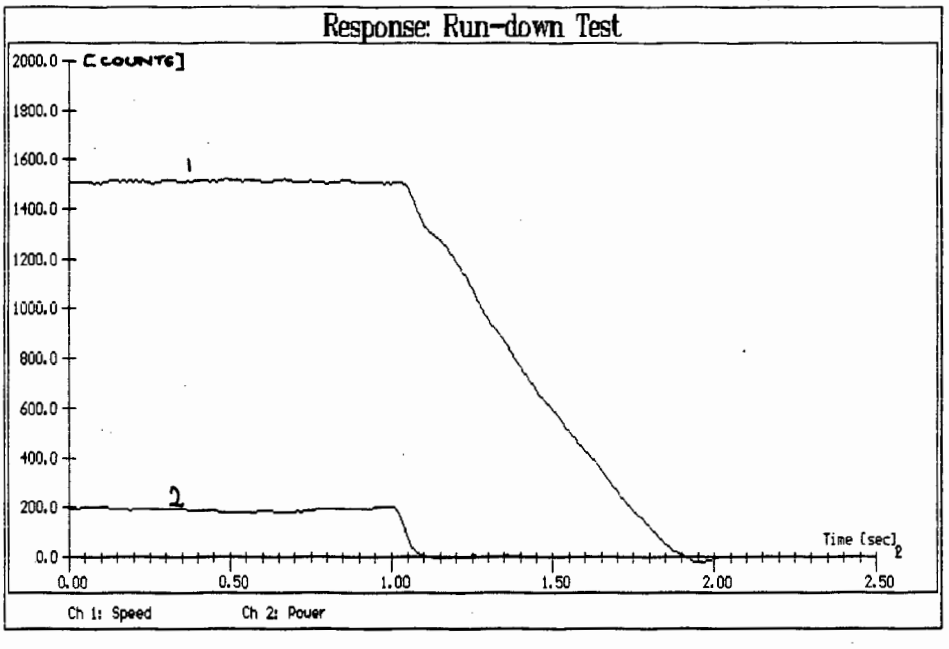


Figure 2.7: Data logging strategy to determine the load force before each test

In the above diagram the time span of $t_0<=t<=t_1$ could be used to determine the load force of the system in the following way:

If the system is made to run at steady state, as can be seen apart from the noise present in the above response for $t_0<=t<=t_1$, the kinetic energy component in Eq.(2.26) can be assumed as non existent.

Solving for F_1 yields:

$$F_1 = \frac{\int_{t0}^{t1} P_d \, dt}{x_{10}} \qquad \ldots(2.28)$$

Thus the steady state period $t_0<=t<=t_1$ can be used to determine the load force of the system in the actual test environment. This overcomes the disadvantage mentioned in section 2.3, that the load characteristics cannot be known or modelled a long time before as they change continually.

Using the energy equation rather than the power equation also has the additional advantage that a perfectly filtered signal is used, as all the noise is averaged out during the process of integration. Thus the values obtained can be assumed accurate.

2.4.3 Determining the mass from step up responses

As this MF measuring technique should be an on-line technique, i.e. a technique which is applied while the system is running, the system has to be accelerated again straight after it has been stopped.

Eq.(2.26) can also be used to determine the mass on the CB from the step-up response. The difference to the Run-down test is that now the power supplied to the system is not equal to zero, but a value much higher than the normal steady state value, as the system has to be accelerated. This higher power needed is due to the required kinetic energy to accelerate the system. As mentioned earlier this kinetic energy is proportional to the mass of the system. If the step-up response takes place during the time span $t_3<=t<=t_4$, Eq.(2.26) can be written down as follows:

$$M = \frac{\int_{t3}^{t4} P_d \, dt \; - \; F_1 * x_{43}}{1/2 * (v_0^2 - v_1^2)}$$

where

v_0- Steady State speed in unperturbed mode, i.e. final speed

v_1- speed in perturbed mode, i.e. initial speed

If the system is started from $v_1 = 0$:

$$M = \frac{\int_{t3}^{t4} P_d \, dt \; - \; F_1 * x_{43}}{1/2 * v_0^2} \qquad \ldots(2.29)$$

Therefore measuring the power and the speed to the system, the mass on the CB can be determined in the following way:

a) Determine the distance covered by the CB during the acceleration, i.e. the time integral of speed for $t_3 <= t <= t_4$. v_0 can be determined at steady state.

b) Determine F_1 either from the force characteristics or from a similar procedure described in the previous section.

c) Determine the integral of the power supplied to the system.

Substituting all these values into Eq.(2.29) will yield an estimate for the mass on the CB from the step-up response.

2.4.4 Adapting the system so that the Conveyor Belt needs only be slowed down.

As mentioned earlier it is not always desirable that the system is stopped, as this results in a cut in production.

A possible solution to this problem would be to slow it down by a certain percentage of full speed. This however cannot be done with a normal induction motor alone, as the speed of an induction motor is proportional to the frequency of the supply to the motor. Hence a frequency inverter is needed, where the output frequency and thus the supply frequency to the motor can be controlled. In a standard Variable Speed Drive (VSD) ,i.e. a frequency inverter, the input voltage control signal is proportional to the output frequency.

It should be noted that the cost of a commercially available VSD could make this technique uneconomical, because the price of a VSD equals that of a nuclear weightometer. Thus the option of only slowing the CB down for the purpose of measuring the MF is restricted to plants where VSDs are already installed for other purposes.

If this control signal is perturbed by a certain voltage the frequency on the output of the VSD will change and thus the motor speed will change as well. In the same way, if this voltage perturbation is removed, the system will return to its original speed.

In an actual test situation a negative voltage

can be added to the frequency setpoint signal in such a way to reduce this signal. Hence the VSD will slow the system down. As soon as the negative voltage is removed the frequency setpoint will return to the original level and thus the system will accelerate again.

The equations to determine the mass on the CB would then have to be adapted as follows:

v_0 is the initial steady state speed, v_1 is the perturbed steady state speed in such a way that $v_1 < v_0$. The time between t_1 and t_0 is the time during which the step-down test takes place, i.e. the step-down signal to the VSD is applied at t_1. In the same way the time between t_3 and t_4 is the time during which the step-up test takes place, i.e. the step-up signal is applied at t_3.

Thus the equations to determine the mass on a CB read as follows

- for the step-down or Run-down situation:

$$M = \frac{-2 * F_1 * x_{21}}{(v_1^2 - v_0^2)} \qquad \ldots(2.30)$$

- for the step-up response:

$$M = \frac{\int_{t3}^{t4} P_d \, dt - F_1 * x_{43}}{1/2 * (v_0^2 - v_1^2)} \qquad \ldots(2.31)$$

2.5 INCORPORATING ALL SUGGESTIONS IN A SINGLE TEST PROCEDURE

Incorporating all the suggestions mentioned in section 2.4 the following test strategy is proposed to overcome various shortcomings of the technique mentioned in section 2.3:

The power and speed responses have to be logged for a period of t_{40}, i.e. from t_0 to t_4. The following events which take place during the time of t_{40} are worth noting:

$t_0 <= t <= t_1$: System is kept at steady state in the unperturbed mode. This cycle is used to determine the load force of the system.

$t = t_1$: Step-down signal is applied to the input of the VSD to slow it down.

$t_1 <= t <= t_2$: Run-down or step-down test takes place. The speed response can be used to determine the mass on the CB.

$t_2 <= t <= t_3$: Steady state in the perturbed mode. This cycle is needed to determine the average speed of the system in the perturbed mode.

$t = t_3$: Step-up signal is applied to the input of the VSD to speed it up again.

$t_3 <= t <= t_4$: Step-up test takes place. From the speed and power response can the mass on the CB be determined.

$t = t_4$: End of test

A typical response of power and speed could look as follows, where t_0 = 0s, t_1 = 1s, t_2 = 2.5s, t_3 = 3.5s and t_4 = 5s:

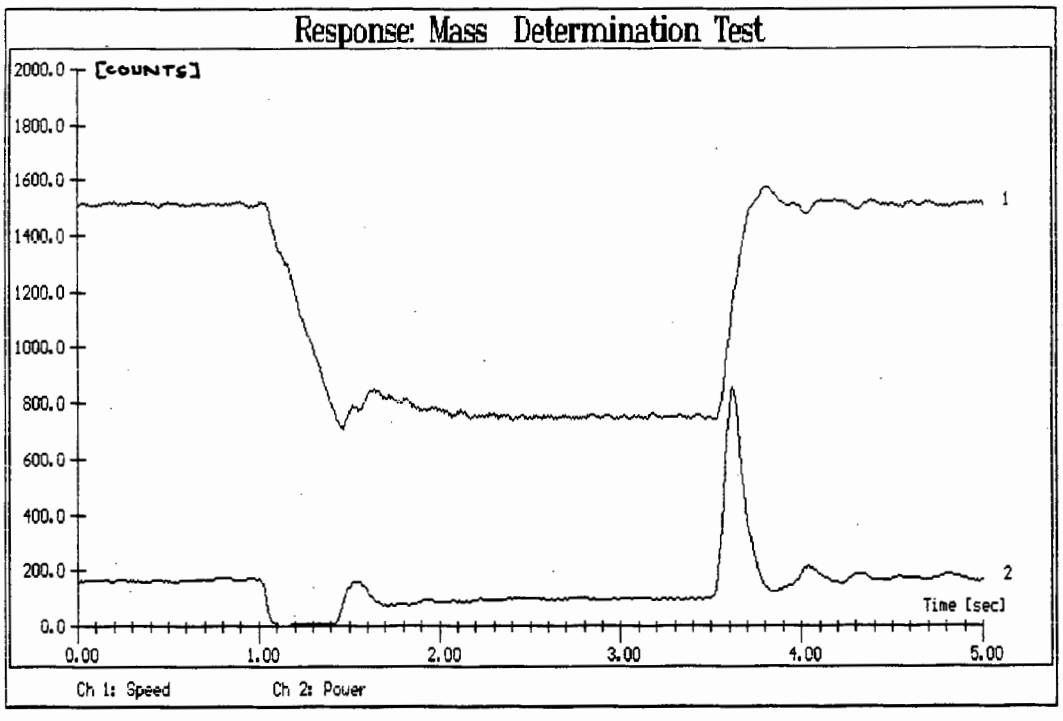


Figure 2. 8: A typical test response of power and speed

The test can thus be summarized in the following flow diagram:

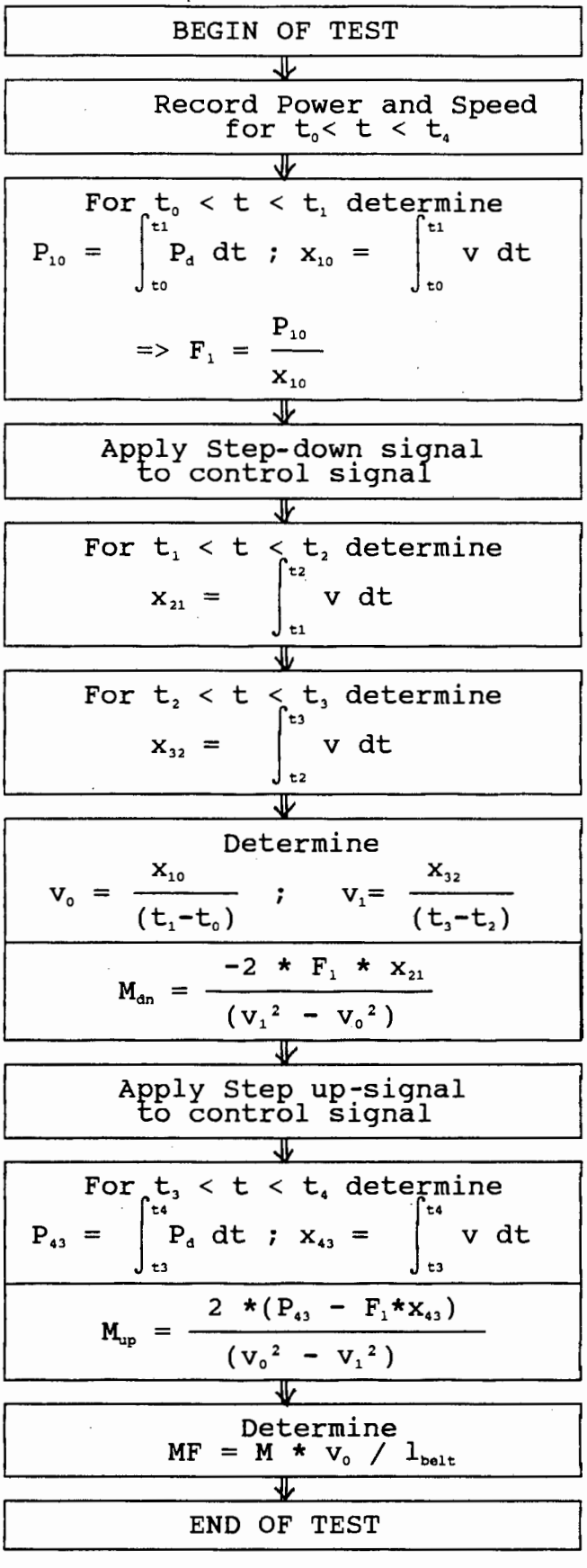


Figure 2.1: Flow diagram describing a proposed test cycle

When a full 100% perturbation is applied note that the run-down and step-up test are just a special case of the more general case, where the system is only slowed down.

2.6 SHORTCOMINGS OF THE TEST PROCEDURE

2.6.1 Modelling the load characteristics with respect to speed

Up till now it has been assumed in all equations that the load characteristics are constant with respect to speed. This can, however, not be assumed in practise as the load characteristics depend both on the speed and the load of the mass on the belt. Bearings, which impose a load force on the system, have a non-linear relationship with speed, as the load force is bigger at low speeds. This force will increase with increased mass on the CB as the mass exerts an additional pressure on the idlers, thereby increasing their drag.

Thus the load friction F_l is not constant, but a function of speed v and mass M:

$$F_l = f(M,v)$$

This load function can be described by a Taylor expansion, which is a higher order polynomial. The following polynomial will approximate F_l:

$$F_l = F_0 + a*v + b*v^2 + c*M + d*M^2$$

Determining the above relationship would involve running the system for different speeds at constant loading of mass, and running the system for different masses at constant speed in steady state. From these tests the coefficients of the

above equation can be found by means of a multi-linear regression.

Determining the above relationship would thus involve extensive test work on each conveyor that this system is going to be installed on. This is not practical in terms of the objectives of this study (i.e a low-cost MFM that can easily be installed on existing CBs). Hence it has to be determined if the gain in accuracy due to the force being modelled in terms of mass and speed is justifiable. This gain can be determined on an experimental basis.

The algorithm incorporating the load force characteristics is described in Appendix A.

2.6.2 Incorporating the efficiency of the motor

In section 2.2 it has been suggested that measuring the power supplied to the motor by means of a power transducer is the most practical way to determine the power delivered by the motor. If the power supplied to the motor is measured, all the power measurements used in chapter 2.5, which are assumed to be measurements of power as delivered by the motor, have to be adjusted for the efficiency of the motor. The efficiency thus has to be determined somehow.

It is, however, virtually impossible to determine the efficiency of a motor if the input and output power are not measured accurately. The efficiency of a motor depends on the speed (or slip) of the motor which in turn depends on the load torque of the motor.

Typical efficiency characteristics and load characteristics (Say[14]) are shown in the following figures:

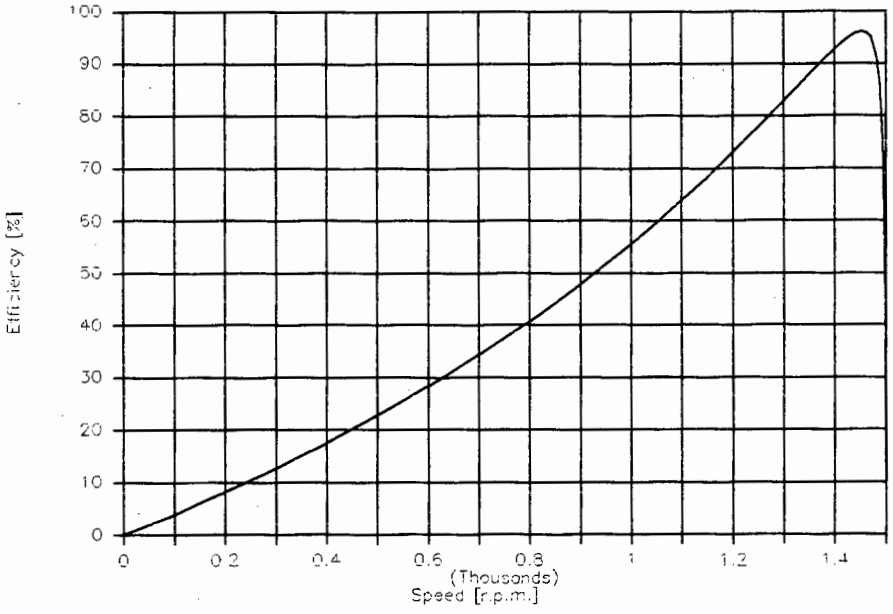


Figure 2.9: Efficiency characteristics of an induction motor

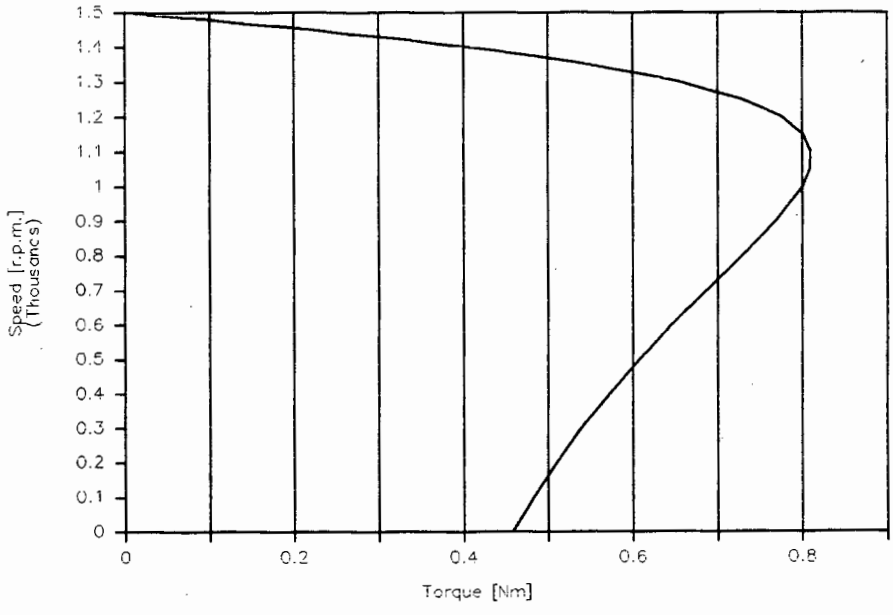


Figure 2.10: Load characteristics of an induction motor

The above characteristics can be determined and modelled by an equivalent circuit. The parameters of the equivalent circuit can be determined by performing a short and open

circuit test on the motor. Even if the parameters of the equivalent circuit are known for a test situation, they cannot be assumed constant with time as they change due to many other influences, like for instance the actual temperature of the motor.

As the efficiency is difficult to determine and because the initial test required to determine the parameter of the equivalent circuit is lengthy, it is therefore not desirable in the light of the objectives for a low-cost MFM, to model the efficiency. Hence it has to be assumed constant. This assumption is acceptable if the motor is loaded with more than 20% of its full load power, as can be seen in the following figure showing the efficiency load characteristics of a motor:

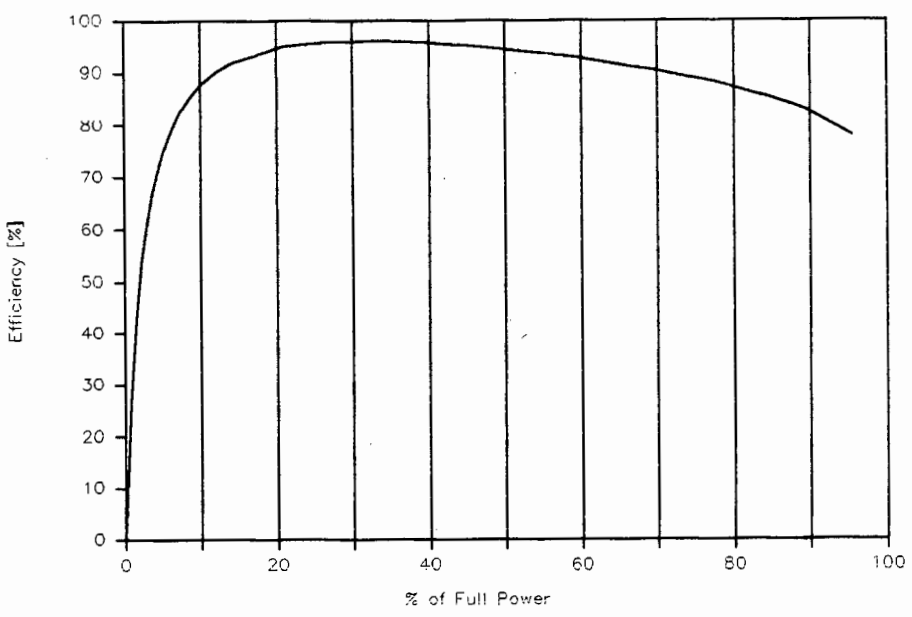


Figure 2.11: Efficiency load characteristics of an induction motor

Other ways to determine the power delivered by

the motor include systems that use load cells on the chassis of the motor. These techniques have also been ruled out as they would require a special adaptation of each driver motor that the system is installed on, which also is not desirable in terms of the objectives of this study.

2.6.3 Measuring the Speed of the Motor

The speed of the motor can easily be measured using a DC Tacho that is mounted on the shaft of the motor. This however means that a special assembly to mount the tacho has to be provided, which is not advantageous in terms of the objectives of this study.

It thus has to be determined if the measurement of speed is essential, as the transient response of speed is not required. Only the steady state values of initial and final speed need to be known, which could be determined from the frequency setpoint signal for example.

CHAPTER 3 :
BASIC STATISTICS AND CALIBRATION

3.1 INTRODUCTION

Using the technique from the last chapter, an estimate for the MF on the CB can be obtained. A decision about the performance of the measurement system has to be made, preferably on a quantitative basis, to determine the accuracy of the measurement. Both Considine[1] and Doebelin [4] are referred to in this chapter.

When studying the performance of a measuring device the ability to measure the desired input and the ability to reject spurious inputs should be investigated. The process of calibration does just this. It concentrates on these two main features:

a) It provides the means to correct the measured readings should they be incorrect;

b) It quantifies the amount of error on the measured signal due to the spurious inputs.

By 'Input' is meant the actual quantity to be measured or the 'true value'. By 'output' is meant the result from the measuring device. The output can also be called the 'unknown quantity' in this case, as it is unknown how 'true' this value is.

Considine [1] suggests that one way to calibrate a measurement device is to compare the measured quantity to a standard of the same physical nature. A proposed design for a measurement device could be compared to another instrument (called the reference instrument) measuring the same quantity, for example.

It has to be noted that the reading of this reference instrument is still not the same as the true value. The true value should normally be regarded as unknown, as a perfectly exact definition of the physical quantity to be measured is impossible. The expression 'true value' thus refers to a measurement obtained by the 'exemplar method' (Churchill Eisenhart / Doebelin[4]). An exemplar method is one, which is agreed on by experts as being sufficiently accurate for the purpose of calibration.

If a measurement device is calibrated against a reference instrument, as suggested above, the reading obtained from this reference instrument also has an uncertainty associated with it, even if this is very small. This is why a device which is calibrated against a second reference device can never be calibrated to an accuracy higher than that of the reference instrument.

After calibration this reference device is not available anymore. To be able still to make a judgement about the accuracy of the measurement device, this error has to be defined and quantified. To do this the source of the error has to be investigated. Then various mathematical models can be used to assign a value to the error.

Thus the error can be defined as follows: The error of a single reading is generally defined as 'the

deviation of the stated value from the true value' Considine[1]. Thus the difference between a true and a measured value is equal to the error of the system. To define the error for a whole measurement system, the individual errors have to be seen in a statistical environment. The error therefore defines the bounds between which a certain number of readings will lie.

Generally, the sources of error and in turn the error characteristics can be divided into two main areas, i.e. the static and the dynamic characteristics of the instrument.

3.2 THE STATIC CHARACTERISTICS OF AN INSTRUMENT

3.2.1 Static Calibration

To be able to make a judgement on the static characteristics of the instrument a 'static calibration' has to be performed on the system. In this calibration test the output of the measurement device will be compared to the true value over a range of input values.

The static calibration involves initially determining the input/output relationship of the instrument, i.e the relationship of the measurement with regard to the true value. It has to be kept in mind, however, that an instrument not only has one input, which is the input of the physical quantity to be measured, but many other inputs which also have an effect to a lesser degree on the output. Many different physical causes can be uncovered each having a different degree of effect on the output. For instance an input of this kind could be ambient temperature.

These kinds of interaction on the output by other inputs have to be accounted for. Firstly, if the input/output relation of the measured value to the true value is determined, the other inputs not investigated have to be kept at a constant level for the duration of the calibration test. Secondly, a family of input/output relationships for these other influences should be determined.

A measurement system can, however, never be totally isolated from its environment. Thus many

external influences acting on a system cannot be modelled by an Input/Output relationship and have therefore to be left uncontrolled. If, however, a certain input has a big influence on the system an attempt should be made to control it as well.

These input/output relations can then be used to determine the accuracy of the system.

3.2.2 Bias and Imprecision

The loss of accuracy of an instrument due to the error of an instrument is due to a combination of bias and imprecision.

Bias is a systematic error, i.e. an error that is constantly repeated. This error has constant magnitude and sign. As this error is always present it can be removed by calibrating an instrument correctly.

Bias could, for instance, be caused by a constant error being made all the time in the measurement process. Such an error could be a wrong physical constant, which is assumed.

Imprecision is a random error associated with the measurement. This error has neither constant magnitude nor constant sign. Due to the random nature of this error, this type of error is not constantly repeated and thus it is impossible to quantify or even model the exact characteristics of this error.

As mentioned earlier the measuring system can never be totally isolated from its environment. This results in a number of external influences that are impossible to account for, either because they are so difficult to control or because the influence caused is so small that it can be neglected on its own. But if all these minute influences are added up, as they would in a normal instrument, they begin to have an effect as manifested by noise on the signal. As no prediction can be made about the behavior of this noise it can be regarded as random.

Generating a set of data will result in a random scatter, which can be described by statistical methods.

3.2.3 Removing Bias

First consider the repeated measurement of the same true value:

As bias is a consistent error, it can be assumed on an infinite scale, that the average of all the random errors equals zero. In other words, if the average is taken from a set of measurement data all the errors due to imprecision should be filtered out. This cancellation of the imprecision is due to the statistical nature of the random error.

The Input/Output relationship can be described as follows:

$$y = x + b_i + e_i \qquad \ldots(3.1)$$

where
- y - actual measurement
- x - value of the true measurement
- b_i - bias of the measurement
- e_i - error due to imprecision

Thus taking the average of the set of measurements the error due to imprecision is removed:

$$\frac{\sum_{i=1}^{n} y}{n} = \bar{y} = x + b_i \qquad \ldots(3.2)$$

Where
- $\bar{y}$ - mean of the measured value
- n - no of readings

Thus the bias can be determined as follows:

$$b_i = \bar{y} - x \qquad \ldots(3.3)$$

The bias once determined, can be removed from all future readings, by subtracting it from the actual measurement.

The above equation assumes that the same true value is measured over and over again. For an actual instrument, however, the true value of input changes over a certain range of values, which has a change in the output of the measurement device as a result. Thus consider the measurement over a range of true values.

Obviously the Input/Output relationship is no longer a single correspondence between two values but a set of correspondences. If these values are plotted against each other on what is called a correlation graph, they should form a straight line if a direct relationship exists between the input and the output.

What has been the average of a set of data before, becomes now the "average" of the line through all these data points. The "average" line is more commonly known as the 'best fit' of a linear equation through a set of data. Such a best fit can be obtained by using the Least Squares Estimation Algorithm (Volk([16]).

The equation of this best fitting linear function can be described by the following

equation:

$$y = m\,x + b \qquad \ldots(3.4)$$

where
- y - measured quantity
- x - true value of quantity at a certain point
- m - slope of the best fit linear
- b - y-intercept of the best fit linear

The slope and the y-intercept of the best fit can be obtained by applying the measurement data to the following equations (Volk[16]):

$$m = \frac{n * \Sigma(y*x) - \Sigma y * \Sigma x}{n * \Sigma x^2 - (\Sigma x)^2} \qquad \ldots(3.5)$$

$$b = \frac{\Sigma y * (\Sigma x)^2 - \Sigma(x*y) * \Sigma x}{n * \Sigma x^2 - (\Sigma x)^2} \qquad \ldots(3.6)$$

The bias in this case is removed from the measured data by using a rearranged form of Eq.(3.4):

$$x = (y - b) / m \qquad \ldots(3.7)$$

Thus, apart from quantifying the amount of imprecision the process of calibration constitutes removing the bias from measured readings. A properly calibrated system thus should have no error due to bias associated with it. This is however very difficult to achieve in practice as the actual measurement systems

change with time and so does the bias.

3.2.4 Quantifying Imprecision

To develop a method to quantify imprecision, the imprecision of a measurement describing the same single true value will be considered first. It is assumed that the bias on the measurement has been removed.

In section 3.2.2 the origin of the error due to imprecision was discussed. It was found that the magnitude and sign of this error change randomly. Thus if the set of measurement data of a single true measurement is plotted on a probability plot a typical random scatter will be obtained. Due to the random nature of the data it can be assumed that the average of the random errors equal zero, i.e the average of the measurement data equals the true value.

This random scatter can be described by statistical methods using the Probability Density Function (PDF) (Doebelin[4]), and thus boundaries can be defined which describe the amount of imprecision associated with a measurement. A typical random scatter is illustrated in the following figure:

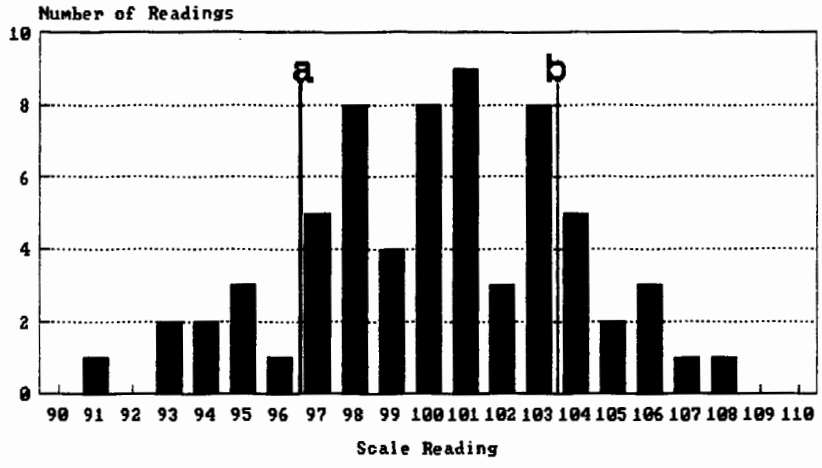


Figure 3.1: A typical random scatter

The probability that a measurement lies between boundaries a and b on the above plot equals P(a<x<b):

$$P(a<x<b) = \int_a^b f(x)\,dx = F(x) \quad ...(3.9)$$

where
f(x) - Probability Density Function (PDF)
F(x) - Cumulative distribution function

The probability that a measurement lies between the limits of plus and minus infinity equals unity as all the measurements definitely lies somewhere on the x-axis.

The most commonly known model to describe such a PDF is the Gaussian Bell curve:

$$f(x) = \frac{1}{\sqrt{2*\pi}\ *\sigma} * e^{\frac{-(x-\mu)^2}{2*\sigma^2}} \quad ...(3.10)$$

where
μ - mean value
σ - Standard Deviation (SD)

The above equation is governed by two constants. The mean value determines the position of the curve on the x-axis. The SD determines the shape of the curve. The function plotted for two different values of SD is shown in the figure below:

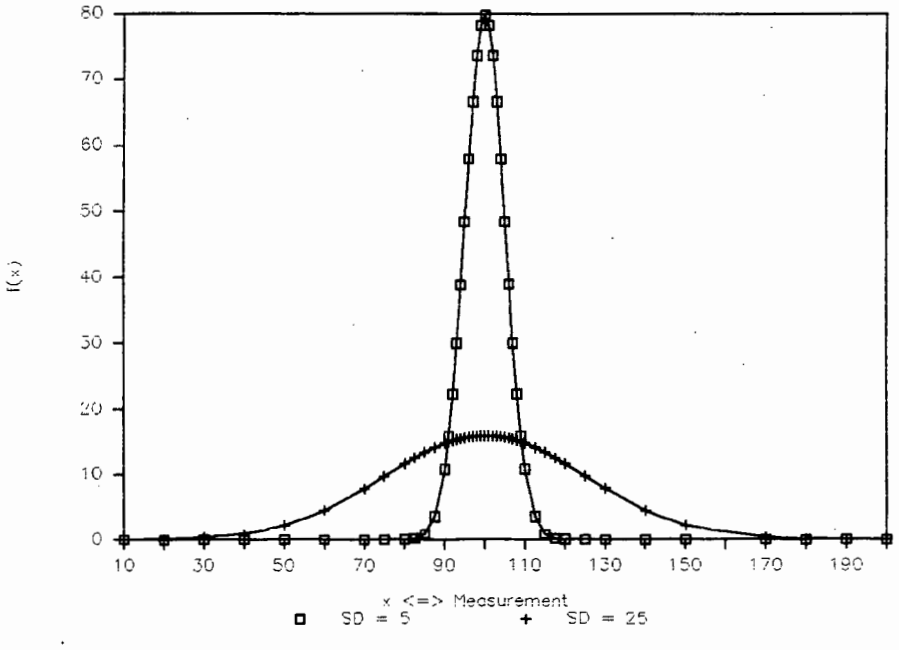


Figure 3.2: The Gaussian Bell Curve

Although the Gaussian PDF may only be an approximation it can be assumed true for an infinite number of measurements taken. Tests to prove the suitability of the model do exist, but they are lengthy to perform. In an engineering environment the Gaussian PDF can generally be assumed accurate unless discrepancies occur.

From the Gaussian distribution the SD can be defined:

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x - \bar{x})}{n-1} \qquad \ldots(3.11)$$

where

$\bar{x}$ - mean of all measurements
x - actual measurement
n - no of measurements

SD can be defined as follows: The SD describes the bounds between which certain amount of readings lie. σ is the boundary between which 68% of all readings lie. 2σ describe the boundaries between which 95% of all readings lie. The table below summarizes a few more of these statistical boundaries:

Boundary	% of readings between boundaries
0.781σ	50
1σ	68
2σ	95
3σ	99.7

Table 3.1: Boundaries of error of imprecision

The SD is the value used to determine a value for the system's error of imprecision. It is obvious that if a certain boundary is quoted it must be qualified, which type of boundary it is, i.e. if the boundary given represents the 68% or

the 99.7% limit. The difference in the value specified for the error is considerable. This qualification is however often left out in practice, which makes the estimate of error unreliable.

The equation of the standard deviation of a single true value can be extended to yield a standard deviation associated with a measurement for a set of true values.

The standard error of the measured value can hence be determined as follows:

$$\sigma_m = (1/n) * \Sigma(m*x + b - y)^2 \quad ...(3.12)$$

where
- m - slope of best fit line determined by using LSE technique
- b - y-intercept of best fit line
- y - measured value
- x - true value

The above equation, however, fixes the standard deviation for the whole range of measurements, even if a future measurement lies outside this range.

To underline this statement consider the following case: Measurements are taken over a range of $r_1 <= r <= r_2$ in an evenly distributed manner. The above equation assumes that the error is constant over this range. But what happens if a measurement near the border or even outside this range is considered. A measurement near the centre of this range is described a lot better, as it can be referred to other measurements on either side. A measurement near the border, however, can only be referred

to measurements on one side, and thus it is described in much less detail. A similar case occurs where the measurements are closely populated over a range near the centre of the scale, with only a few values describing the extremes of the range. The error near the extremes, which is much less described than the centre of the range, cannot be assumed to have the same error as the measurements near the centre. Obviously the error near the extremes should be bigger than near the centre.

A process accounting for this shortcoming is described by Miller[11]. He describes the error bound not by a linear, but by an elliptical equation.

Miller distinguishes between determining the error associated with a certain measurement and between determining the error of a future measurement. He calls these processes 'Prediction' and 'Discrimination' respectively.

In practise these two processes differ in that in prediction the true values are used to determine the error associated with a set of measurements like in the process of calibration. In discrimination the results from the process of prediction are used to determine the error and the true value from a future measurement. Prediction is thus the qualification of the unknown from the true, whereas Discrimination is the determination of the true from the unknown using the results from prediction.

In the model that Miller describes it has to be noted that the x-value is always regarded as the true value or the independent value. The y-

value is always the measured value or the dependant value. The function looks as follows, where adding or subtracting the error bound E defines the upper and lower boundary limit respectively.

$$f(x) = y = b + m*x \pm E \quad ...(3.13)$$

where
E - Error which defines boundary

$$E^2 = F * \sigma_m^2 * [1 + 1/n + \frac{(x-x_m)^2}{\sum_{j=1}^{n}(x_j-x_m)^2}] \quad ...(3.14)$$

where

F - F - distribution number [Miller(11)]

σ_m - standard error of measurement as before (Eq.(3.12))

x_m - mean value of the whole range of true values

The F-distribution number can be looked up in a F-distribution table. It is dependant on the percentage boundaries required, e.g. the boundaries that contain 95% of all readings. The F-number depends also on the number of samples used for the calibration or the prediction as Miller calls it, and on the number of future true values to be predicted (usually assumed one for a single estimation process). A copy of the F-distribution table for the 95% and 99% limits is given in Appendix B.

Thus after having determined the best fit line

through a set of data, the above equation can be used to plot the elliptical boundaries for the set of data representing a certain percentage boundary. Such a typical best fit line and the associated 95% boundaries is shown in the figure below:

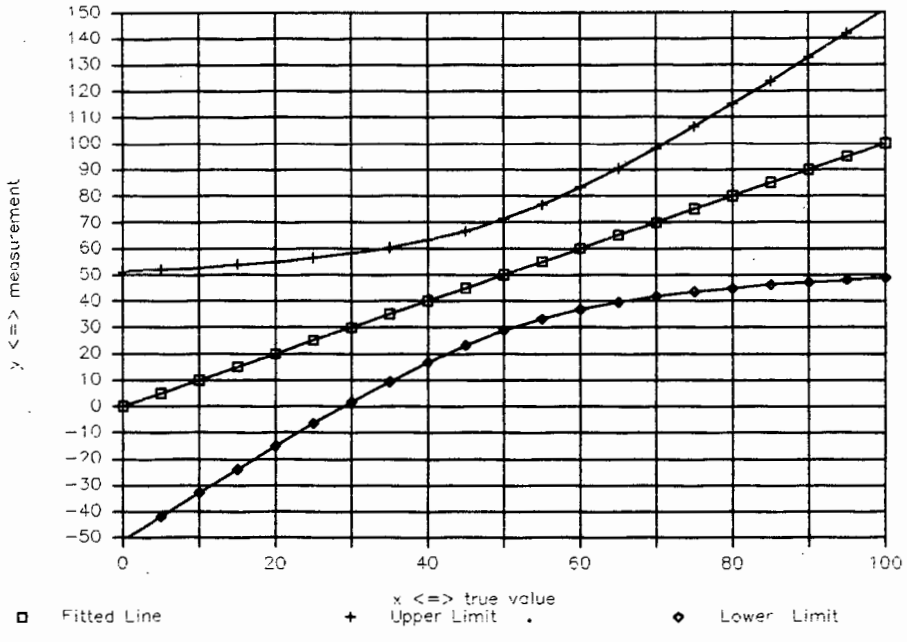


Figure 3.3: A typical best fit line and the associated 95% boundary limits

As can be seen the distance between the best fit line and one of the boundaries is the least near the centre of the range of true values and increases as one moves away from the mean. It should be noted that the standard error of a measurement is the vertical distance between the best fitted line and the boundary function.

To use the Eq.(3.14) in the process of discrimination two constants are defined, which will simplify the writing of the equation:

$$C_1 = F * \sigma_m^2 * \frac{1}{\sum_{j=1}^{n}(x_j - x_m)^2} \qquad ...(3.15)$$

$$C_2 = F * \sigma_m^2 * (1+1/n) \qquad ...(3.16)$$

Eq.(3.14) thus reduces to:

$$E^2 = C_2 + C_1 * (x-x_m)^2 \qquad ...(3.17)$$

In the process of discrimination a measurement is given. From this measurement the true value represented by the measurement and the error associated can be determined. Note that now the error between the determined true value and one of the boundaries is required. This error is equal to the horizontal distance between the best fitted line and a boundary in Figure 3.3.

To determine the actual true value and the associated error Eq.(3.13), (3.14), (3.15) and (3.16) have to be rearranged and solved for x. This is a rather lengthy process. The final result as stipulated in Miller[11] can be written down as follows:

$$x = x_m - \frac{mY \pm \sqrt{m^2Y^2 - (C_1-m^2)(C_2-Y^2)}}{(C_1-m^2)} \qquad ...(3.18)$$

where
$Y = y - y_m$
y_m - mean of all measurement data

The above equation can be split up in order to distinguish between the portion due to the true value and the portion due to the error:

$$x = \frac{y-b}{m} - \frac{C_1 * Y \pm m * \sqrt{[m^2 Y^2 - (C_1 - m^2)(C_2 - Y^2)]}}{m * (C_1 - m^2)} \qquad \ldots(3.19)$$

The first term (y-b)/m represents the true value and the rest equals the error on the system.

3.3 THE DYNAMIC CHARACTERISTICS OF AN INSTRUMENT

The dynamic characteristics are relevant when the true value to be measured changes a lot.

To determine the dynamic characteristics of an instrument the transfer function of the instrument has to be determined. The transfer function is a mathematical model, that indicates how the system will respond to a certain input.

If the output of an instrument is fast responding to changes on the input, the dynamical characteristics of the instrument have little effect on the overall characteristics of the instrument. If however the response time of the instrument is slow in comparison to the changes on the input, the dynamical characteristics have to be accommodated in the process of calibration.

The figure below illustrates a way to account for the dynamics of an instrument:

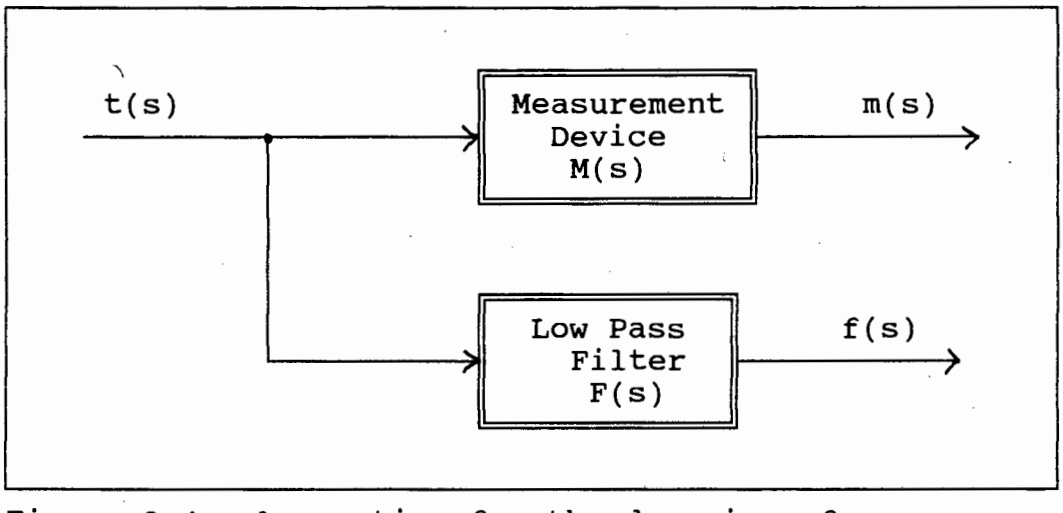


Figure 3.4: Accounting for the dynamics of a measuring device

From the above diagram it can be seen that the signal

of t(s) representing the true value to be measured can not be directly correlated to the measured signal m(s), unless:

a) The system is in steady state, i.e. t(s) is constant. In this case, if the bias of the system is removed, the input equals the output.

b) The response time of the measurement device is so fast, that its response can be regarded as instantaneous. Another way to view it, is to have a look at the pole distribution of the transfer function M(s). If the pole closest to the origin is positioned at negative infinity in the s-plane the response of such a system to a step change on the input can be regarded as instantaneous, i.e. the output follows the input.

If the above two cases are not true, the best way to solve the problem is to filter the signal of the true value t(s) with a low-pass filter F(s), which has the signal f(s) as output. If the time constant of the filter is chosen to be similar to that of the measurement device, the two output signals can be assumed to have similar dynamical characteristics and hence these two signals can be correlated against each other.

It has to be noted that often instruments are only calibrated for steady state. To do this, the system has to be insured to be at steady state when performing the calibration. Also a response time has to be specified. The response time gives an indication of how long after a step change is applied to the input, steady state is reached.

A further case when the dynamics of a system have to

be considered is, if no direct reference measurement can be made of the true value for the purpose of calibration. Consider the following case:

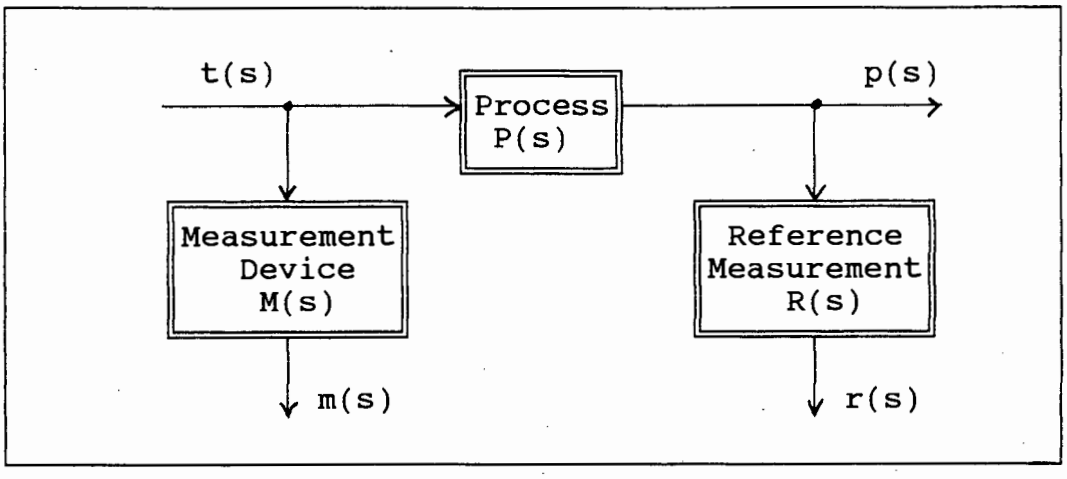


Figure 3.5: Incorporating the dynamics of a process

In the above figure it can be seen that signal t(s) can only be measured after the process of P(s), which obviously changes the dynamics of the signal. Thus the signal of the reference device r(s) cannot be directly calibrated to the signal of the measurement device m(s). One way to overcome this problem is to model the dynamics of the measurement process and the process itself and thus to account for these models by means of a low or high pass filter on the output of the reference signal in a similar way as it was done above.

If the process is however not easy to model, there is a different way to perform the calibration, which is however a crude method. A low pass filter with a time constant longer than that of any of the processes, should be put on the output of both the measurement and the reference device. The effect of these filters will override any dynamical changes on the system and thus it will filter the undesired effects due to dynamical changes. Only definite slow changing signals will be reflected on the outputs of the two filters. As the filters should be filters with the same time

constant the outputs can be directly correlated against each other or used for the purpose of calibration.

3.4 CONCLUDING REMARKS

As the field of statistics is so involved it is often neglected to some degree. As a result misconceptions exists on how to determine the error of an instrument, as the error is not clearly defined.

A typical example was confirmed by Doebelin[4], where he describes an example on how the error is quoted in the normal engineering practice. He writes that often the accuracy of an instrument is given by a single measurement, sometimes even as a percentage of full scale. It is not made clear to what the exact meaning of the single error value is, and the full scale reading is not specified either, if one wants to know the value of the actual error.

This even happens if a calibration was carried out, but the actual error is not calculated. The error is sometimes taken as being the largest deviation from the fitted line of calibration. Clearly this is somewhat unsatisfactory as this one reading used may be a freak reading and thus is definitely not representing the normal.

Manufacturers also quote percentage accuracies which often are too good to be true. Still they are not quoting an incorrect error bound, but the error bounds between which very few of the total readings lie. Another person would regard this limit as the error bounds for a high percentage of the readings, and thus is going to be disappointed very soon.

Due to these misconceptions experienced during these studies it was thought necessary to make an attempt in the previous chapter to clarify these. The techniques described will be applied in the process of

calibrating the prototype instrument.

The results are again going to be stated as a single error value. But if this value is qualified to what exactly it represents, this is acceptable.

The values given for the imprecision in this study will always represent the 95% confidence or error bands. The value stated will be percentage error of full-scale at the mean of the range of readings.

An example for the how the single value for the imprecision E_I is obtained is shown in the following figure. x_m is the mean massflow, e is the error at the mean and x_f is the full-scale reading of massflow.

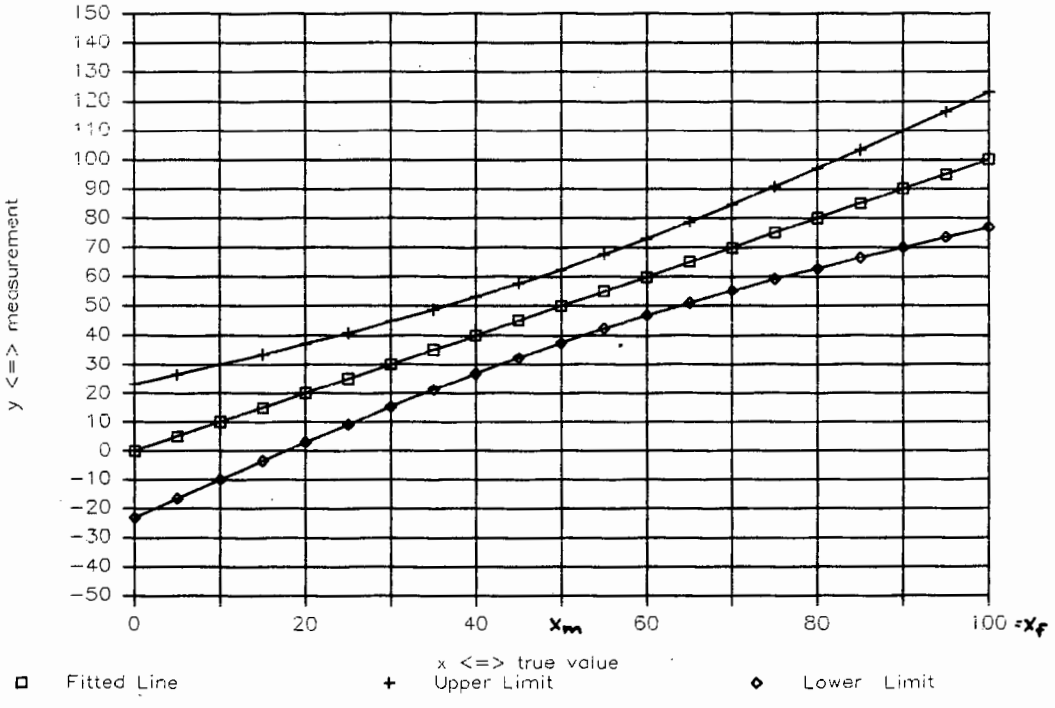


Figure 3.6: Typical correlation graph (Best fit line and error bounds)

Thus the mean error will equal:

$$e_i = e/x_f * 100\% = (13/100) * 100\% = 13\%$$

CHAPTER 4 :
THE SOFTWARE USED TO MEASURE AND CALIBRATE THE MASSFLOW ON A HORIZONTAL CONVEYOR BELT

4.1 GENERAL REMARKS ABOUT SOFTWARE

4.1.1 Basic Aims of Software

The main aim of the software packages written is to implement the MF estimation technique described in chapter 2 and the calibration techniques described in chapter 3 into suitable software packages. These packages are to serve as a prototype instrument which can be run from a Personal Computer.

4.1.2 Hardware Requirements

The programs can be run on any IBM compatible PC. The PC used in this application was a 12MHz AT machine manufactured by MIAD.

The graphics adapter used was a Hercules Graphics Adapter. The programs are however written in such a way that they can run on any conventional graphics adapter.

No special Memory is required. The standard 640kB available on most PCs is sufficient to run

the programs. The machine used in this study had a numeric co-processor (80287) on the mother board. This will only have an effect in the compilation and the speed of running of the programs. Programs that are compiled assuming that a numeric co-processor is available can not run on machines without a numeric co-processor.

The Input/Output card used was a Data translation DT2801 card. This card consists of 16 linear (8 differential) analog to digital converter channels and 2 digital to analog converter channels. The software developed to drive this card is contained in the DT2801_4.PAS sub-routine.

4.1.3 General Software Development Policies

While developing the software it was aimed to make the software as user friendly as possible. It has to be kept in mind, however, that the software developed serves only as a prototype. The emphasis was thus laid on functionality rather than user friendliness. An instrument developed for the market should not take the form of a software package implemented on P.C., but rather a microprocessor based unit.

The software was modularized as far as possible. The tasks were broken down into unit processes, which were then coded into subroutines in the Turbo Pascal Compiler Version 4.0 [15]. Turbo Pascal provides the excellent facility of units, which are subroutines, that can easily be linked and called up in any main program.

These unit processes where programmed as general

as possible, so that they could later be used in other programs without adapting them. The possibility exists that these subroutines might be used in a future instrument. This explains why no global variables where used in the programs as these would have made the subroutines more specific. If all parameters are passed to and from a subroutine it is a lot clearer what is happening in the execution of the subroutines.

In the programming process the subroutines and main programs where defined into a hierarchical structure. The main program is in the highest level coordinating all subroutines. The subroutines are part of lower levels in the structure, depending on how specialized their task is. A more specialized subroutine can only call up a more general subroutine, which is a subroutine on a lower level. The four levels of the hierarchy are shown in the following diagram:

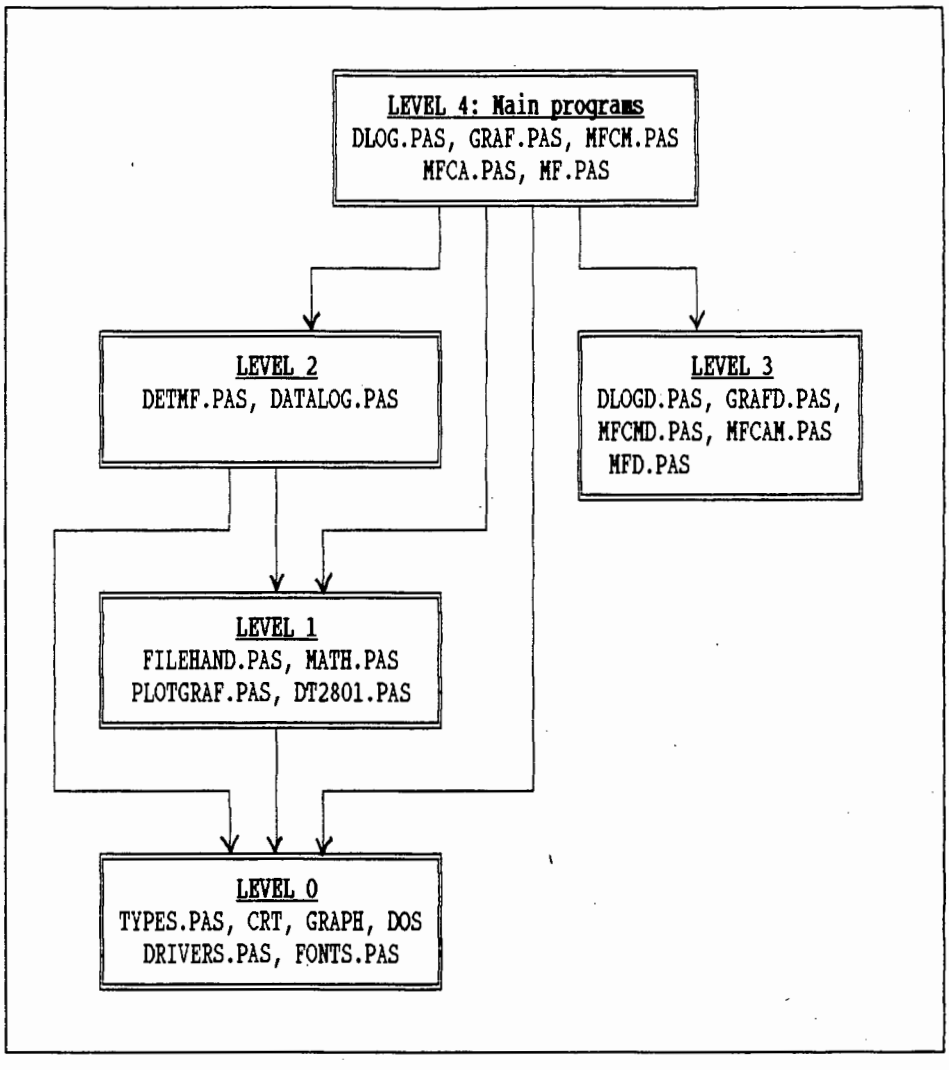


Figure 4.1: Hierarchical structure of programs and subroutines

Thus describing the levels of the hierarchy in more detail:

Level 0: This is the base of all subroutines. It only includes TYPES.PAS, which mainly defines certain data records and performs very basic functions like highlighting items on the screen, or reading an integer or a real from the screen. Subroutines which are part of the Pascal Compiler (i.e. Dos, Crt, Graph, Drivers and Fonts) can also be regarded as being part of Level 0 .

Level 1: General file handling, mathematical and graphics subroutines form part of this level. The aim of subroutines in this level is to enhance the subroutines available in the Pascal compiler.

Level 2: The subroutines in this level are designed to support specific main programs. The subroutines are tailored to perform more specific tasks and functions needed by a specific main program. As these subroutines may be needed by more than one main program, they are part of a Unit so that they can be used by all main programs.

Level 3: This level of subroutines only contains data subroutines. The reason to put the data into a separate subroutine and not into the main program is to keep lengthy data records, like screens for different menus out of the main program. This will make the structure of the main program clearer.

Level 4: This level contains the actual main program. It accesses all the subroutines in the different levels, and coordinates these accordingly.

The actual listings of all the programs and subroutines together with an index can be found in Appendix C.

Data format contains the actual samples taken for up to six channels as well as the sampling frequency, no of samples, no of channels, size of step applied to output channel and when the step was applied, legends of the channels logged and title of response. File data in this format is denoted by the file extension '*.dat'.

2) Massflow Correlation Data Format (Data B): This data format contains actual massflow samples taken from the reference instrument (channel 1) and massflow samples determined from the step-down and step-up responses of power and speed (channel 2 and 3 respectively). All samples are saved in their uncalibrated mode, i.e. the values obtained have not yet been adjusted by the calibration coefficients. Also contained in this data record are the number of samples, title of the set of samples and regression results (i.e. results if channels 2 and 3 are correlated versus channel 1). File data in this format is denoted by the file extension '*.cor'.

3) Massflow Data Format (DATA C): This data only contains the calibrated massflow data obtained from the step up response of power and speed. It also contains the time when a specific massflow sample was taken to be able to make a judgement on the average massflow. File data in this format is denoted by the file extension '*.dam'.

Examples of the exact data formats used are shown in Appendix E, whereas the definition of the data record that contain all the data is

shown in the listing of TYPES.PAS in Appendix C.

4.2 THE SOFTWARE

To explain the function of the individual software packages one basic aim has to be kept in mind:

> *The software is to obtain data either from the actual process or from disc, process this data and then store it on disc again. The processing of the data should be done to serve the basic aims of the software development outlined in Section 4.1.1.*

The function of the software is shown in a general block diagram below:

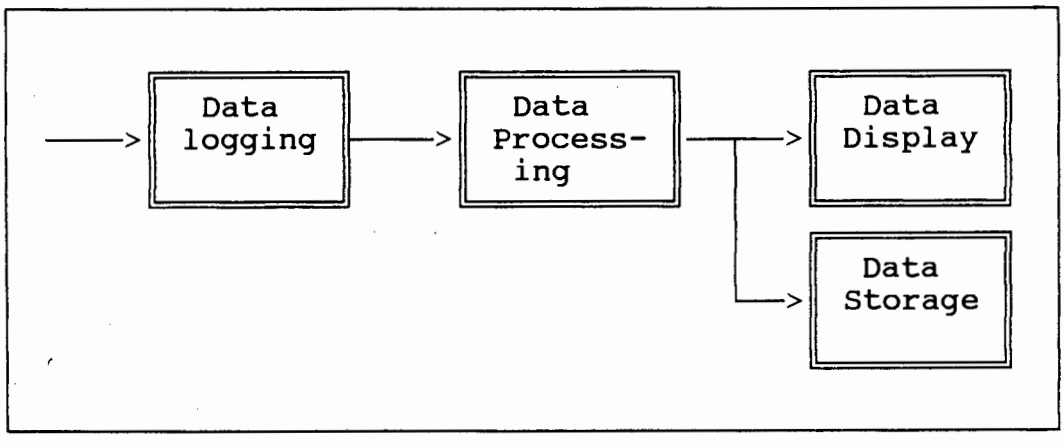


Figure 4.3: Basic function of software

To define the function of the individual unit processes of the above process the blocks in the above figure have been broken down into smaller blocks. It has to be kept in mind that not all the unit processes might be needed in the same software package, but to illustrate the relation of these unit processes to one another in the gross context they have all been included in this one diagram. Which of the unit processes is needed for which program is explained in the following sub-sections.

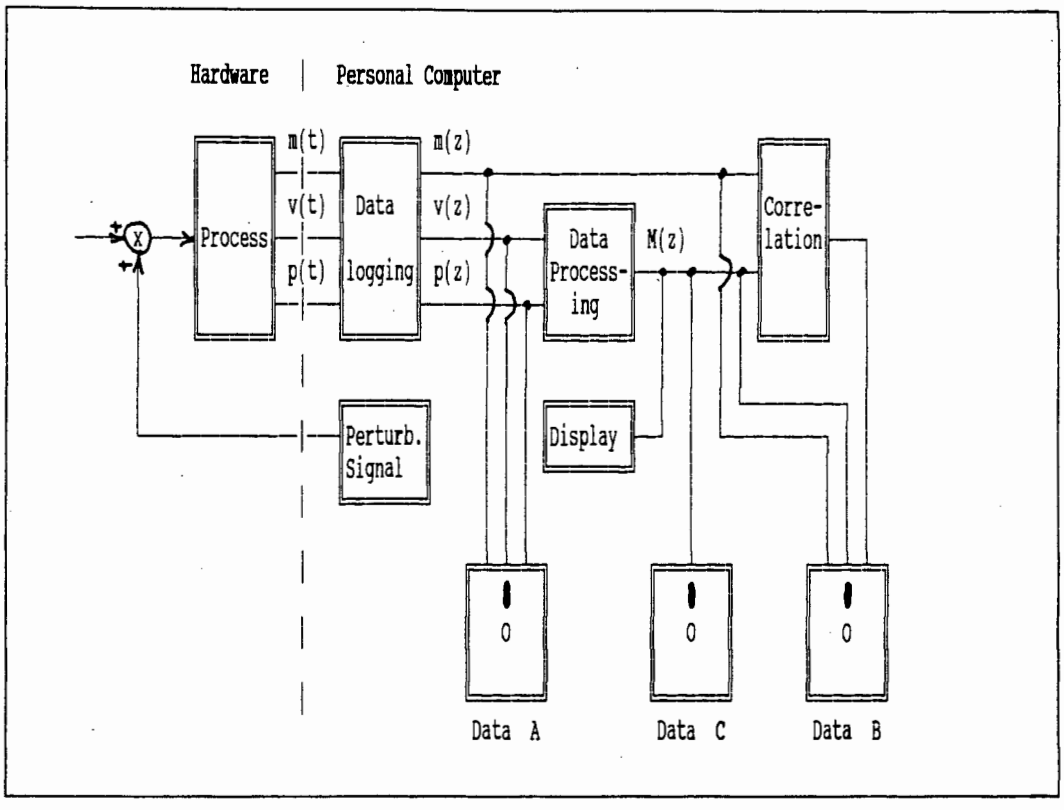


Figure 4.4: Required function structure of whole process

Three groups of software packages can thus be distinguished:

A - Manual data acquisition and correlation

B - Automatic data acquisition and correlation

C - Automatic data acquisition

The function of the programs in each of these levels are described in the following sub-sections. The actual listings of the programs and subroutines are given in Appendix C, whereas the actual menu structures and the detailed functions of the individual options in the programs are described in Appendix D.

4.2.1 Manual Data Acquisition and Correlation

The aim of this group of software packages is firstly to obtain a set of real time pulse responses of speed, power and reference massflow which are then stored onto disc. These responses are then recalled from disc one by one and thus the massflow can be determined from these responses. This technique has the advantage that the real time responses are saved, and thus they can be used to evaluate different massflow estimation techniques.

The data acquisition program is a very general data logging program (called DLOG.EXE) which sends out a pulse to perturb the system and thus logs up to six input channels. The data is saved to a file on disc in the DATA A format. The functional block diagram is shown below:

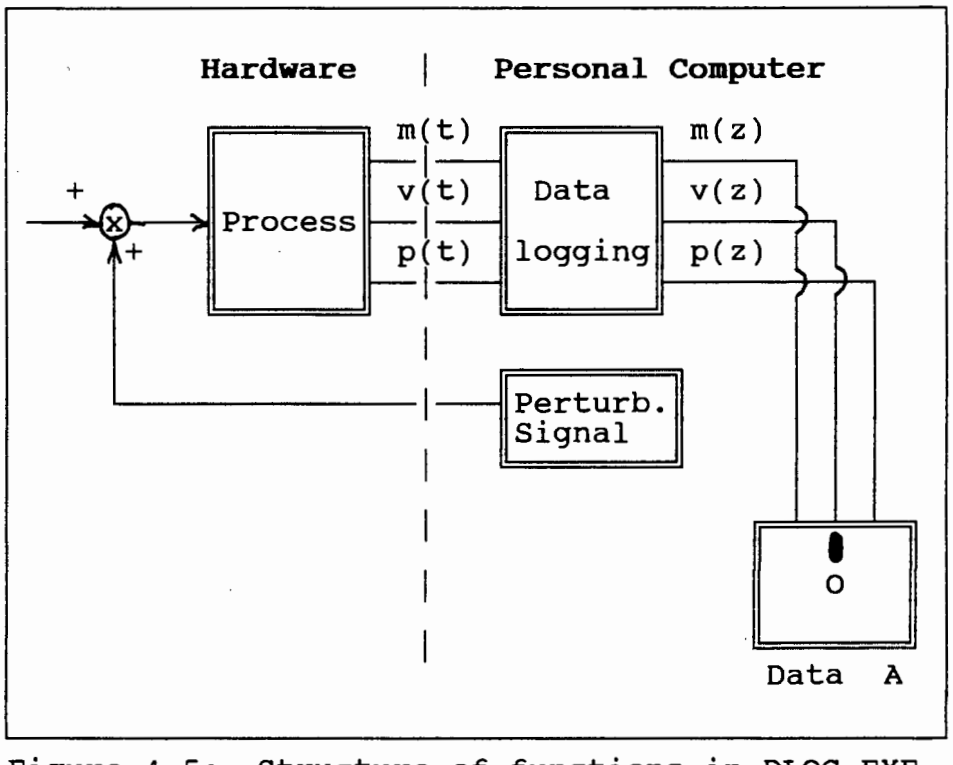


Figure 4.5: Structure of functions in DLOG.EXE

In the above diagram m, v and p refer to massflow, speed and power respectively. The independent variables t and z indicate whether the signal is in the continuous or sampled time domain.

Once the data for the responses of different massflows has been stored on disc, they can be manually called up into the data processing package (called MFCM.EXE) to process these responses to yield an estimate of the massflow in the system. This can be repeated for all the responses in a test series to yield a set of massflow data that can be used to calibrate the instrument versus a reference massflow meter. To allow for the dynamics of the instruments an option exists to filter the signal appropriately. The set of massflow samples obtained can be saved on disc using the DATA B format. The function block diagram of MFCM.EXE is shown below:

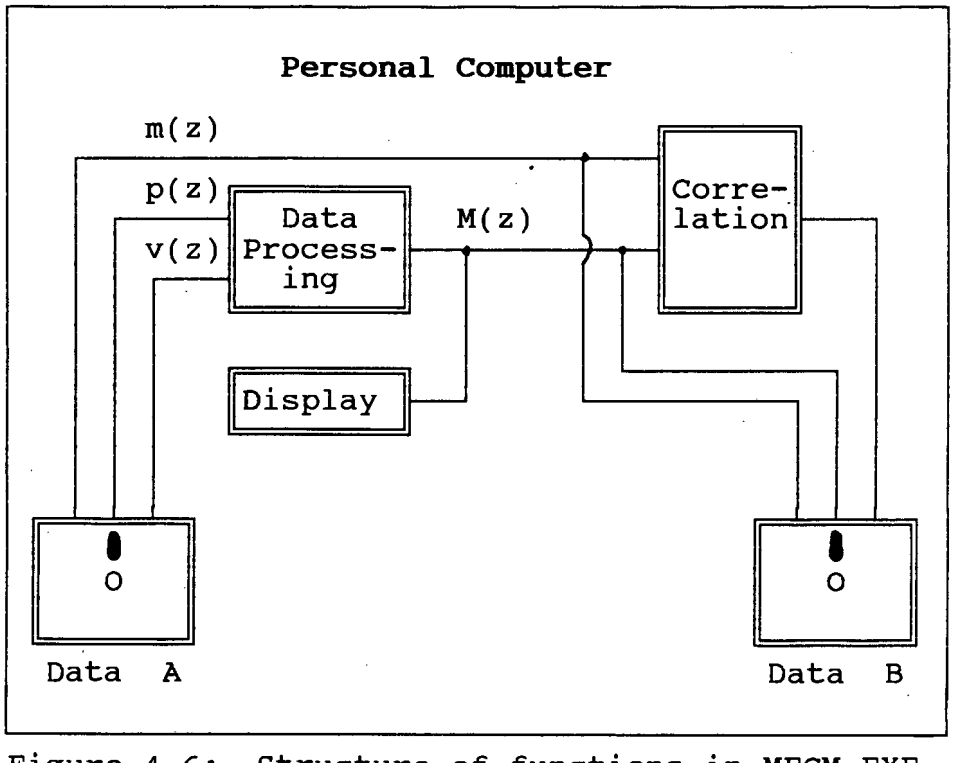


Figure 4.6: Structure of functions in MFCM.EXE

A third package that should be part of this group of programs is a package to display the real time response graphs on screen. Here not only the option of displaying one graph alone, but the option to compare various responses to one another is important. These two requirements have been incorporated into one program called GRAF.EXE (It should be noted that the units of the vertical axis of the graphs always represents binary PC counts, unless otherwise specified).

4.2.2 Automatic Data Acquisition and Correlation

This group of programs consists of one program called MFCA.PAS. The aim of this program is one step further from the group of programs in the last sub-section. Once a suitable technique to estimate the massflow on a conveyor belt has been established, the saving of real time responses to disc is not required anymore. In MFCA the response from a pulse perturbation is recorded and immediately processed to yield an estimate for the massflow on the conveyor belt. This massflow estimation is also performed in as On-Line fashion as it is performed as soon as the system has reached steady state. The massflow estimates can then be calibrated against a second reference massflow reading to prove their correlation. The massflow data samples can then be stored on disc in the DATA B format. A block diagram describing the functions of the program can be viewed below:

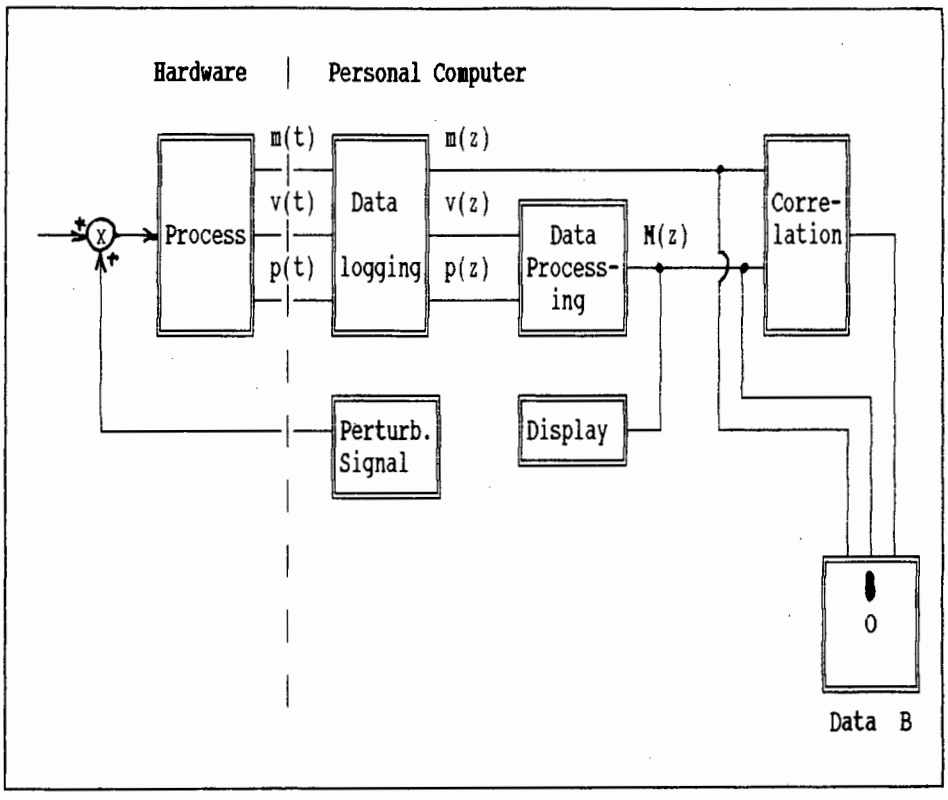


Figure 4.7: Structure of functions in MFCA.EXE

4.2.3 Automatic Data Acquisition

The aim of the third package, called MF.EXE, is to propose an algorithm that can be used in a final instrument. Thus the option of calibration is not needed anymore. Only the subroutine to log the power and speed responses of a pulse perturbation as well as the data processing routine to determine the massflow of the system are the same as before. Normally included in commercial massflow meters is the option to determine the average, total, maximum and minimum of massflow over a specified period of time. Such an option has also been included in this program. The massflow estimates obtained in this program can be stored on disc together with the time when the sample was taken in the DATA C format. The block diagram below explains the working of the program:

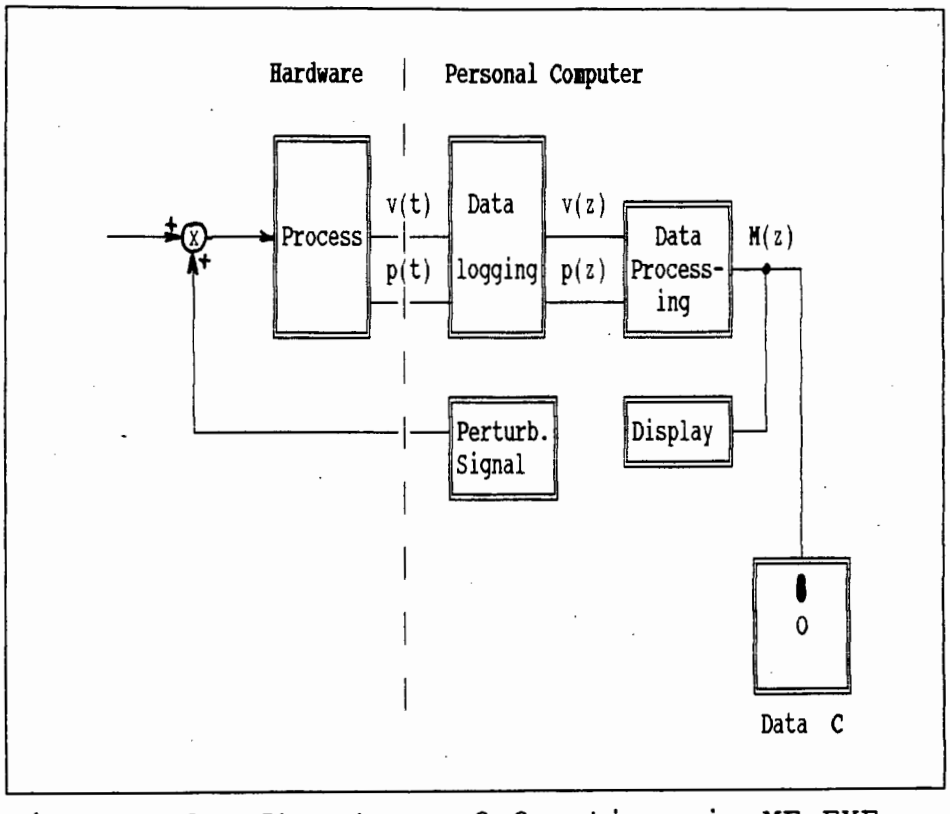


Figure 4.8: Structure of functions in MF.EXE

CHAPTER 5:
VALIDATION AND CALIBRATION OF THE MASSFLOW ESTIMATION TECHNIQUE IN A LABORATORY ENVIRONMENT

5.1 DESCRIPTION OF BASIC PLANT

The aim of the laboratory set-up at the DIAMOND RESEARCH LABORATORIES (DRL) was to have a plant which could effectively be used to measure and calibrate the MF on a HCB for different MF rates. The DRL developed a test plant which consists of three main parts:

- A Hopper bin (HB), which includes a vibrating feeder
- A horizontal conveyor belt (HCB)
- An inclined conveyor belt (ICB)

These three items are arranged as shown in the diagram below:

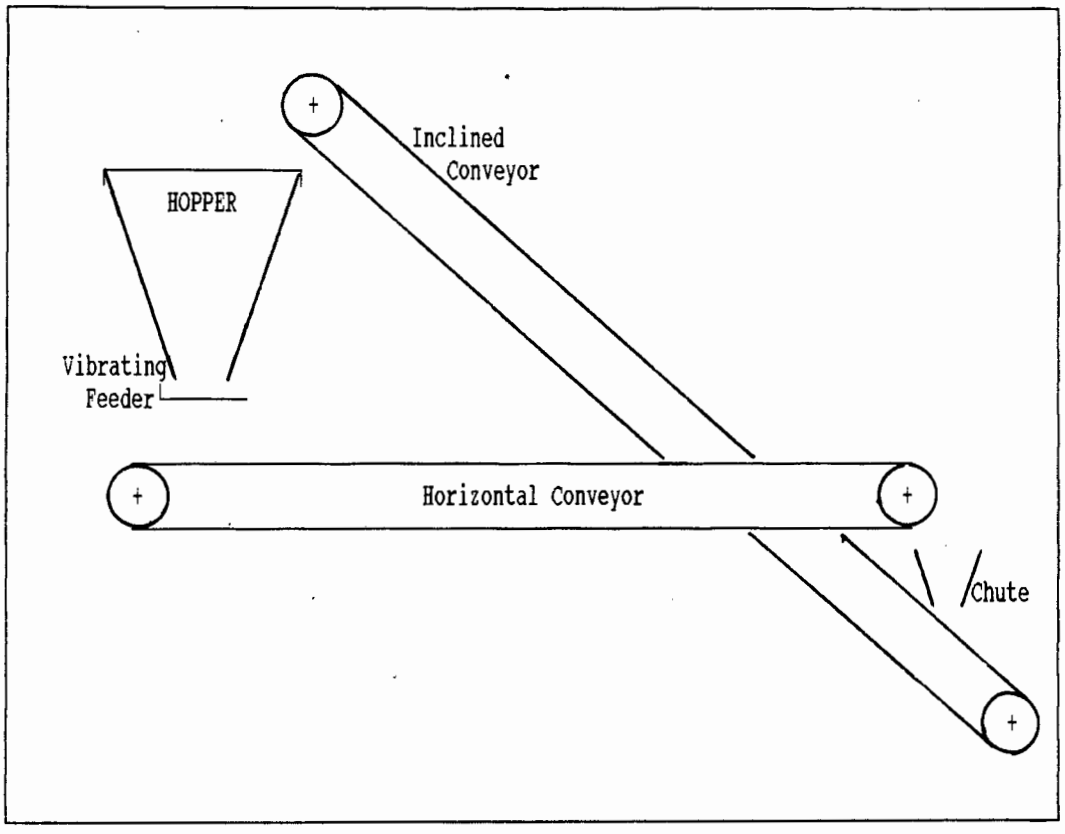
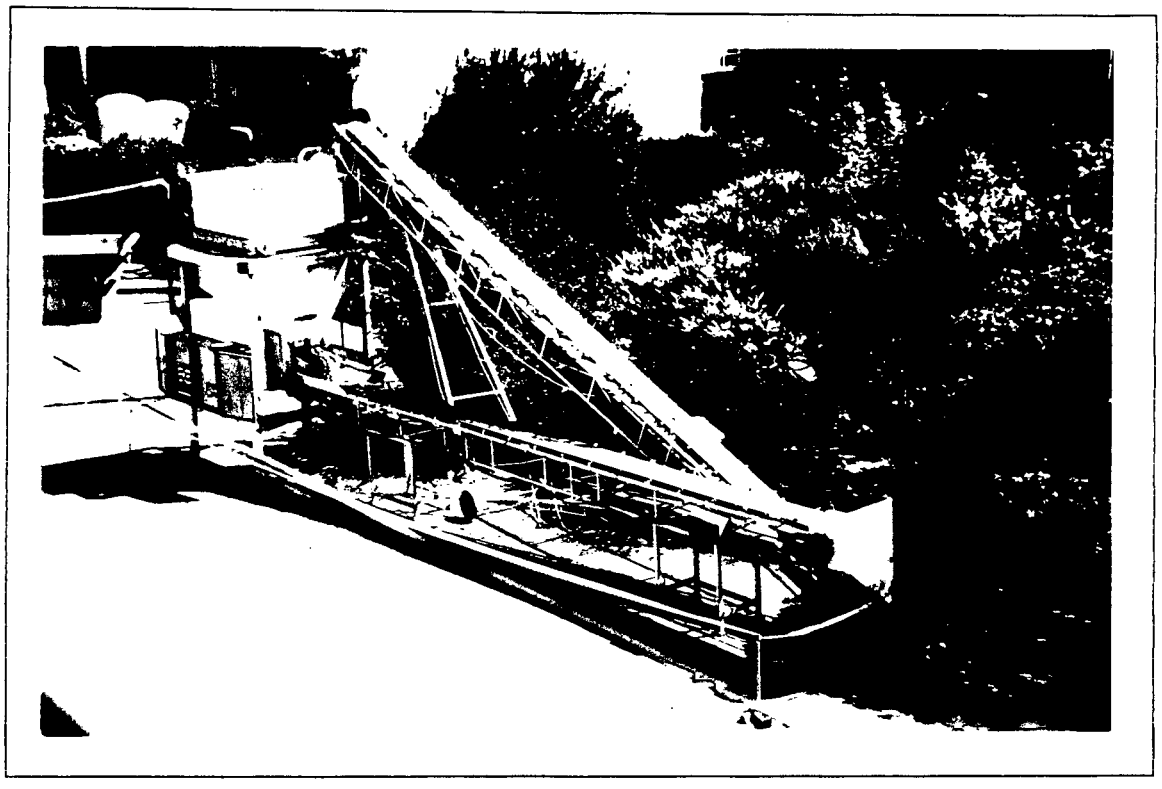


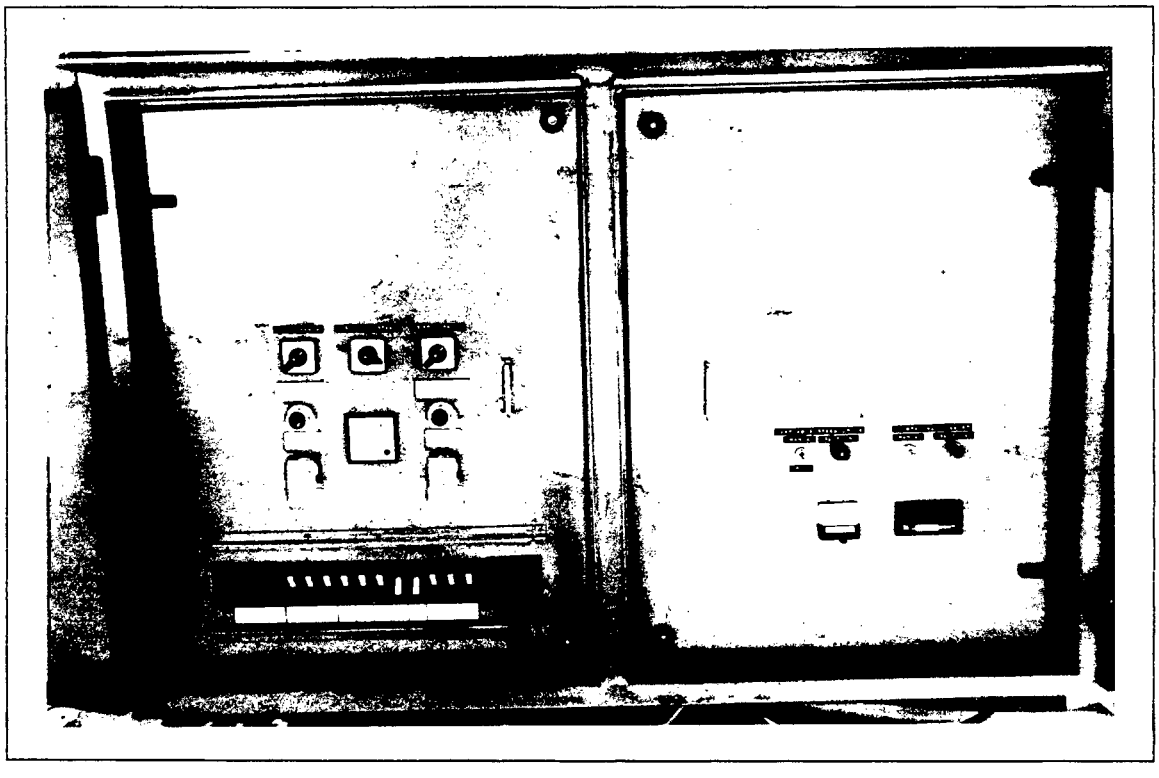
Figure 5.1: Schematic diagram of laboratory test rig

The HB acts as a storage for the gravel to be used. In a test situation the gravel would be fed via the vibrating feeder onto the HCB. From the HCB the gravel would fall through a chute onto the ICB which feeds the gravel back into the HB.

Below the test plant and the control panel are shown photographically:



Photograph 5.1: Photograph of the test plant at the DRL



Photograph 5.2: Photograph of the control panel of the test rig

The following sections will discuss the specifications

of the high voltage and instrumentation circuitry.

5.2 SPECIFICATIONS OF THE 380V CIRCUITRY

5.2.1 Horizontal Conveyor Belt

As explained in Chapter 2 an option was required to perturb the speed of the HCB. The following options were suggested:

- For plants where a VSD is already in use it was suggested to use this VSD.

- Using Solid State Relays (SSRs) to cut the power to the HCB is a more economic option for those plant applications where the use of a VSD is not justified.

Both these options have been included in the laboratory test plant. In the following diagram a circuit diagram of the 380V supply to the motor of the HCB is shown:

Two switches provide the following options:

A - The Automatic/Direct Selection Switch provides the option of supplying the HCB with 380 Volts, 50 Hertz mains supply (Direct), or to perturb the supply in a way as specified by switch B (Automatic).

B - If switch A is selected to Automatic, switch B gives the option between using the SSRs or the VSD to perturb the system.

Both the VSD and the SSR are controlled by 0..10 Volt signals. In a normal plant environment it is however preferable to use current signals (4..20mA). This has the advantage of less external influences and hence less noise on the signals. Thus the current signals that control the SSRs and the VSD (denoted by (b) and (c) respectively) are converted to a voltage signals (0..10V) using current to voltage converters.

The output of the VSD is isolated from the mains by a relay to prevent any feedback effects due to the VSD if the system was running in the direct mode.

The power transducer (PT) that is switched in series to the motor will be dealt with in the next section.

The starting and stopping circuit for the HCB is the standard circuit used to switch 3 phase loads on and off by using a 3 phase contactor.

5.2.2 Inclined Conveyor Belt

The starting and stopping circuitry of the ICB is also the standard circuit as is used for the HCB. A circuit diagram Appendix F shows the actual circuit used in this application.

5.2.2 Vibrating Feeder

The gravel is fed onto the HCB by means of a vibrating feeder. This vibrating feeder is driven by another VSD, where the frequency supplied to the vibrating feeder is proportional to the feed rate and thus the massflow of gravel. By varying the voltage control signal to this VSD the massflow could be controlled and thus the system could be tested for different MF rates. A diagram in Appendix F illustrates the circuit used in this utilization.

5.3 INSTRUMENTATION CIRCUITRY

The instrumentation circuitry consists of a system that feeds the controller signals from a PC to the test plant and to measure the variables of speed, power and reference massflow. The diagram down below shows the actual circuit diagram:

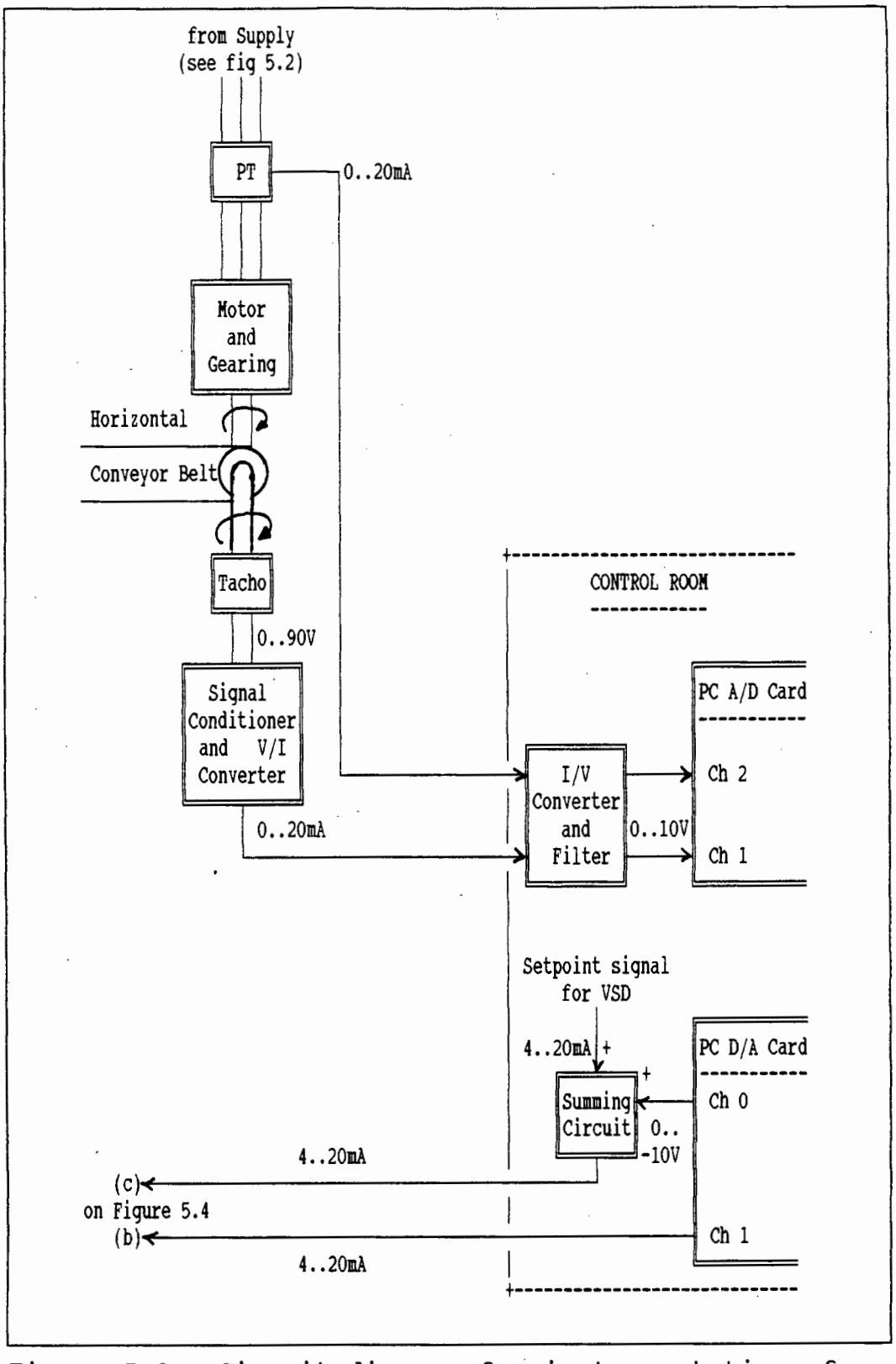


Figure 5.3: Circuit diagram for instrumentation of the horizontal conveyor belt

5.3.1 The Control Signals for the SSRs and the VSD

The control signal for the SSRs consists of generating a DC voltage (5..10V) on the output of channel 1 of the PC which is then converted into a current signal to comply with the current signal standards.

In the case of the VSD a setpoint signal normally exists already, either from some control loop that uses the speed of the CB to control some parameter (eg. level of ore in a crusher) or just from a manual setpoint control unit. In the case of the laboratory test rig the setpoint was supplied from a manual setpoint control unit. To this signal a perturbation signal from the PC was added, to slow the CB down. Adding currents is not easily performed and it was thus suggested that the current signals be converted to voltage signals which can be added with an op-amp adding circuit. The signal from the PC was already a voltage signal which thus need not be converted. The block diagram below shows how this adding of current control signals was done:

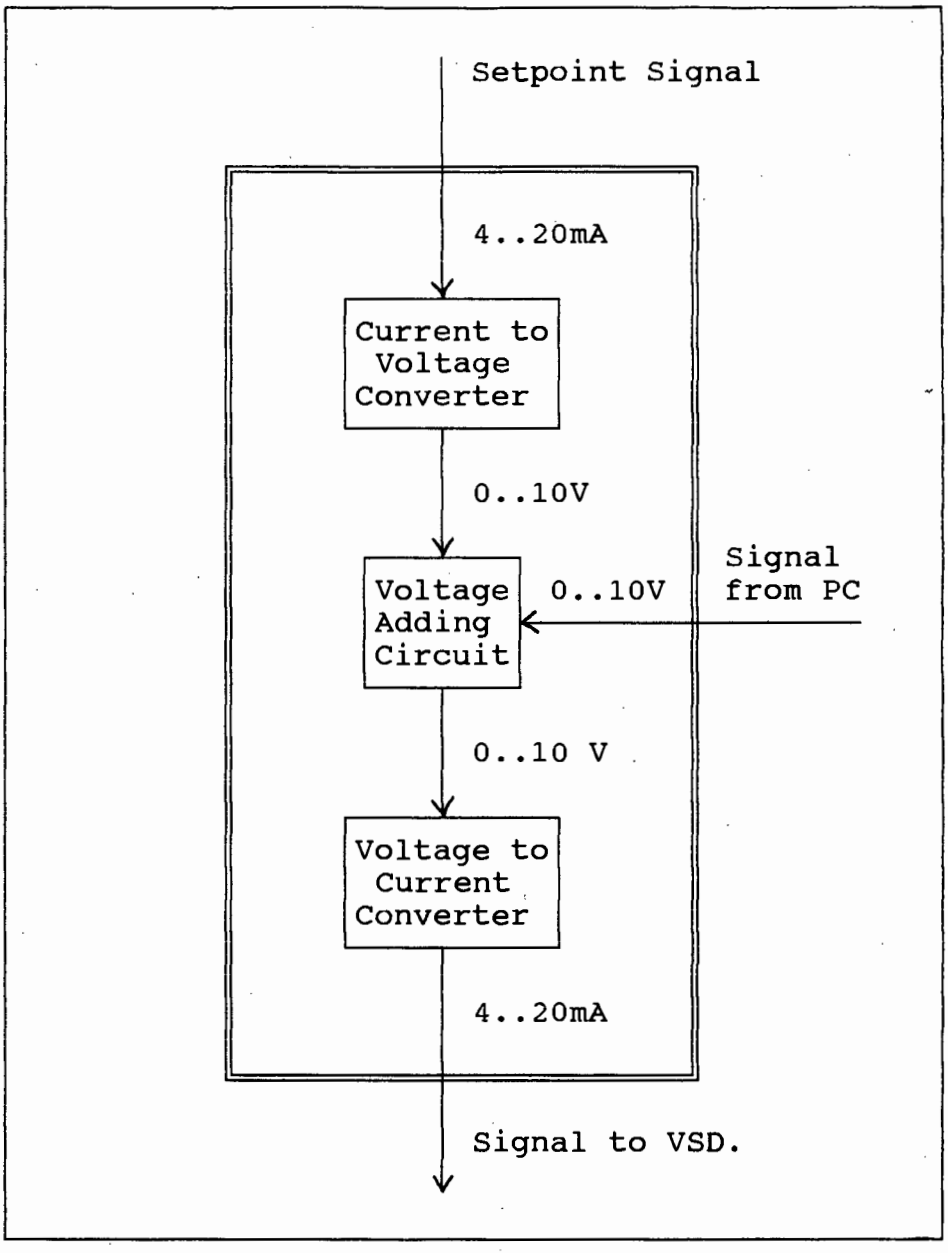


Figure 5.4: Block circuit diagram to add currents

A circuit diagram in Appendix F shows the actual op-amp summing circuit used to add two voltage signals together (Horowitz and Hill[8].

Hence if the PC generates a negative voltage on the output of channel 0, the sum of this signal and the setpoint signal is less than the original setpoint signal. Thus the CB will slow down. As soon as zero Volts is supplied by the PC the sum of the signal will return to the

original setpoint and thus the CB will return to the original speed. In this way a perturbation to the control signal can be performed easily.

5.3.2 Speed

The speed is measured with a DC Tacho that is mounted on the shaft of the motor driving the HCB. It supplies a 0..90V signal for speed varying from 0 to 1500 r.p.m. To convert this signal to a standard voltage signal it needs to be attenuated to give a 0..10V signal for a 0 to 1500 r.p.m. signal. This can be done with an attenuation circuit or by a potential divider. This signal is then converted to a current signal by means of a voltage to current converter.

Near the PC this current signal is converted back to a voltage signal, as the Analog to Digital converter in the PC can only read voltages.

To reduce the noise on the signal it needs to be filtered. A first order low-pass passive filter is sufficient to filter the high frequency noise. It also prevents aliasing which is an effect whereby high frequency noise is interpreted as a low frequency signal by the sampling circuit (EEE417 Control Engineering II Course Notes [6]). The time constant of the filter τ_f depends on the sampling frequency of the PC τ_s which in turn depends on the time constant of the plant τ_p again. In a later section τ_p and thus τ_s and τ_f are determined. The circuit used for the filter is shown in Appendix F.

Calibration of the speed signal involved determining the relationship between translational speed [m/sec] and binary counts of the PC. This was done by measuring the translational speed (by timing the revolution of the CB) and the corresponding amount of PC binary counts as determined by the A/D converter of the PC for different speeds. By plotting translational speed versus PC binary counts the calibration curve for the speed was established. The slope of this calibration graph corresponds to the conversion factor needed to convert PC binary counts to meters per second:

$$1 \text{ count} = 4.98809 * 10^{-4} \text{ m/sec}$$

The calibration graph and the results of the linear regression performed to obtain the slope of the graph is show in Appendix G.

5.3.3 Power

Measuring power can be done most economically by using a commercially available Power Transducer (PT), which measures the power of a system using the two or three Watt meter method. Two PTs were considered for this application. The first PT (PTa) uses the three Watt meter method, whereas the second PT (PTb) uses the two Watt meter method to measure power. Both units were installed and tested in a configuration shown in the following two circuit diagrams (Dover[5]).

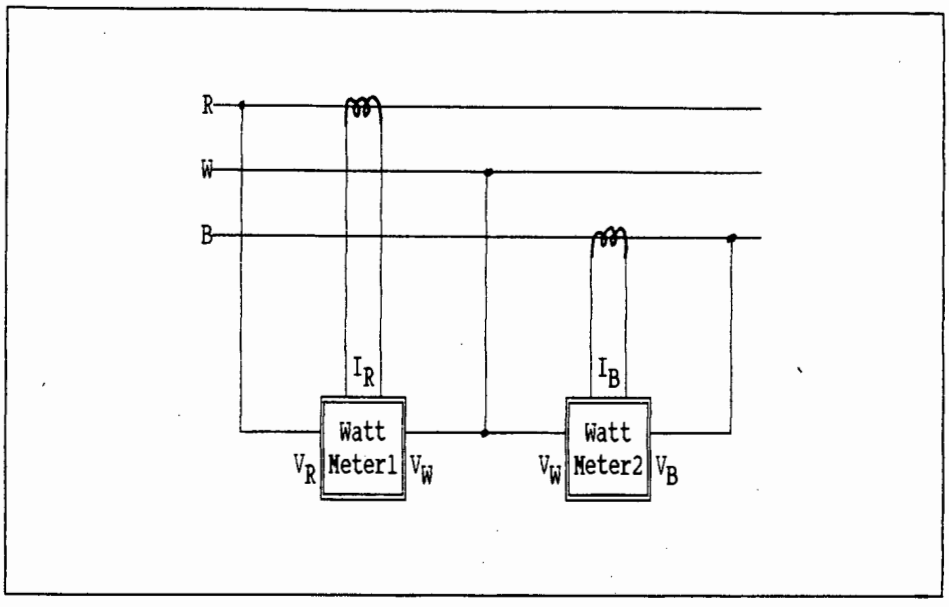


Figure 5.5: Measuring power using PT using the two Watt meter method

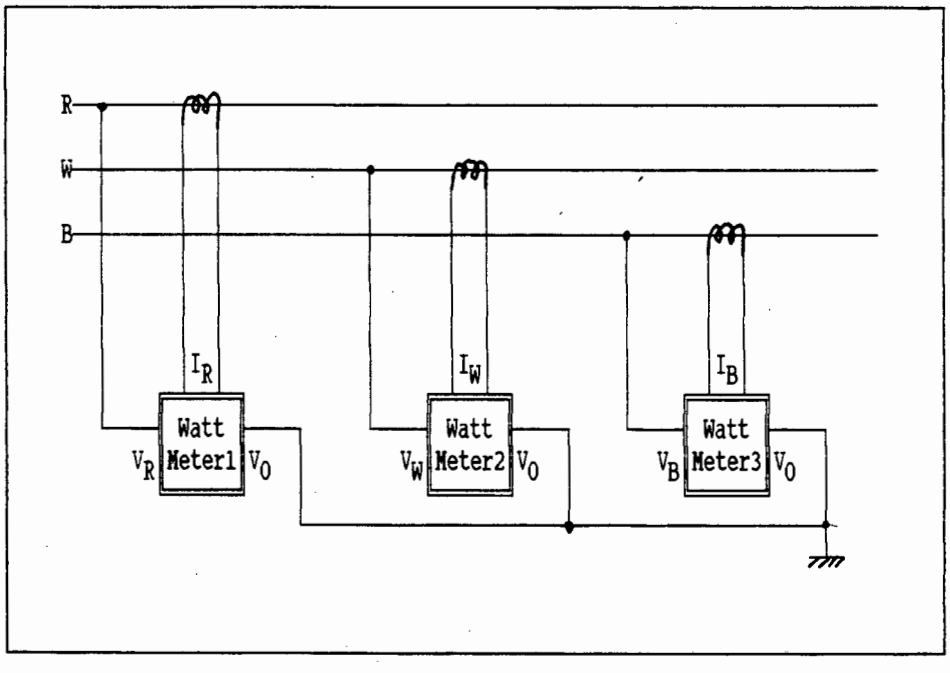


Figure 5.6: Measuring power using PT using the three Watt meter method

The results from the two PTs were found to give inconsistent readings in the following instances and hence they were considered unreliable:

- if the HCB was run under constant loading

(eg. with no mass on the belt) using the VSD, the readings from PTa did not correspond to those of PTb.

- if the HCB was running under constant loading once being supplied directly from the mains supply and then being supplied by the VSD (which was adjusted so that the belt speeds were the same), the readings from the PTs did not correspond. This test was done for both PTs and non-correspondence was identified for both.

- if the system was running at 50% speed using the VSD, the reading from PTa was negative, indicating, that if the reading was true, power was fed back into the supply. However the reading from PTb was positive under these running conditions.

The following possible sources of error were identified:

a) The power transducer or the Current Transformers (CTs) were incorrectly wired up.

b) The CTs were only suitable for true 50 Hertz sine waves (RMS).

c) Running an induction motor at low efficiencies causes power to be drawn at a low power factor. The PTs can only measure power correctly at power factors close to unity.

d) The PTs were not suitable to measure the power of non-sinusoidal wave shapes, i.e.

wave shapes which consisted of a fundamental and harmonics, as supplied by the VSD.

The possible sources of error were investigated consecutively:

a) The wiring of both the CTs and the PTs were confirmed to be correct.

b) The CTs were characterized, by investigating the current wave shapes on the input and the output of the CT for frequencies between 10 and 50 Hertz. A typical trace of these current wave shapes is shown in the following figure:

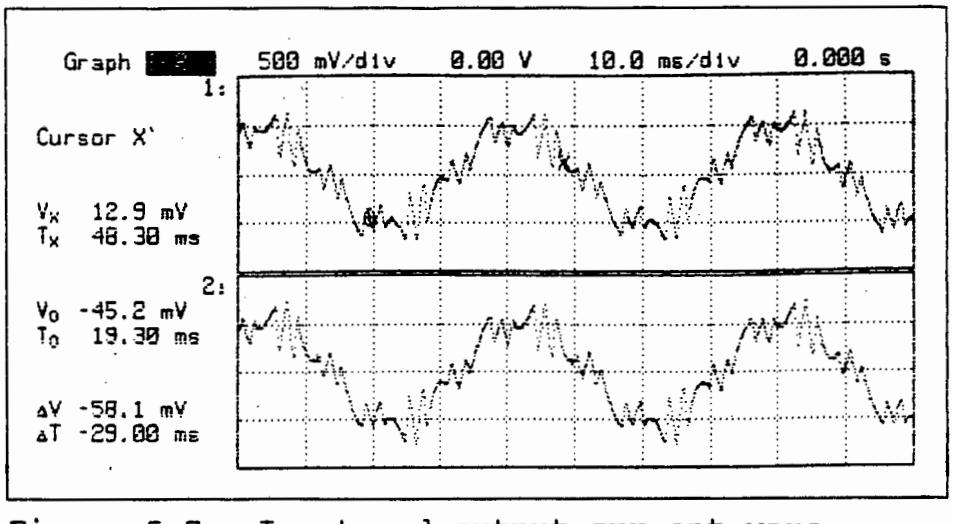


Figure 5.7: Input and output current wave shapes of a current transformer

From the current wave shapes the attenuation and the phase shift generated by the CT could be determined for various frequencies. The attenuation was found to correspond with the winding ratio of the CT, if the amplitude of the input and the output wave shape were compared. No phase

shift could be observed for the range of frequencies investigated.

As the wave shapes on the output of the CT were not distorted in any way, i.e. they were an exact replica of those on the input, it was assumed that the high frequency components of the wave were not affected either, otherwise the wave shape on the secondary of the CT would have a different shape to that on the primary.

It could thus be concluded that the CTs are not responsible for the lack of correspondence between the readings of power.

c) Considering the load characteristics of an induction motor with respect to the power factor of the system it can be seen that the power factor (pf) is undesirably low if the motor is loaded at less than 40 % of full power (i.e. pf < 0.8). These load characteristics are shown in the figure below for a typical induction motor (Say[14]):

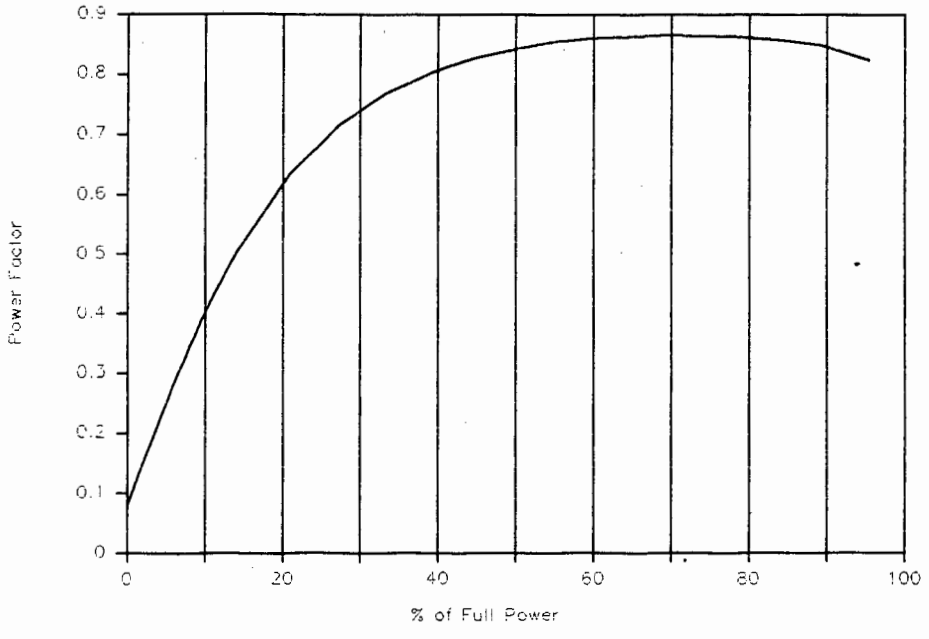


Figure 5.8: Power factor load characteristics of an induction motor

It has been mentioned in chapter 1 that if the system runs at above 20 % of full load the efficiency can be assumed constant. This requirement has to be increased to 40% of full load to insure that the phase angle between the voltage and the current phasors is less 35 degrees.

The motor driving the HCB was determined to be operating at less than 30 percent of full power if the CB is fully loaded, thus clearly below the acceptable limit.

Hence the 3HP motor was replaced with a 1HP motor. This resulted in an improvement in the readings of the PTb, but not in those of PTa.

d) The last problem that the PTs were not suited to the application of measuring power when harmonics are present on the

system can only be solved if the error due to the harmonics can be characterized and thus be accounted for. The calibration of the power readings from the PT versus a moving coil Watt meter will give a good indication of the error involved as the reading from a moving coil Watt meter can be assumed accurate even in the presence of harmonics.

The only PT that showed an improvement with any of the above suggestions was PTb. Thus this was the PT chosen for the Tests.

The only shortcoming of PTb was its slow response time. After consultation with the manufacturer the time constant of the filter on the output of the PT was changed from 220 msec to 12.3 msec. The steady state readings were confirmed not to have changed. The circuit diagram of the PT as supplied by the manufacturer is shown in Appendix F.

This PT already has a 0..20mA output signal, which can hence be used as is. Near the PC this signal was converted to a voltage signal by a current to voltage converter and then filtered. This was done by a first order low-pass filter in a similar way as for the speed signal in the previous section.

The PT was then calibrated against a moving coil Watt meter to establish the relationship between actual Watts and Watts as determined by the power transducer. The calibration curves and the regression results are shown in Appendix G.

From these calibration graphs it can be seen

that the correspondence between the readings from the moving coil Watt meter and the PT is acceptable. The direct and linear relationship between these two watt meters was observed In both modes of operation, i.e. direct mode and VSD mode. There is however also an offset present on the readings. If the system is operated in the direct mode the offset amounts to 47 Watts (0.94% of full scale) and if the system is operated with the VSD the offset amounts to 23 Watts (0.46% of full scale).

As the slope of the calibration graph is linear and equals unity, the relationship between PC binary counts and the power delivered can be determined as follows:

$$5 \text{ KW} = 20 \text{ mA} = 10 \text{ V} = 2048 \text{ counts}$$

Where 20 mA and 10 V denote the maximum possible control signal

Thus

$$1 \text{ count} = 2.4414 \text{ Watts}$$

5.3.4 Massflow

The MF on the HCB needs to be measured with a reference massflow instrument to be able to calibrate the massflow estimation technique. This can be done by using another massflow meter or by a technique suggested by the DRL. This technique relies on the fact that the gravel is fed in a circular fashion from the HB along the HCB to the ICB and then back into the HB. Refer to Figure 5.1 for a schematic model of the plant.

It thus follows that the mass that is not in the HB is distributed along the belts. The technique of the DRL assumes that both the inclined and the horizontal belts run at synchronous speed. The MF can then be determined as follows:

$$M_{belt} = M_{tot} - M_{hopper} \quad ...(5.1)$$

where
- M_{belt} - Mass of gravel on the belts
- M_{tot} - Total mass of gravel in the system
- M_{hopper} - Remaining mass of gravel in the hopper bin

Thus the MF equals:

$$MF = M_{belt} * v / l_{belt} \quad ...(5.2)$$

where
- v - speed of belt [m/s]
- l_{belt} - total length of belt (horizontal and inclined)

Hence

$$MF = M_{belt} / t_{cycle} \quad ...(5.3)$$

where
- t_{cycle} - cycle time of a particle of mass from the time it leaves the HB until it falls back into the HB, or the time to cover distance l_{belt} at speed v

This technique needs to be extended to incorporate the varying speed of the HCB due to the VSD. This can be done as follows:

$$M_{belt} = M_h + M_i + M_{ch1} + M_{ch2} \qquad ...(5.4)$$

where

M_h - mass on HCB at any instance in time
M_i - mass on ICB at any instance in time
M_{ch1} - Mass in the chute that directs mass from HCB onto ICB at any instance in time
M_{ch2} - Mass in the chute that directs mass from the ICB back into the HB at any instance in time

As the HCB, chute 1, the ICB and chute 2 are all in series it follows that the massflow is constant along the CBs. The chutes will be approximated by a small CB of length l_{ch1} and l_{ch2} running at speed v_{ch1} and v_{ch2} for the first and second chute respectively. Substituting Eq.(5.2), Eq.(5.4) becomes:

$$M_{belt} = MF * \left(\frac{l_h}{v_h} + \frac{l_i}{v_i} + \frac{l_{ch1}}{v_{ch1}} + \frac{l_{ch2}}{v_{ch2}} \right)$$

It will be assumed that the speed of chute CBs equals the speed of the belts preceding the chute.

Thus:

$v_{ch1} = v_h$
$V_{ch2} = v_i$

Hence:

$$MF = \frac{M_{belt}}{\frac{l_h + l_{ch1}}{v_h} + \frac{l_i + l_{ch2}}{v_i}} \quad ...(5.5)$$

Thus the following constants need to be determined:

l_h, l_i, l_{ch1}, l_{ch2}, v_i

The length of the HCB and ICB were determined with a tape measure:

$l_h = 9.50$ m

$l_i = 11.25$ m

The equivalent length of the chutes was determined by measuring the time a particle took to pass through the chute at constant and known speed for the HCB and ICB:

$l_{ch1} = 2.06$ m

$l_{ch2} = 1.33$ m

The speed of the ICB was measured by measuring the time of 10 revolutions of the belt:

$V_i = 1.12$ m/s

The variables to be measured consist of:

M_{belt}, v_h

v_h is already measured with a tacho and thus only needs to be entered into the above equation. M_{belt} can be determined by measuring M_{tot} when no mass is on the conveyor belts (i.e. before a test) and M_{hopper} can be measured during a test.

A problem experienced during the testing was the

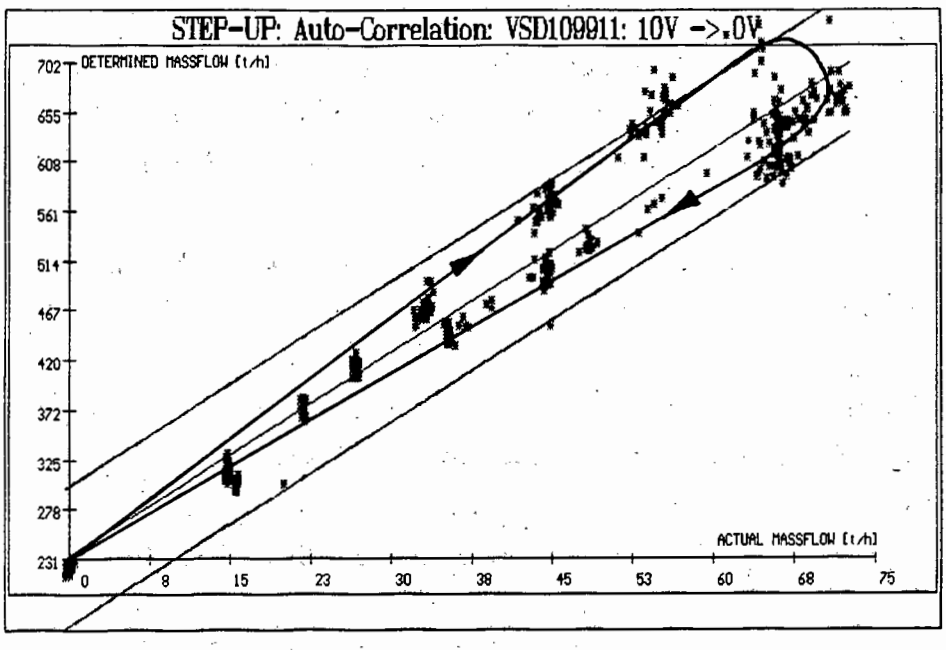


Figure 5.9: Correlation graph illustrating the effect of Hysteresis due to the reference massflow determination technique

This hysteresis explains why later in this chapter the correlation tests were only performed for the MF being increased in small steps or decreased in small steps, but never both.

Hence a device is needed which measures the weight in the HB and feeds this signal back to the PC. This can be done by mounting the actual HB on load cells. In the case of the Laboratory test rig at the DRL three strain gauge type load cells are used to support the entire weight of the HB.

The signal from these three load cells is sent to a load cell amplifying circuit. In this circuit the signal from each load cell is amplified on its own, before the three signals are added together and the offset (to accommodate the empty mass of the HB) is subtracted. Lastly the reading is calibrated to

PC, is filtered by a first order low pass filter.

The relationship between the mass in the HB and PC binary counts can be established as follows:

1626.6 kg = 8.03 V
10V = 20mA = 2048 Pc counts

Hence
1 count = 1.011032 kg

To confirm the validity of the technique to measure the massflow on the conveyor belt, the results determined from Eq.(5.5) were compared to actual belt cuts. The two massflow estimates correlated well as can be seen by the calibration graph and results in Appendix G. The maximum deviation between the determined value using Eq.(5.5) and the value obtained from a belt cut was 3%.

5.3.5 Other Variables to consider

A - Friction of the system

As mentioned in chapter 2 the function of load force F_1 with respect to mass M and with respect to speed v has to be determined. This can be done by running the conveyor at constant speed and mass on the belt and thus measuring the power supplied to the belt. Dividing the power measurement by the speed measurement will yield an estimate for the load force of the system.

This test was performed for different speeds and

masses on the belt, and thus the first and second order correlation between speed and mass versus force were established. The correlation graphs together with the regression results are show in Appendix G. The load force characteristics thus could be modelled by a:

- 1st order model:

$$F_l = 416 + 1.85*M - 256*v$$

- 2nd order model:

$$F_l = 533 + 0.25*M + 0.00693*M^2 - 524*v + 293*v^2$$

In the following section it is determined experimentally if this lengthy correlation process of the load force is necessary and justifiable, or whether it would not be easier to just assume the load force as being constant.

B - Sampling frequency

To accurately log the system data with a PC the system response had to be sampled at the correct frequency. Too low sampling frequencies would result in a loss of information and too high frequencies would result in excessive, unnecessary data being captured.

A rule of thumb to chose the sampling frequency is, that the period of sampling should be approximately one tenth of the time constant of the plant (EEE417 Advanced Control Notes [6]). A further rule of thumb to determine the time constant of the filter is, that it should be

twice the sampling period.

Thus the time constant of the test plant was determined. A typical response and the calculation of the time constant is shown Appendix G. The response time can be approximated to equal 100 msec.

Hence the sampling period was set to 10 msec, and the time constant of the filter to 20msec.

C - timing of Datalogging Routine

The timing of data capture routines can be calibrated in various ways. If the sampling time of the routine is in the order of seconds the Dos clock of a PC can be used to ensure the correct timing. The Dos clock samples time to a resolution of 100th of a second only. Thus if the sampling time is in the order of milli seconds as in the case of this study, other means have to be thought of.

One solution would be to introduce a certain number of wait states in form of wait loops between successive samples. The number of wait loops depends on τ_s and on the speed of the PC. The exact number of wait loops can only be established by trial and error.

An option has been included in the software to do a certain number of sampling tests with a specified number of wait loops. From the total time it takes to log all the samples (determined by using the dos clock), the actual and the required sampling time can be determined, which is then displayed on the screen. Thus the

number of wait loops can be increased or decreased accordingly. An example of what the screen looks like after such a trial and error test is shown in the following table:

TIMING CALIBRATION OF DATALOG

treq[msec]:	tact[msec]:	noloop:
20.000	1.860	0
20.000	24.280	1050
20.000	23.280	1000
20.000	21.100	900
20.000	20.000	850
20.000	20.000	850
20.000	19.980	850

OK? [N]

Table 5.1: Example of timing calibration of datalogging routine

The timing of the data logging routine was confirmed by specifying a pulse of length 250 msec to be put out by the PC. This pulse was then logged with a high speed digital oscilloscope and the pulse length was measured. An example of the trace of the digital scope is given in the figure below:

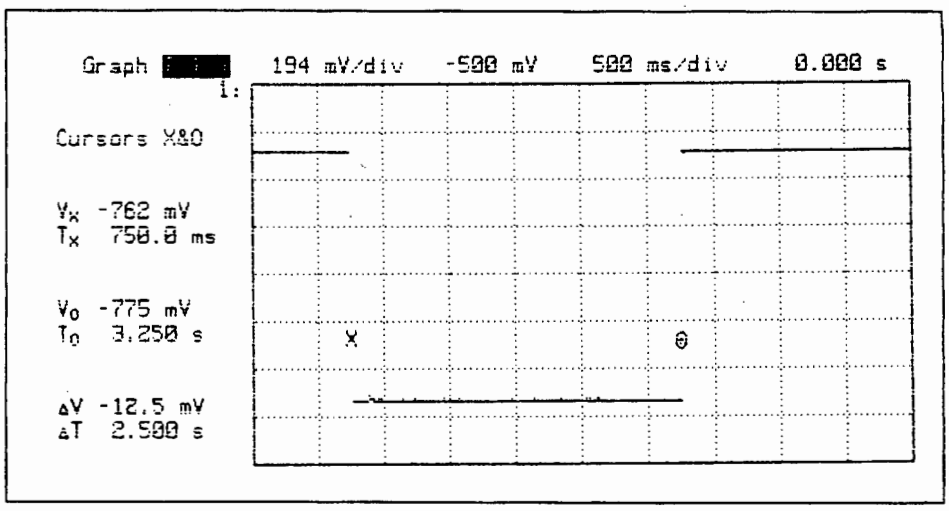


Figure 5.10: Confirming the timing of the data logging routine with a 250 msec pulse

As can be seen from the above figure the pulse length is indeed 250 msec, and thus the routine to calibrate the sampling time can be assumed correct. A problem was however experienced in the way that the number of required wait states varied from one execution of the program to the next. It is thus suggested that the timing of the data logging routine is checked each time after a program is re-started.

5.4 RESULTS OF TESTS PERFORMED

5.4.1 Validating the Technique to use

This section aims to validate the technique to be used in the instrument that is going to be proposed in a next section and which is then going to be tested on a mine. The following options will be investigated in more detail:

- Assuming the load force of the system is constant with respect to speed and mass on the belt

- Modelling the load force with a first order model

- Modelling the load force with a second order model

- Disregarding the dynamics of the speed signal and thus assuming the speed signal equal to a pulse.

To investigate the above cases and to compare the various options objectively against each other, real time responses of power and speed were obtained for different massflows by using the DLOG package. These responses were then used to estimate the massflow on the HCB using each of the above techniques. As a reference massflow was obtained with each of these responses from the difference of mass in the hopper bin, the MF estimates could be correlated against each other and thus the imprecision for the different techniques could be compared on an equal base.

The real time responses were obtained for various modes of operation. The operation modes differed in that the size of steps varied. The size of step applied equalled 100%, 75%, 50%, and 25% of full speed, whereas the initial setpoint always equalled 100% or full speed.

A typical pulse response of speed and power for the system running at 100% and a 75% perturbation or step being applied, is shown below. Typical pulse responses for speed and power for the other modes of operation are given in Appendix H.

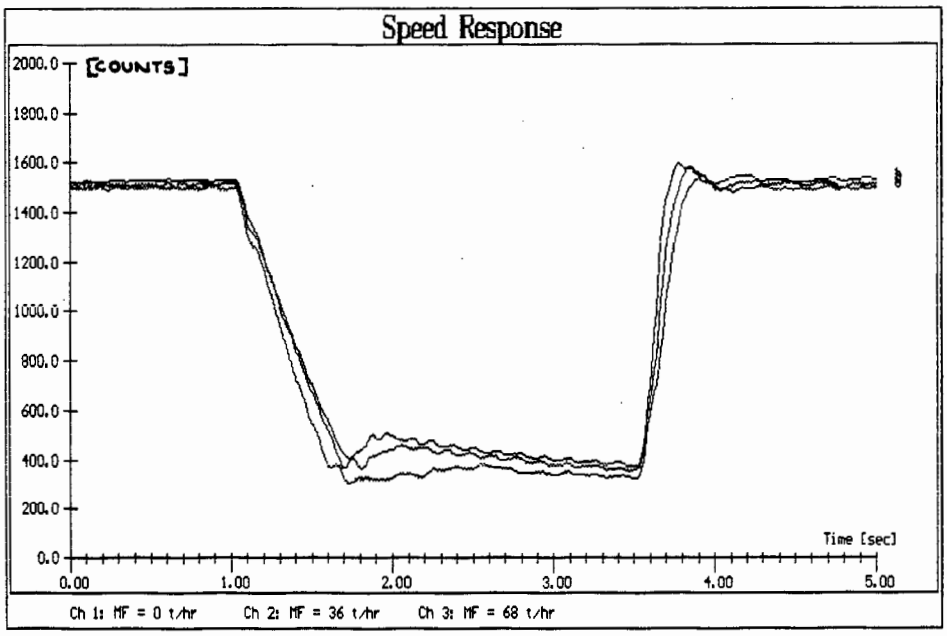


Figure 5.11: Typical speed pulse response for a 75% step in speed

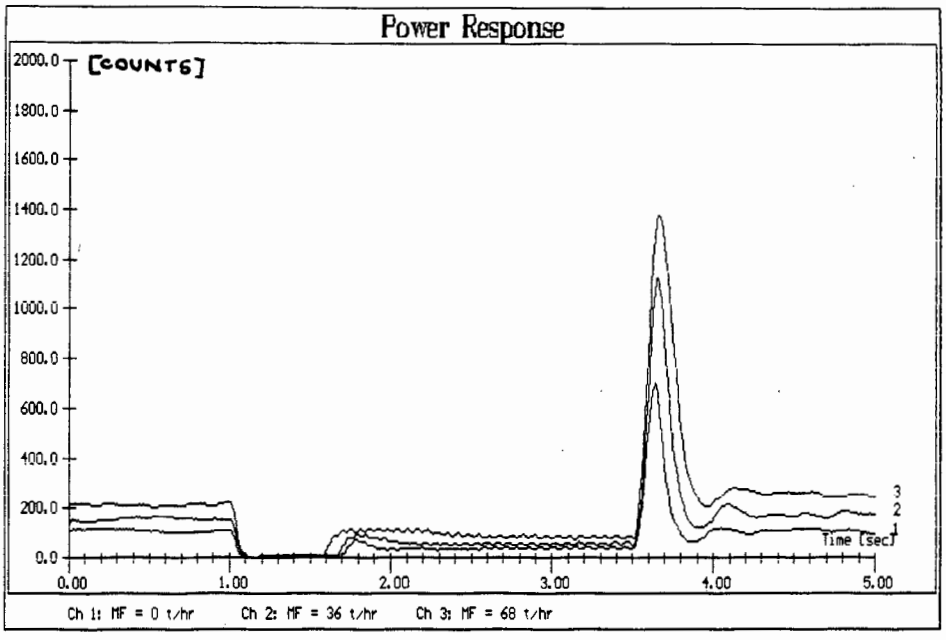


Figure 5.12: Typical power pulse response for a 75% step in speed

These real time responses were used in the MFCM package to estimate the MF on the CB for the different options mentioned above. The values of imprecision will give an indication of the loss of accuracy due to the application of the above options in the techniques. The calibration results are shown for both massflow being estimated from both the step-up and step-down response.

The values given for the imprecision will represent the 95% confidence or error bands. The value stated will be percentage error of full-scale at the mean of the range of readings.

The results for the different options and for the different modes of operation can be summarized in the following table:

	% Step from full speed			
Option of Operation:	100	75	50	25
F_1= Const	6.15	8.24	9.89	19.94
F_1= av+b	6.18	8.24	9.88	19.90
F_1=av+bv²+c	6.17	8.25	9.90	19.94
Speed not logged	9.29	7.98	10.28	21.70

Table 5.2: Percentage imprecisions of massflow estimates as determined from step-up responses

	% Step from full speed			
Option of Operation:	100	75	50	25
F_1= Const	12.24	78.50	36.70	69.90
F_1= av+b	15.04	69.70	37.90	74.50
F_1=av+bv²+c	14.40	75.60	35.80	64.84
Speed not logged	98.50	29.67	136.10	65.10

Table 5.3: Percentage imprecisions of massflow estimates as determined from step-down responses

Comparing the two tables it can be seen that the imprecisions from the MF estimates from the step-up responses are superior to the ones from the step-down response.

Comparing the first three cases in the two tables above, where the load force is modelled as a constant, a first order polynomial and a second order polynomial, it can be seen that the

imprecision hardly changes in the case where the MF estimates are determined from the step-up response. In the case of the step-down response the imprecision varies: For a 100% perturbation the imprecision deteriorates whereas for the 25% perturbation the imprecision improves as the load force is modelled with a higher order model. Generally it can said that the modelling of the load force with a higher order polynomial has little effect on the imprecision. In the light of the effort connected to determining the load force characteristics it is of great benefit to assume the load force as a constant force independent of speed and mass on the belt.

Considering the last case of interest, where the speed is not measured, but assumed to be of a pulse form, similar to the actual perturbation, the imprecisions for the MF estimates as determined from the step-down response have deteriorated considerably, whereas only a slight deterioration in the MF estimates from the step-up response can be observed. This confirms the statement made in chapter 2, that the dynamics of speed will have little effect on the MF estimate from the step-up response. It can thus be concluded that if the step-up response is used to estimate MF on the conveyor belt it is not entirely necessary to measure the speed of the system. This has the advantage that no tacho is needed for the measuring instrument. The setpoint to the VSD can be used to determine a value for the speed of the belt.

Hence the following strategy is suggested:

- Use the step-up response to estimate the massflow on the HCB, as firstly the

precision is superior and secondly the speed needs not be measured.

- The load force can be modelled by a constant as determined from a steady state test before each pulse test.

During the following tests the speed, however, will still be measured for confirmatory reasons.

5.4.2 On-line Calibration Results

The strategy suggested in the previous section will thus be used to estimate the massflow on the HCB in an on-line fashion. This will be done using the MFCA package.

Four modes of operation were investigated:

A - Using the VSD to perturb the system: Keeping the initial setpoint speed at 100% of full speed and thus applying various steps of speed.

B - Using the VSD to perturb the system: Running the system at different initial speed setpoints and thus applying such a step that the CB will stop.

C - Using the SSRs to effect the perturbation: The system is stopped by cutting the power to the system

D - Using the SSRs to effect the perturbation: The power to the system is cut for 0.5 seconds. Hence the HCB will only slow down and then speed up again.

For the cases where the VSD was used to perturb the system, only the imprecisions from the step-up response will be discussed. The regression results from the step-down are however always shown in Appendix H.

Operation mode A was investigated for steps of 100%, 75%, 50% and 25%. A typical speed and

power responses for a 50% perturbation is shown in the following two figures. Typical responses for the all sizes of steps investigated are shown in Appendix H.

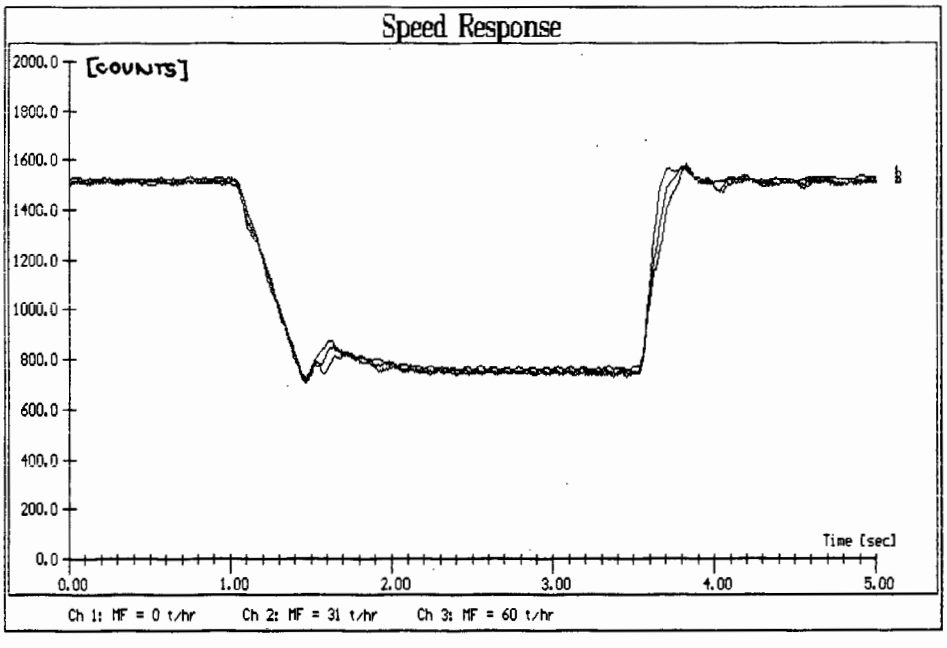


Figure 5.13: Typical speed pulse response for a 50% step in speed

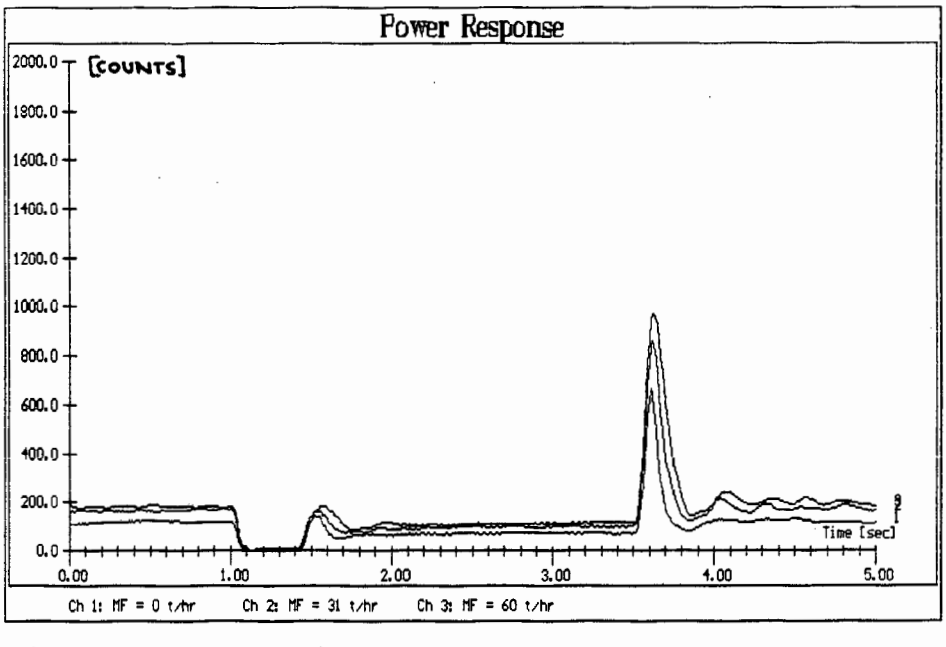


Figure 5.14: Typical power pulse response for a 50% step in speed

The imprecision equals the 95% confidence band expresses as a percentage error of full-scale at the mean of the range of readings. The results of imprecision for tests run at different sizes of step are given in the table below:

STEP [%]	IMPRECISION [%]
100	5.34
75	8.92
50	11.71
25	24.1

Table 5.4: Imprecisions of massflow estimates for various steps with constant setpoint

The correlation graphs and the regression results for the above tests are shown in Appendix H.

Plotting these imprecisions versus the percentage step applied will yield the following graph. Given a certain required imprecision for the MFM this graph can then be used to determine the size of step that has to be applied.

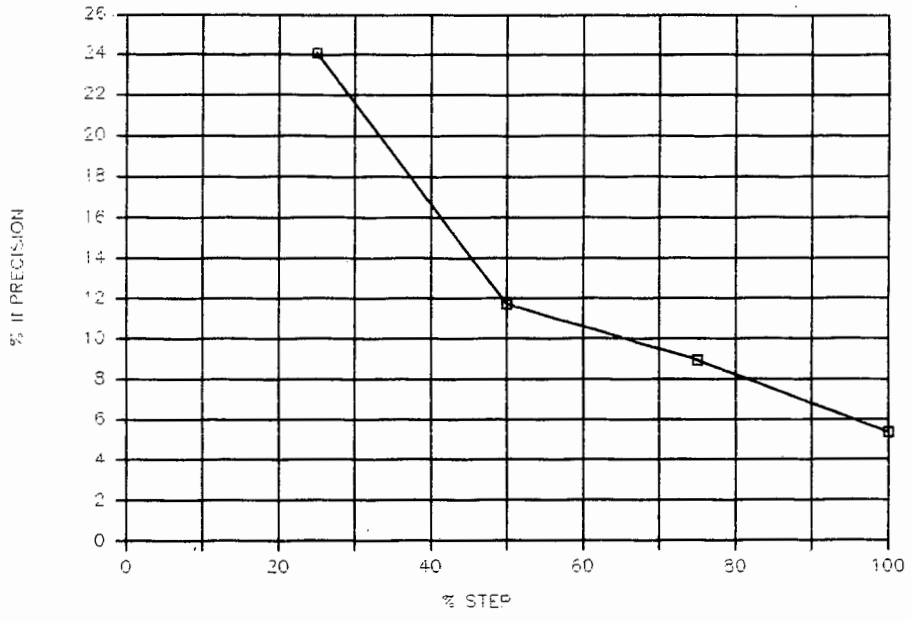


Figure 5.15: Imprecision versus percentage step applied.

Operation mode B was investigated for initial speed setpoint of 100%, 75% and 50% of full speed, while always applying such a step that the system would stop.

A typical speed and power response for the system running at 50% of full speed is shown in the following figures. Responses for all the setpoints are shown in Appendix H.

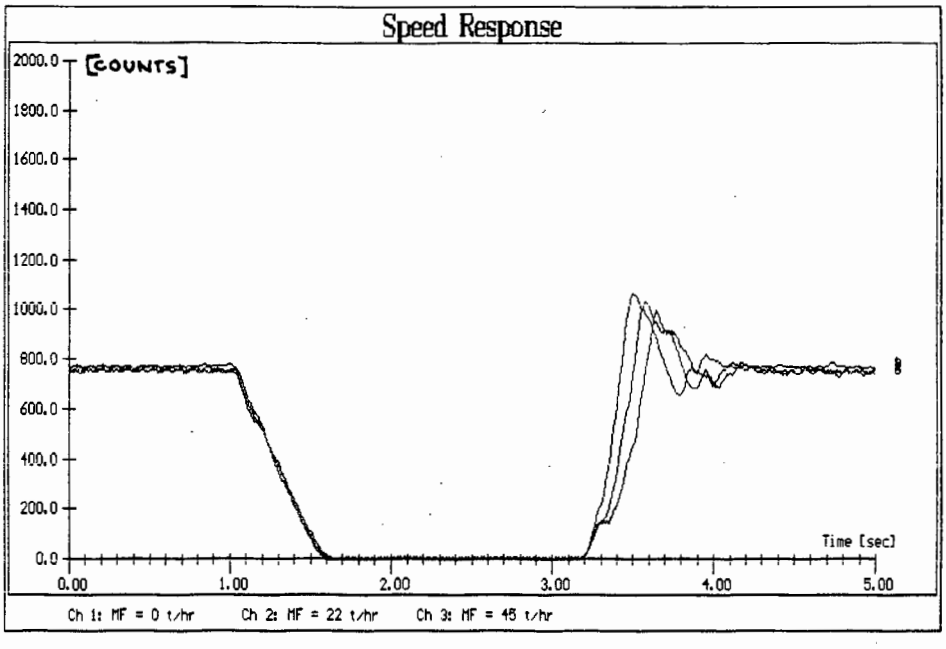


Figure 5.16: Typical speed pulse response for an initial speed setpoint of 50% of full speed

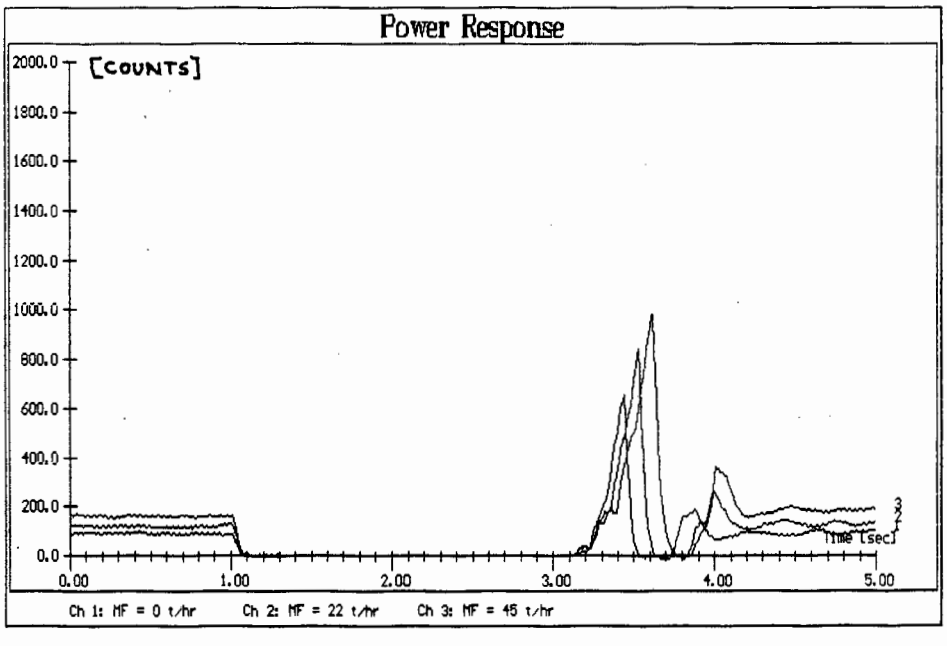


Figure 5.17: Typical power pulse response for an initial speed setpoint of 50% of full speed

The resulting percentage imprecisions as determined from a series of tests is summarized in the following table. The actual correlation

graphs and the regression results for each initial speed setpoint are shown in Appendix H.

INITIAL SPEED SETPOINT [%]	IMPRECISION [%]
100	5.64
75	9.21
50	8.04

Table 5.5: Imprecisions of massflow estimates for various steps with constant setpoint

The imprecisions for this series of tests are not of prime importance. More interesting are the regression results. It can be seen that for different initial speed setpoints the regression results in terms of y-intercept and slope differ considerably. This might impose a problem if the HCB is operated under speed control as then the calibration changes each time the speed setpoint changes. Correlating the slope and the y-intercept against the size of the initial speed setpoint yielded the following graphs:

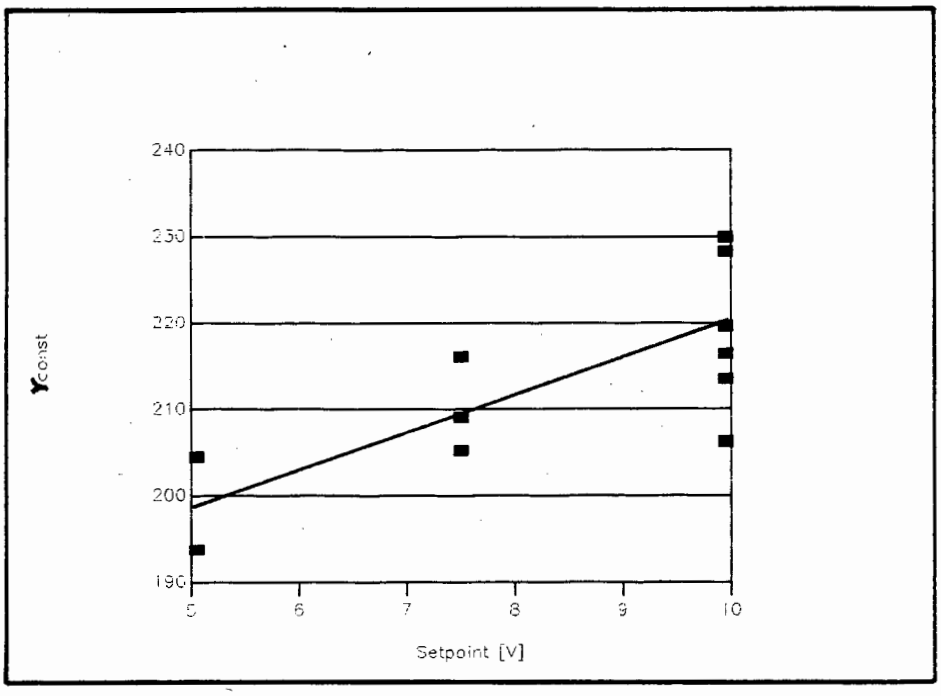


Figure 5.18: Correlation of Y-intercept versus initial setpoint

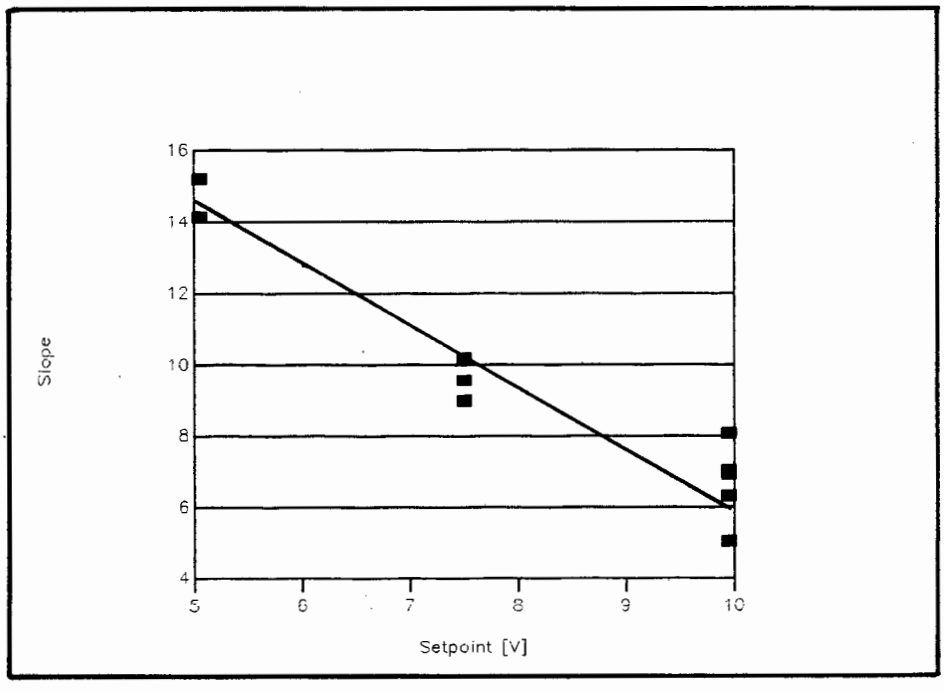


Figure 5.19: Correlation of slope versus initial setpoint

In a case where this change in calibration constants does impose a problem, the calibration coefficients could be modelled by fitting a function to the above graphs and thus modelling

the change in calibration coefficients. This would however complicate the installation of a MFM that should be an instrument easy to install.

It also follows that the imprecisions for a belt running at constant speed will be superior to the case where the speed is controlled, as the calibration coefficients can always be assumed accurate.

In operation mode C the SSRs were used to perturb the motor in such a way that the CB was stopped. A typical speed and power response is shown below:

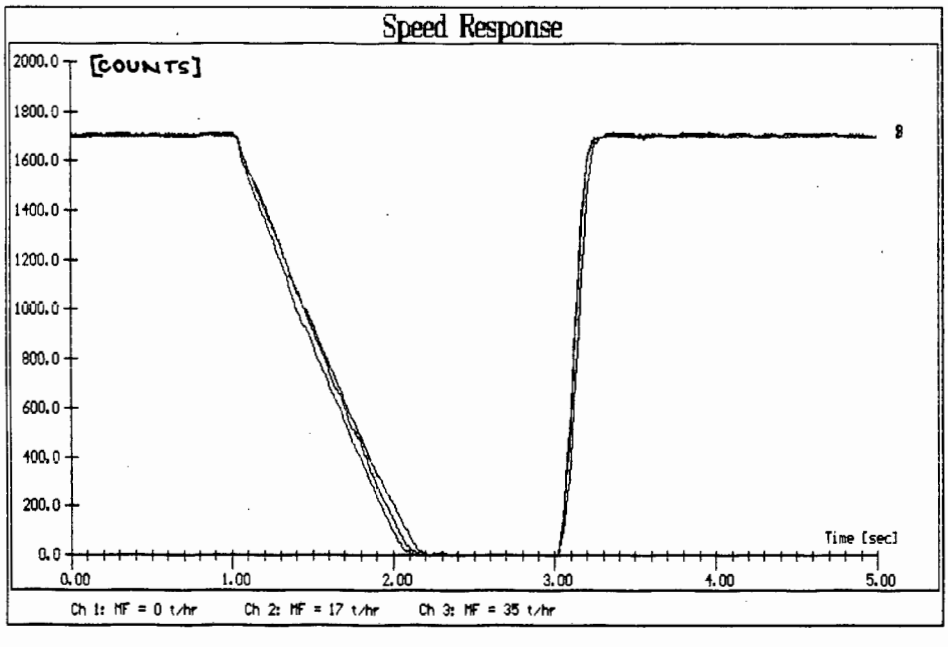


Figure 5.20: Typical speed pulse response for the system being perturbed with SSRs

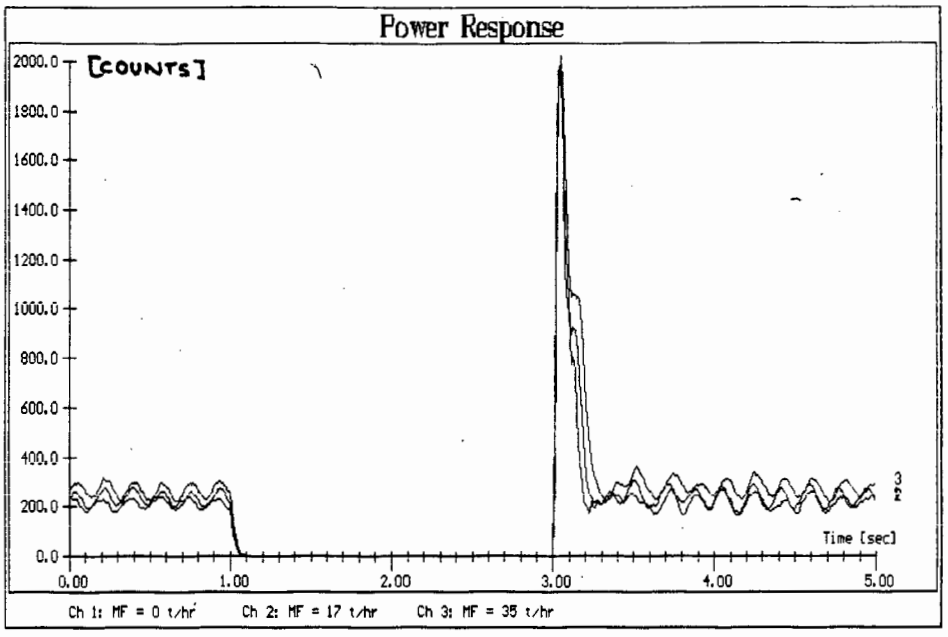


Figure 5.21: Typical power pulse response for the system being perturbed with SSRs

The imprecisions as derived from the step-up and step-down responses are shown in the following table. The actual correlation graphs and regression results are shown in Appendix H.

	IMPRECISION [%]
Step-up	25.60
Step-down	20.72

Table 5.6: Imprecisions of massflow estimates when perturbing system with SSRs

It follows that the imprecision has deteriorated, if this case is compared to the case where the system is perturbed with a VSD when running at full speed and applying a 100% step. This deterioration of imprecision is however justifiable if cost is of prime

importance and hence if the cost of a VSD in not justified. Also note that superior results for the precision are now obtained from the step-down response and not from the step-up response as in the case where the system is perturbed with a VSD.

In operation mode D the SSRs were again used to perturb the system, but in such a way that the power was only cut for 0.5 seconds. The system was thus not allowed to stop, but only to slow down. A typical speed and power response are shown below:

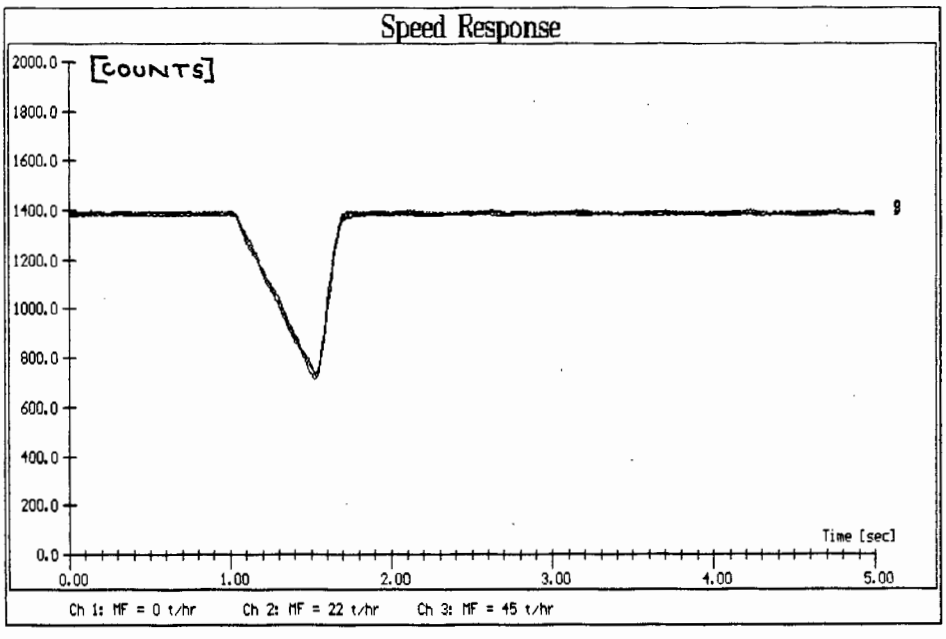


Figure 5.22: Typical speed pulse response for the system being only slowed down with the SSRs

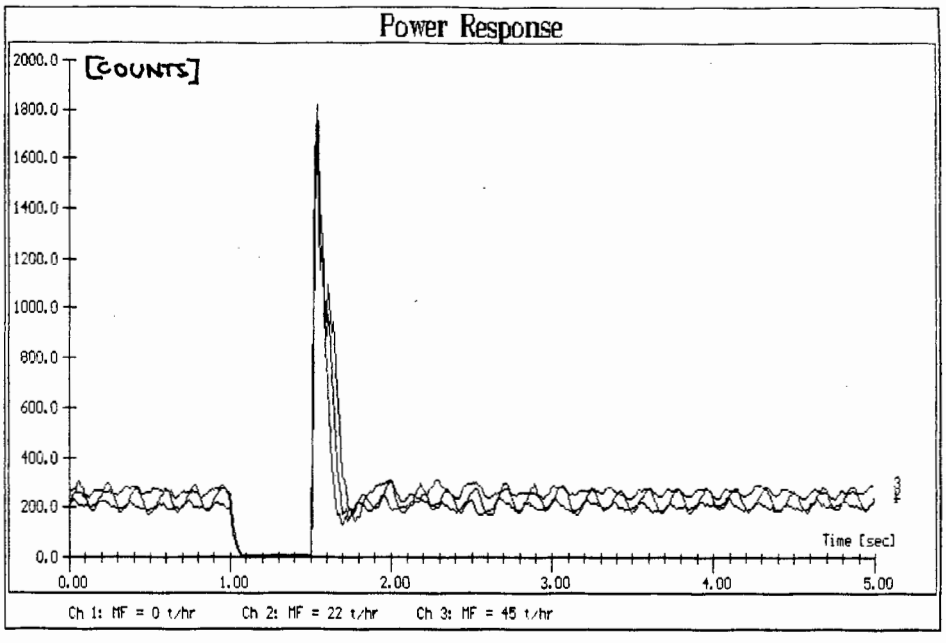


Figure 5.23: Typical power pulse response for the system being only slowed down with the SSRs

The imprecisions as derived from the step-up and step-down response are shown in the following table. The actual correlation graphs and regression results are shown in Appendix H.

	IMPRECISION
Step-up	91.45 %
Step-down	22.08 %

Table 5.7: Imprecisions of massflow estimates when perturbing system with SSRs

A radical deterioration of imprecision can be observed for the step-up response, whereas the imprecisions of the step-down response only deteriorated slightly. The option of applying a shorter perturbation and thus letting the system only slow down is only of benefit if the step-down response is used to determine the massflow

on the CB.

5.4.3 Cross Validation Results

Cross Validation is done to prove the consistency of the results. It is done in such a way that a certain portion of the data is used to calibrate the instrument and thus these calibration coefficients are applied to the rest of the data. The error that results is an indication of the error that could be expected in practice if the instrument is calibrated on a few samples and is then left to itself.

As only the total amount of tons recorded per hour or per shift is of interest the cross validation will be performed on a basis of the error in the total amount of tons measured. For instance if the Reference meter indicates that 100 tons have flow across the CB in the past hour and the MF estimation technique indicates that only 95 tons have flown across, the error will equal - 5 %. It has to be noted that the error of instantaneous readings will be higher as the totalizing of the MF to yield tons in a certain portion of time acts as an integrator on the measurements, i.e. a filter with infinite time constant (see chapter 2), and thus the random error or the imprecision on the reading will be filtered away (see chapter 3).

The cross validation will also only be performed for one test mode, which is the test mode which yielded the best imprecision. The test mode investigated is the case where the system is running at full speed and a 100% step is applied with the VSD. The system was calibrated by slowly increasing the MF on the belt until full capacity. A typical time plot of MF as

determined by the reference instrument and as estimated is shown below:

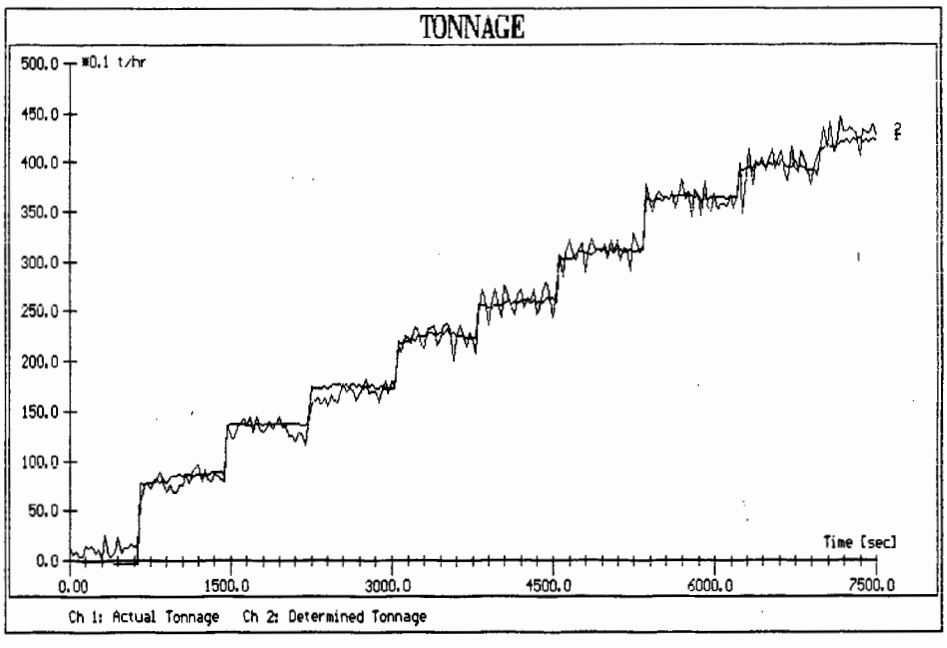


Figure 5.24: Time graph of reference and estimated massflow

This response was used to perform a cross validation in the following way: The time scale was split up into 5 equal intervals:

0	<=	t	<=	1000 sec
1000	<=	t	<=	3000 sec
3000	<=	t	<=	4500 sec
4500	<=	t	<=	6000 sec
6000	<=	t	<=	7500 sec

The interval 3000 <= t <= 4500 was used to calibrate the instrument as it represents the readings at the centre of the scale of measurements. Normally the first time interval would be chosen, but the calibration is of no use if carried out for the instrument operating at only 10% of its full scale.

The resulting calibration coefficients were as follows:

Y-Intercept: 207.95
Slope: 7.15

These calibration coefficients were then applied to all the other time intervals. The table below shows the actual tons flown across the CB in that particular time interval and the associated error if this reading is compared to the reference reading.

Time Interval [sec]	Total Tons [t]	Error [%]
0<=t<=1500	2.01	20.450
1500<=t<=3000	6.52	-2.060
3000<=t<=4500	9.98	0.224
4500<=t<=6000	13.82	-0.733
6000<=t<=7500	16.64	-0.673
0<=t<=7500	50.94	0.155

Table 5.8: Table of cross validation results

It can be seen that the error between the estimated massflow and the actual massflow in very small. Only in the first case between 0 and 1500 seconds the error is considerable. This error, however has little effect on the total tons flown past the CB in the two hours that this test was conducted. The 2 tons measured in the first 1500 seconds only contribute 4% of the total tons recorded. It

can hence be concluded that the MF estimation technique gives excellent results if the results are totalized.

In the last case illustrated in the table above, where the same calibration constants have been applied to all the data, the error in total tons measured is even lower, which confirms the statement made above that a totalizer acts as a filter with infinite time constant.

The correlation graph for this series of tests can be viewed in Appendix H.

5.4.3 Effects of Filtering

All the above samples have been taken as is without applying any kind of filtering. The effect of filtering to reduce the imprecision associated with an instrument is investigated in this section.

The Reference and the estimated MF samples were filtered three different filters:

a) τ_f = 0 secs (i.e no filtering)

b) τ_f = 60 secs = 1 min

c) τ_f = 600 secs = 10 mins

The correlation coefficients resulting from this filtering were as follows:

Filter Time Constant [s]	Correlation Coefficient
0	0.9963
60	0.9986
600	0.9991

Table 5.9: Correlation factors versus filter time constant

It can be seen that the filter only results in a slight improvement of the precision. This is due to the fact that there aren't any dynamical effects which the filter can account for. Thus it can be assumed that the dynamics of the Reference instrument and the estimation technique are similar.

The correlation curves and the time plots of massflow for the three filters applied are shown in Appendix H.

5.5 PROPOSED INSTRUMENT

The results in the previous sections indicate that the precision for a system using the VSD to perturb the system are superior to a system where the SSRs effect the perturbation of the system. The cost of a VSD justifies an application using a VSD only to such a system where a VSD is already installed. The cost of the VSD thus has no effect on the overall cost of this MFM. If a VSD is not available on a system that this MFM should be installed on, the SSRs is an option to effect the perturbation to the system, at reduced accuracy however.

Results from the previous sections also showed that the load force of the system can be approximated as a constant, and that the speed signal need not be recorded, if the step-up response is used to determine the massflow on the system.

Hence a prototype instrument is proposed, which ties all the results from the previous sections together and using the theory as described in chapter 2. This instrument also aims to fulfill the objectives of this study of being a low-cost massflow meter easy to install on existing HCBs.

The hardware can be summarized in the following block diagram:

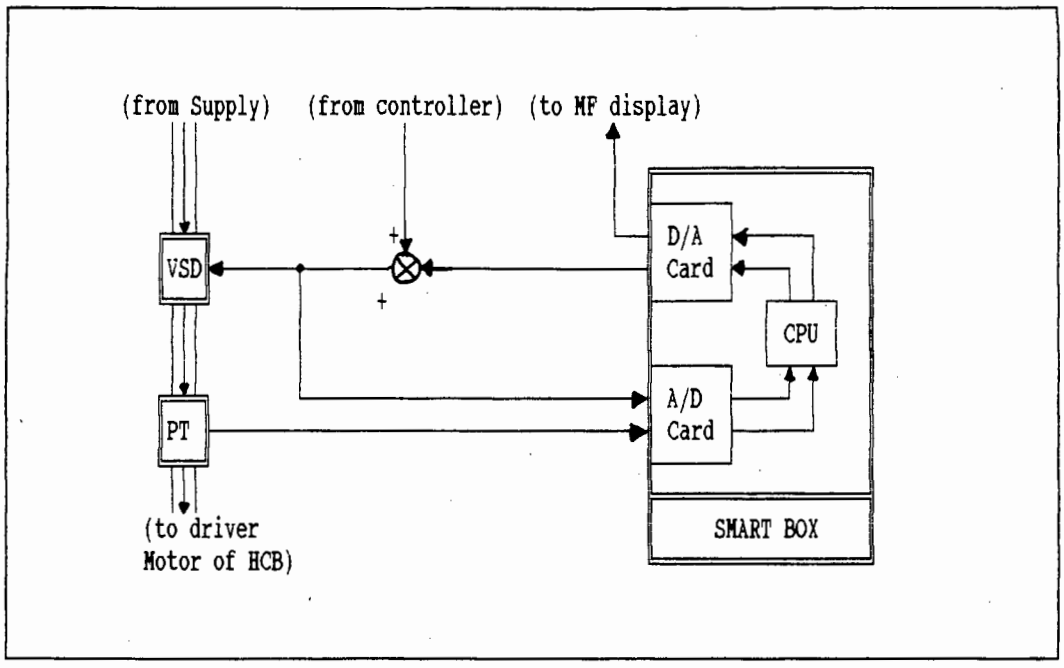


Figure 5.25: Block diagram to illustrate functional layout of proposed instrument

The system consists of a smart box, an adding circuit and a PT.

The smart box is basically a micro processor based device with some memory, a high speed two channel A/D converter and a two channel D/A converter. The CPU logs the power and control signal (which is proportional to the speed of the CB), while perturbing the control signal to the VSD with a pulse. From the response it can determine the MF. Thus it will send a signal proportional to the MF to the second output channel which may be connected to a display or some data logging device.

The adding circuit is a normal op-amp adding circuit. The adding in this case is not performed by the smart box, because in a case the smart box being switched off, the system now still operates further.

The PT is a commercially available device. The device should however be checked if it delivers consistent readings if used in conjunction with a VSD.

The algorithm to be programmed onto the microprocessor is explained as follows:

The power signal P_d and the control signal c have to be logged for a period of t_{40}, i.e. from t_0 to t_4. The following events which take place during the time of t_{40} are worth noting:

$t_0 <= t <= t_1$: System is kept at steady state in the unperturbed mode. This cycle is used to determine the load force of the system.

$t = t_1$: Step-down signal is applied to the input of the VSD to slow the CB down.

$t_1 <= t <= t_2$: System slows down.

$t_2 <= t <= t_3$: Steady state in the perturbed mode. This cycle can be short, as only the average speed of the system in the perturbed mode needs to be determined.

$t = t_3$: Step-up signal is applied to the input of the VSD to speed it up again.

$t_3 <= t <= t_4$: Step-up test takes place. From the power response the mass on the CB can be determined.

$t = t_4$: End of test

A typical response of power and control signal will look as follows, where t_0 = 0s, t_1 = 1.0s, t_2 = 2.5s, t_3 = 3.5s and t_4 = 5.0s:

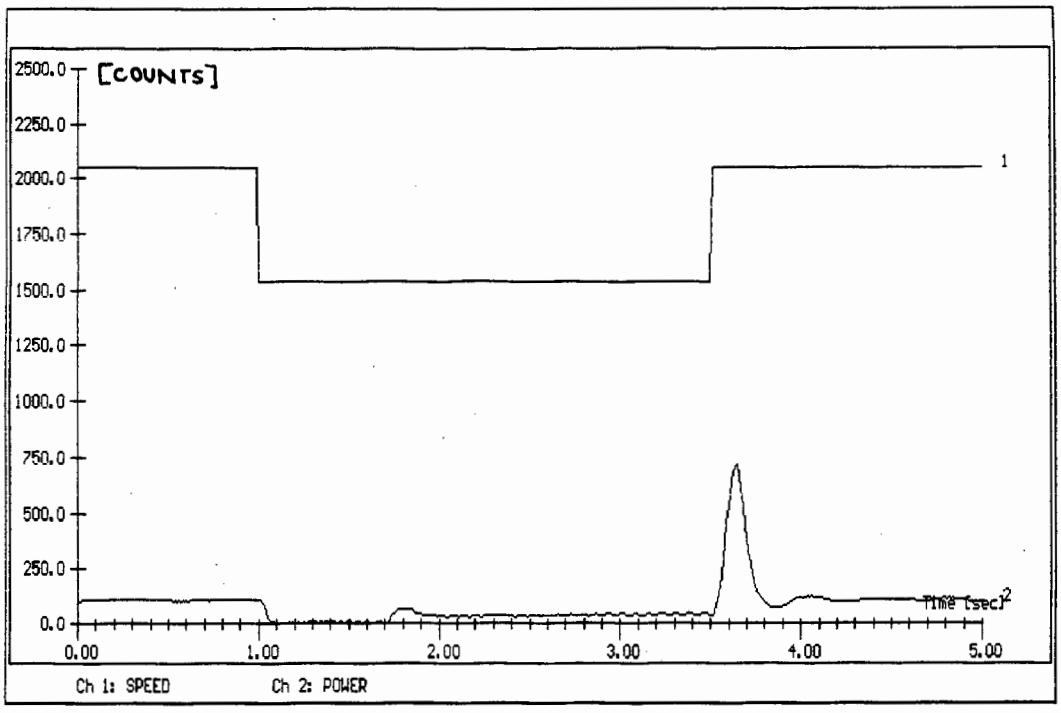


Figure 5.26: A typical test response of power and speed

The algorithm is thus be explained in the following flow diagram:

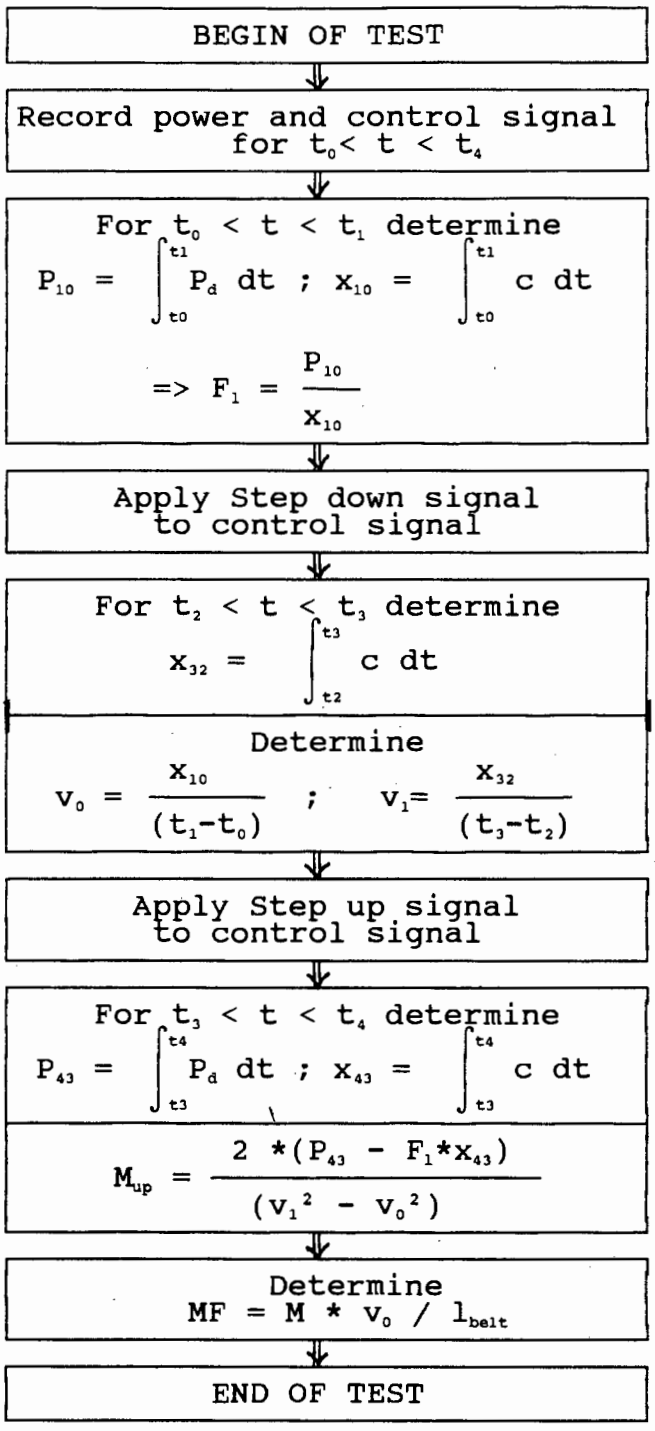


Figure 5.27: Flow chart to illustrate the algorithm to be used in the proposed instrument

This test can be repeated at equal intervals or can be performed if a certain condition is met, like that the speed control signal to the VSD has to reach steady state.

The length of the test determines the minimum sampling period of the instrument. It is however suggested that the sampling period should be much longer than the length of the test as this will decrease the influence the instrument has onto the system. In the example of a pulse response given above the length of the test equals five seconds. During the test series the test was however only applied every 30 seconds.

The requirement of the instrument being easy to install are met. Only the power to the motor needs to be cut to install the PT. The adding circuit has to be switched in the control signal cable before the VSD.

The low cost was the second requirement of this instrument. The cost of such a circuit could be summarized as follows (excluding the price of a VSD):

- Smart box : R 2500.-
- PT : R 900.-
- adding CCT : R 10.-

Total : R 3410.- (excl. GST)

This cost estimate at July 1990 prices shows a considerable saving if the price of a nuclear weightometer (R20.000.-) or the price of a Load Cell arrangement (R17.000.-) is considered.

CHAPTER 6:
VALIDATION OF THE PROTOTYPE IN AN INDUSTRIAL ENVIRONMENT

6.1 DESCRIPTION OF BASIC PLANT

The aim of the test plant in an industrial environment was to test the prototype as suggested in the last section of the previous chapter in an industrial environment to prove its suitability and usefulness. It can generally be assumed that the external influences in a true industrial environment are bigger than in an laboratory environment. This explains why a portion of the tests as performed on the laboratory test rig are repeated on a mine test rig.

The criteria used to choose a test rig was that a MFM had to be available on that particular process stream, in order that the instrument could be calibrated. The mine chosen for the tests was the De Beers Consolidated Diamond Mine in Kimberley.

The test rig consists of one of the feeder belts to a crushers in the Re-crush Section of the Treatment Plant. Oversize material was fed into a bin above the HCB. Through an adjustable gate the gravel fell onto the HCB which transported it to the crusher. Behind the crusher the crushed material was send on a CB past a nuclear weightometer to screens, where the oversized material was sorted out again. A schematic diagram of the plant is shown in the following diagram:

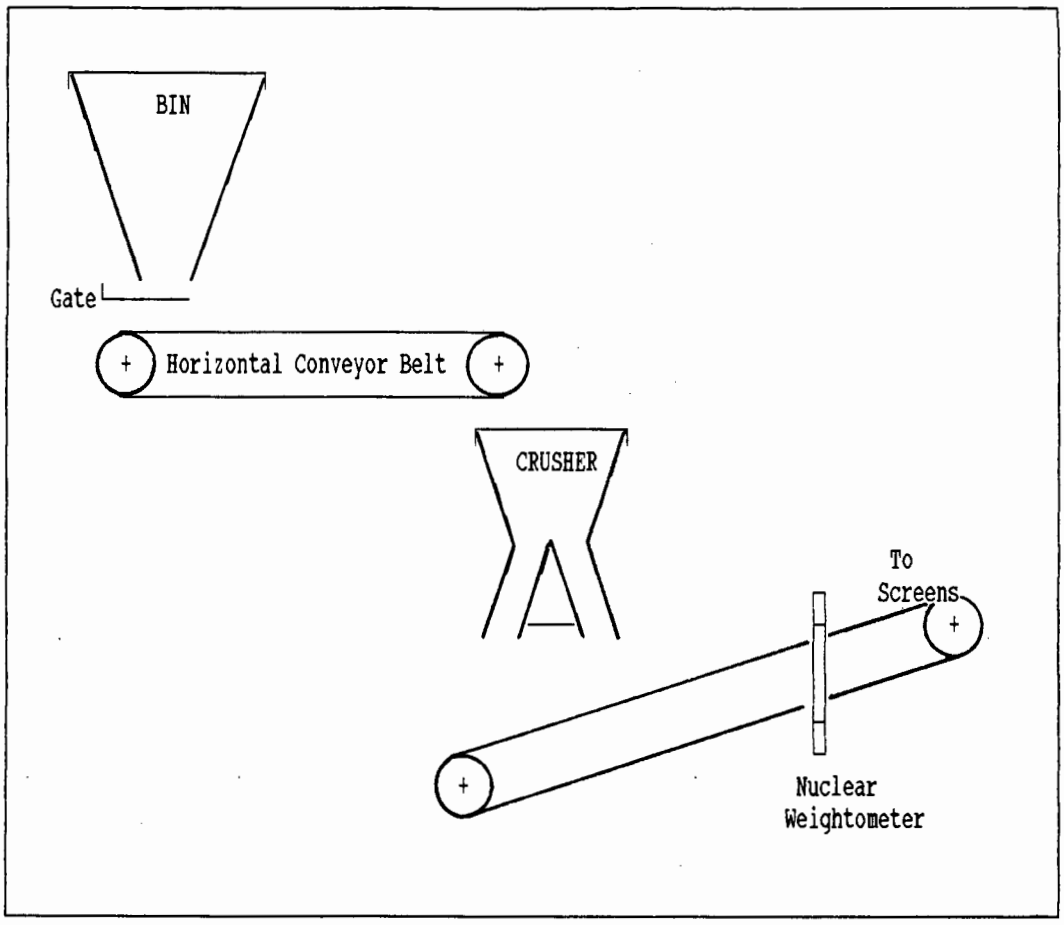
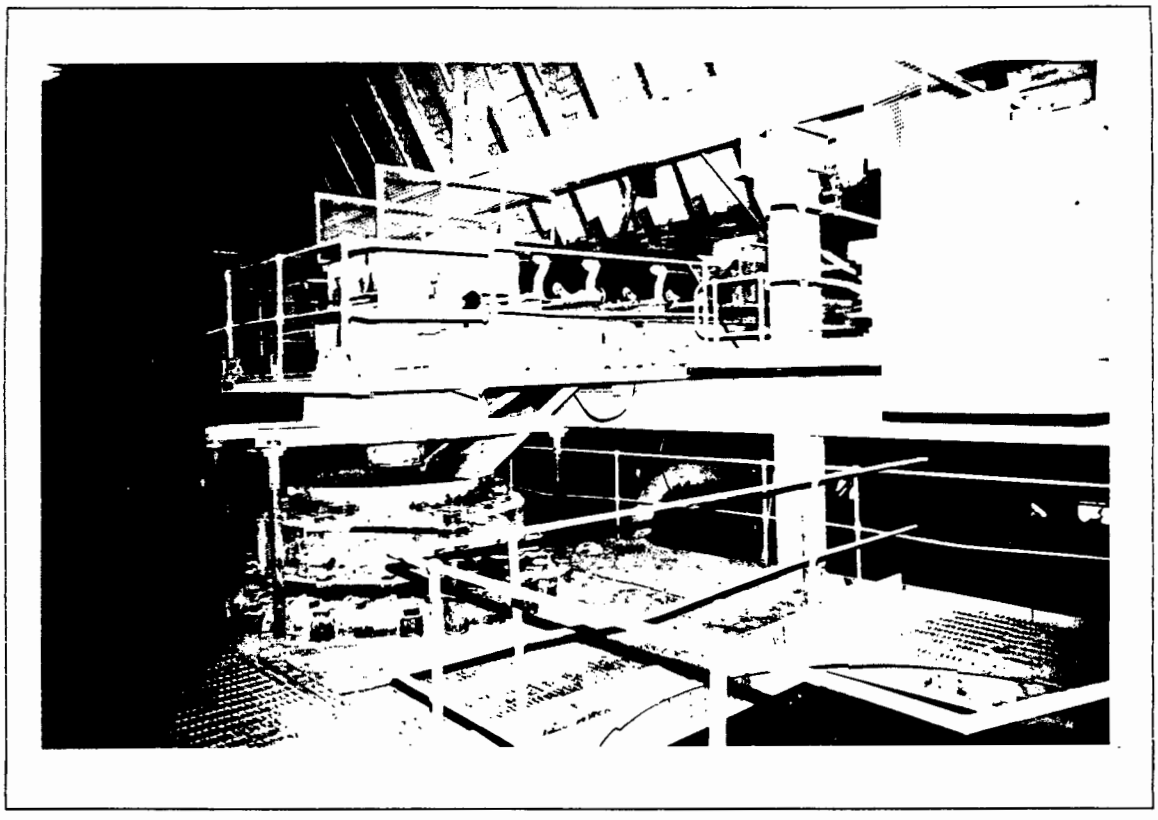
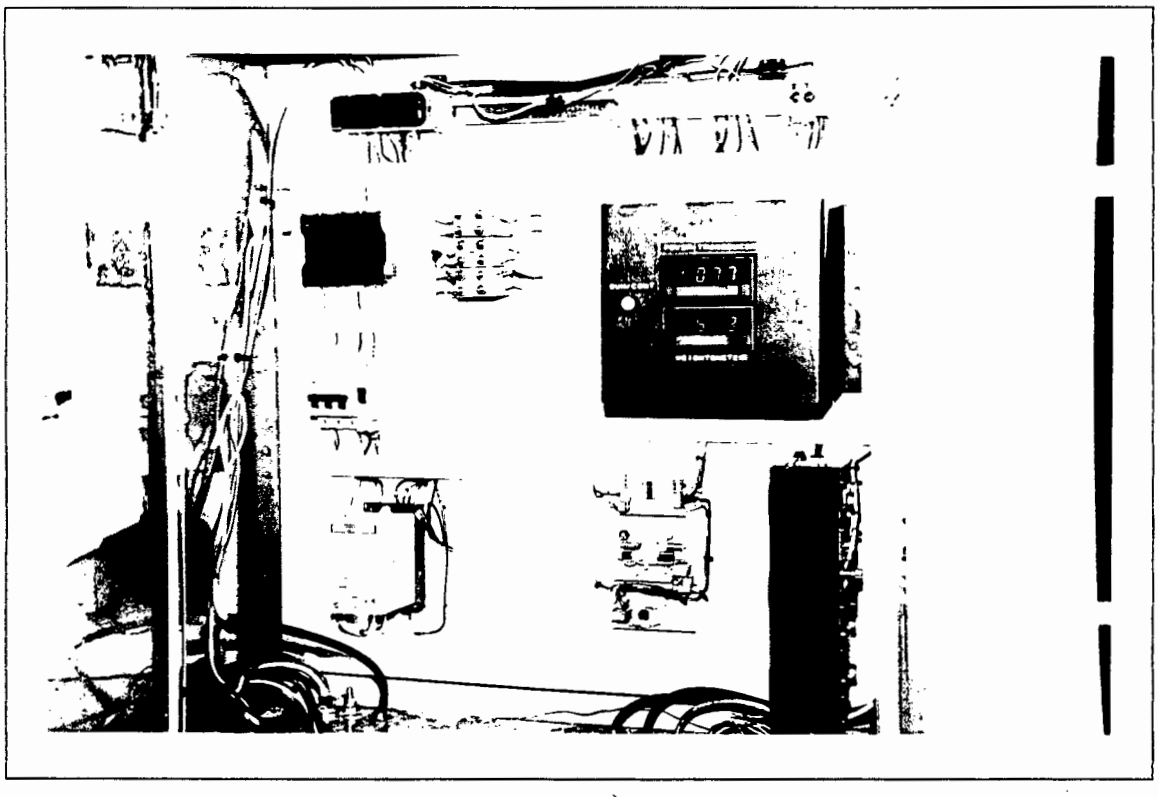


Figure 6.1: Schematic diagram of test rig in an industrial environment

In the photographs below the laboratory test plant and the control panel of the test plant are shown:



Photograph 6.1: Photo of the test rig on a Mine



Photograph 6.2: Photo of the control panel of the test rig on a mine

During the installation of the prototype it was important that the time of interruption of the power to the CB was kept to a minimum. The same applied for the actual tests done.

In the following sections the specifications of the test plant are given more detail.

6.2 SPECIFICATIONS OF 380V CIRCUITRY

The only option of perturbation investigated on the mine was that with a VSD. The VSD was already existent on the mine as the level of the crusher was controlled by controlling the speed of the belt. As the belt was already installed in the plant no start/stop circuits were required. Thus the direct supply of the power to the motor was only interrupted to install a PT.

Below a schematic circuit diagram of the required changes are shown. The dotted line indicates the interruption of the power supply cable. A detailed circuit diagram of the system is shown in Appendix I.

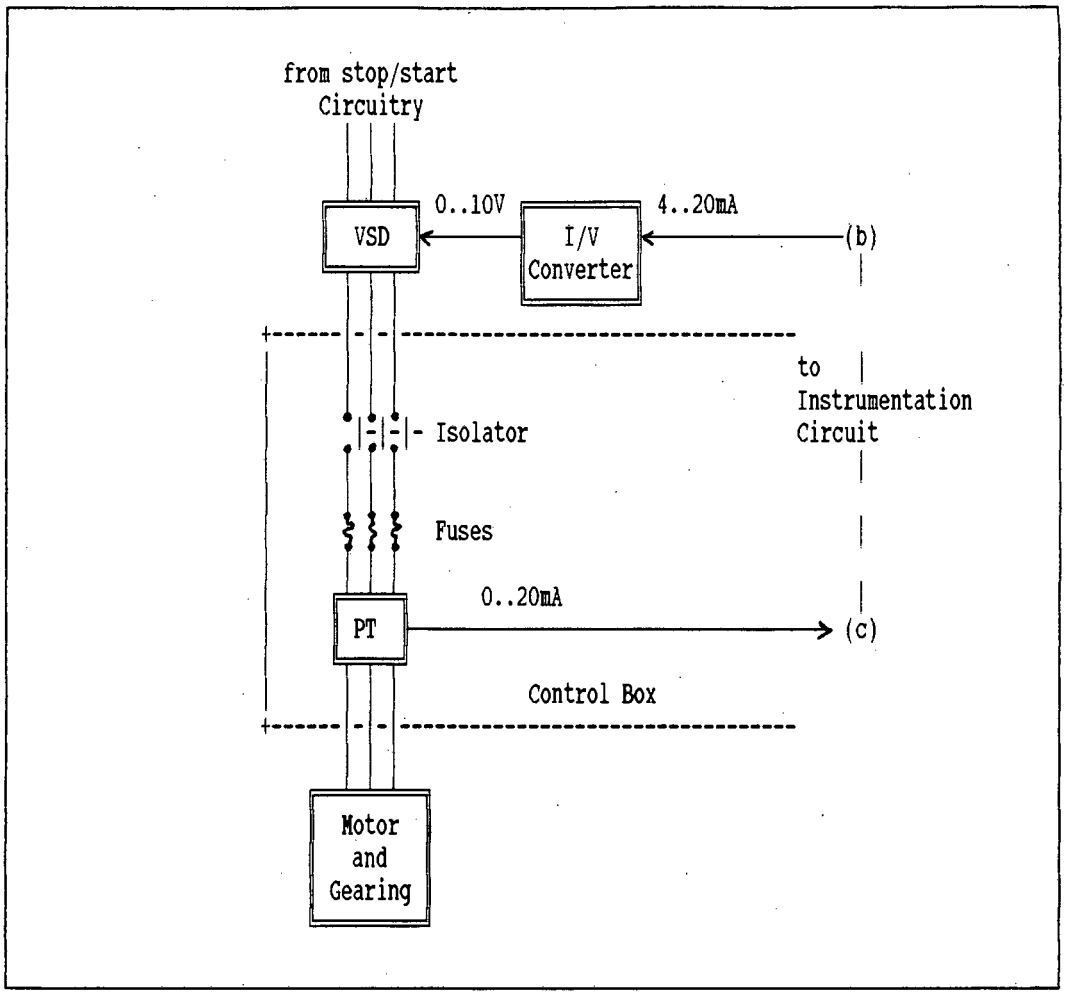


Figure 6.2: Circuit diagram for the 380 V network of the test rig used on a mine

As the signal controlling the VSD has to be a voltage signal and as the signal standards require a current signal the current signal controlling the VSD has to be converted from a 4..20mA signal to a 0..10V signal. The signal from the PT is already a current signal which can hence be used as is.

6.3 INSTRUMENTATION CIRCUITRY

The instrumentation circuit basically consists of feeding the signals of speed, power and reference massflow from the test plant back to the PC and to perturb the control signal of the VSD from the PC. A schematic circuit diagram is shown below, whereas a

6.3.1 The Control Signal for the VSD

The setpoint signal for the VSD is obtained from a Proportional control loop, which controls the level in the crusher by varying the speed of the belt. This speed signal is to be perturbed by a pulse to slow the belt down and subsequently to speed it up again.

As the speed signal is changing continuously depending on the level in the crusher, the tests would obviously be influenced by these changes. Thus the setpoint signal had to be kept constant for the duration of the test. This can best be done with a sample and hold circuit which is switched in series with the setpoint control signal. This Sample and Hold circuit is triggered when the signal from the PC is stepped negative. The circuit then holds the output at the value of the input for the duration of the test. After the test the output follows the input again. A detailed circuit and timing diagram of the circuit is shown in Appendix I.

As mentioned before the signal from the Sample and hold circuit is then passed through an adding circuit. On the adding circuit the perturbation signal from the PC is added to step the signal to the VSD first down and then up. As the signals should normally be current signals the same circuit as used for the laboratory test plant is suggested to add currents. The operation of this circuit has already been explained in the previous chapter and the circuit diagrams have already been shown in Appendix F.

6.3.2 Speed

The speed was again measured with a DC Tacho generator. In this case the tacho was not mounted on the shaft of the driving motor, but a wheel mounted on the shaft of the tacho which touched the actual belt. The tacho was mounted on a spring loaded lever which would always ensure that the wheel was pressed against the belt. A photograph below illustrates the arrangement:

Photograph 6.3: Photo to illustrate mounting of tacho.

The voltage signal from the tacho is converted to a current signal to conform with the systems standards. Just before the PC it is converted back to a voltage signal. It is then filtered by a low pass filter to eliminate high frequency noise, before it enters channel 1 of the input channels of the A/D card.

The calibration was performed by timing a number of revolutions of the belt and at the same time logging the speed on the PC. Thus a calibration curve could be plotted (m/s vs. PC counts). From the regression results the relationship between the translational speed and PC counts could be established:

$$1 \text{ PC count} = 6.216 * 10^{-4} \text{ m/sec}$$

The calibration graph and the regression results are shown in Appendix J.

6.3.3 Power

During initial tests at the DRL it was established that the most consistent readings of power were obtained from the PT using the Two Wattmeter method. This same PT was thus chosen for the tests in an industrial environment.

It was however impossible to confirm the validity of the power readings against a moving coil Watt meter for instance, as was done in at the DRL. The readings from the PT thus had to be assumed to be true. This might be a foolish assumption as it can also be assumed that the motor driving the CB is normally overrated. As mentioned in the previous chapter this has the disadvantage of a low power factor and an uneconomic efficiency. Thus inaccurate power readings can be expected from the PT.

As the output signal from the PT is already a current signal it needs not be converted, but it is converted to a voltage signal near the PC. It is then filtered by a low pass filter before

being send into channel 2 of the A/D card of the PC.

The conversion coefficient could thus be established as follows:

10kW = 20mA = 10V = 2048 PC counts

Thus

1 PC count = 4.8828 Watts

6.3.4 Massflow

The reference massflow to calibrate the MF estimation technique against was obtained from a nuclear weightometer which was mounted behind the crusher.

The signal from this reference MFM is already a 4..20mA signal. Thus the signal need only be converted and filtered near the PC before it is read by channel 3 of the A/D card of the PC.

The calibration of the nuclear MFM was performed by a professional firm. A copy of the calibration results is shown in Appendix J. Thus the conversion coefficient should be established as follows:

300 t/hr = 20ma = 10V = 2048 PC counts

Thus

1 PC count = 0.1464 t/hr

During the tests an offset error was established however. This offset error equalled 10.5 t/hr

if the CB was running empty.

As the nuclear weightometer is switched behind the crusher the dynamics of the crusher have to be taken into account, if this massflow signal is to be used to calibrate the MF estimation technique. In chapter 3 it is suggested that both the massflow signals should be filtered by a filter that has a much larger time constant than the crusher itself. In this way all dynamical effects are suppressed.

6.3.5 Time constant of plant

To determine the time constant of the system τ_p and thus the sampling frequency τ_s and the time constant of the filter τ_f, a similar procedure as in the case of the laboratory test rig was followed.

The τ_p was determined to be approximately equal to 200 msec. A detailed calculation of the time constant is shown in Appendix J.

Hence τ_s was taken as 20 msec and τ_f was taken as 40 msec.

6.4 RESULTS

The tests on the mine served the purpose to validate the MF estimation technique in an industrial environment. The following strategy was used in the tests:

- The load force of the system was assumed to be constant

- The speed was logged for confirmatory reasons. This was however not entirely necessary.

- For all the tests performed the CB was perturbed in such a way that the system was stopped totally and then speeded up again.

- The dynamics of the crusher in the process of calibration were accounted for, by filtering both the reference and the MF estimate.

6.4.1 Constant Speed Tests

Firstly the CB was investigated and calibrated for constant speed applications. The system was run at constant speeds of 100%, 75%, 50% and 25% of full speed and thus perturbed to stop. A typical response of speed and power is shown in the following figures for a speed setpoint of 50% of full speed. Copies of typical speed and power responses for the other initial speed setpoints are shown in Appendix K.

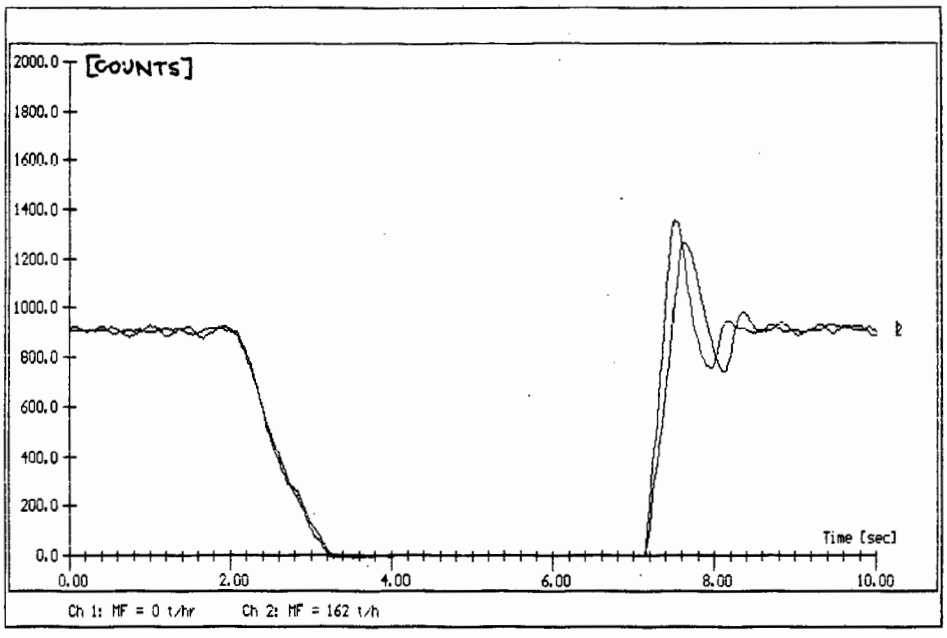


Figure 6.4: Speed pulse response for v_0 = 50%

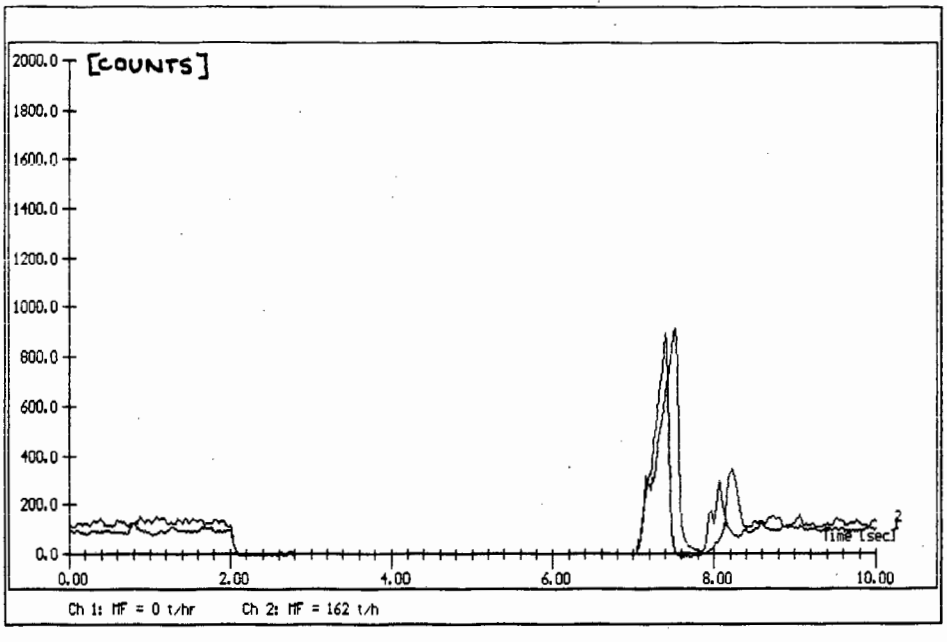


Figure 6.5: Power Pulse Response for v_0 = 100%

A direct correlation between the MF estimated and the reference massflow would not yield any results, as the dynamic characteristics of the crusher would distort the results. The results thus have to be filtered. To chose an adequate filter time constant both MF samples were filtered for different values of τ_f and the resulting envelopes of the reference and the MF

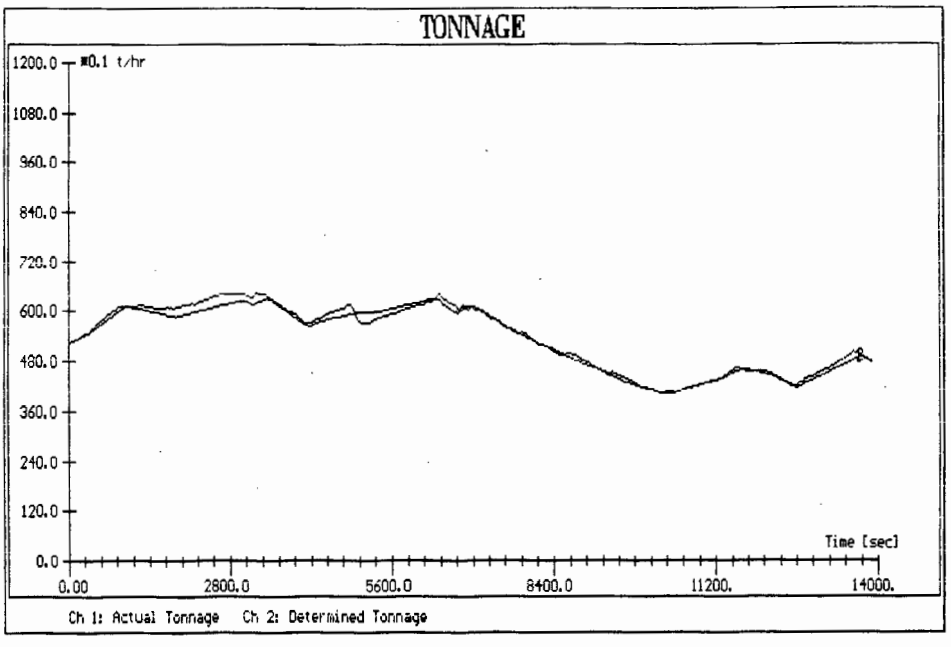


Figure 6.8: MF time plot for τ_f = 6000s

Correlating these filtered traces against each other for different τ_f will give an indication of the improvement in correlation due to the filtering. The following table shows the correlation coefficients for different τ_f's. The actual correlation curves are shown in Appendix K.

Filter Time Constant [s]	Correlation Coefficient
6000	0.9879
600	0.9714
60	0.8122

Table 6.1: Table of filter time constants versus correlation coefficients

From the above table it can be seen that there is a significant increase in the correlation coefficient if τ_f is increased from 60 to 600 seconds. This is confirmed by the time traces

above, where the traces coincide for the case where the filter time constant equals 600 seconds, but there is a lot of noise on the traces for the case of 60 seconds.

A further increase in τ_f from 600 to 6000 seconds only results in a small increase in the correlation coefficient. On the time traces it can be seen that the two traces do not coincide any better due to this higher τ_f. In fact dynamical information seems to be lost as all the dynamical changes are filtered out. Thus the filter time constant of 600 seconds (or 10 minutes) was assumed best and was for the rest of the tests.

Hence correlation tests were performed for all tests at different setpoints. In the table below the correlation coefficients are shown for different initial setpoints. The time plots and the correlation graphs for these test are shown in Appendix K.

Initial Setpoint	Correlation Coefficient
25%	0.9559
50%	0.9714
75%	0.9556
100%	0.9242

Table 6.2: Table of initial setpoints versus correlation coefficients

The test investigated was 10 000 seconds (2 hours 45 minutes) long. In the cross validation the initial 2000 seconds (33 minutes) were used to calibrate the instrument. These were determined to be the following for τ_f = 600s:

Y -intercept : 2389.90

Slope : 6.30

These calibration coefficients were then applied to the rest of the test in 2000 second intervals. As mentioned above the total estimated tons recorded and the error on this reading if it is compared to the true amount of tons flown past are of interest. The table below illustrates the results for each individual time period.

Time Interval [sec]	Total Tons [t]	Error [%]
0<=t<=2000	31.21	0.19
2000<=t<=4000	22.04	19.32
4000<=t<=6000	31.37	16.40
6000<=t<=8000	33.14	12.70
8000<=t<=10000	24.52	22.00
0<=t<=10000	142.27	13.40

Table 6.4: Table of cross validation results for v_0 = 100%

From these results it can be seen that the errors for the 2000 second intervals vary between 0 and 24 per cent. Extreme cases of the

error being 50 to 55% occurred in two instances, but these high error have little effect on the overall error measured, as the mass measured during this period is small compared to the total mass recorded.

It has been noted that the error of the total amount of tons over the whole period of the test is significantly lower than over the 2000 second intervals above. Errors for the whole interval varied between 2 to 13 percent, which is an improvement from the results above. This assumes that the calibration coefficients as determined in the first 2000 seconds is applied to the whole test.

calibration coefficients were assumed constant, i.e. accurate only for one particular initial speed setpoint. Two possible solutions do exist to this problem:

- Modelling the change in calibration coefficients with respect to the speed of the system.

- Filtering the responses with a filter that has a longer time constant and thus eliminates the dynamical changes.

The first possible solution can generally be ruled out, as determining the model of the calibration coefficients with respect to speed would greatly complicate the installation of a instrument.

Thus the second solution of filtering the envelopes with a filter of bigger time constant was investigated. The time plot where the MFs were filtered with a filter with a time constant of 6000 seconds is shown below for Test 1 and in Appendix K for Test 2:

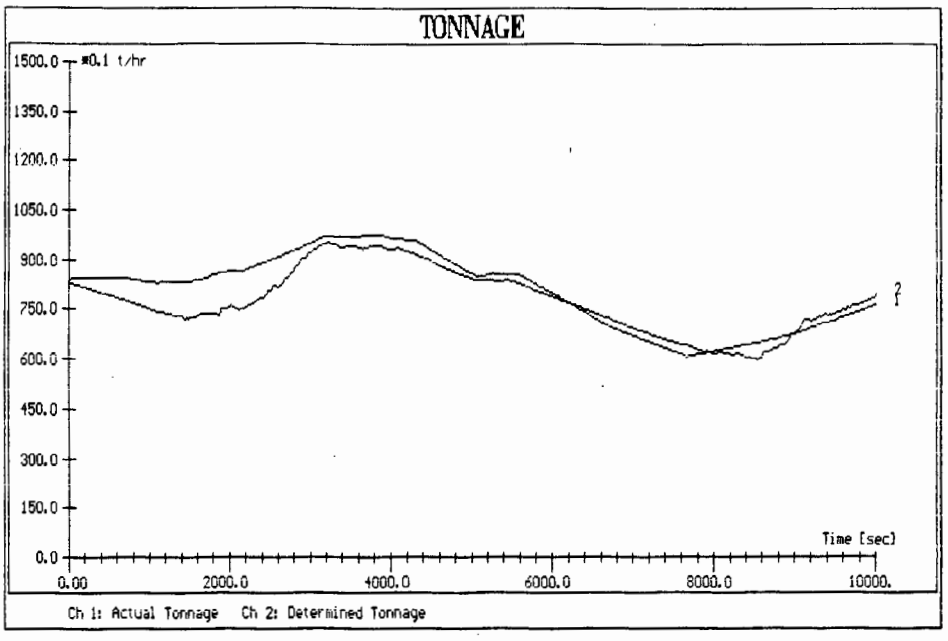


Figure 6.10: Time plot for Test 1 when speed is controlled; τ_f = 6000 s

The correlation coefficients for the two tests for both filter time constants are shown below. The actual correlation graphs for Test 1 and 2 are shown in Appendix K.

	τ_f [s]	
Test:	600	6000
1	0.8671	0.9305
2	0.9194	0.9367

Table 6.5: Table of correlation coefficients for different filter time constants

From the correlation coefficients it can be seen that there is a considerable improvement by filtering the results with a filter with a longer time constant. The disadvantage is however that the dynamics of the MF are lost.

The Cross Validation is also here an important aspect. It will be performed in a similar way as in the constant speed application. A filter time constant of 6000 seconds will be assumed for this cross validation, and the first 2000 seconds will be used to calibrate the instrument.

The results from the cross validation can thus be summarized in the following table for the two tests done:

	Test 1		Test 2	
Time Period [sec]	Total [t]	Error [%]	Total [t]	Error [%]
0<=t<=2000	45.81	0.41	70.55	-0.42
2000<=t<=4000	51.27	12.21	75.24	-1.43
4000<=t<=6000	48.22	17.46	77.55	1.567
6000<=t<=8000	36.92	9.39	75.70	-4.77
8000<=t<=10000	37.42	4.73	-	-
0<=t<=10000	216.80	11.27	297.03	-1.23

Table 6.6: Cross validation of MF results when speed is controlled

From the above table it can be seen that the errors vary between 0 to 17 % if the totals of mass from the estimation is compared to the actual tons flown past. Again the improvement of the error can be observed if the total is taken over a longer time period.

The calibration constants have not yet been

mentioned in the application of the speed being controlled. This is due to the fact that the calibration technique employed to calibrate the instrument in the instance of the speed being controlled needs, some explaining.

In the constant speed applications the calibration coefficients were obtained by using the correlation coefficients as a guide, thus obtaining a zero offset unity slope result from the regression if the calibrated MF estimates were correlated against the reference MFs. Using this technique in this application yielded no results. Thus another technique was thought of:

The Y-intercept of the calibration coefficients was obtained by running the belt empty for at least 10 minutes and applying the perturbation tests to the system. The average of the uncalibrated MF estimates equalled the Y-intercept of the calibration coefficients. The calibration slope was obtained from a typical test series like Test 1 or Test 2. The slope was determined on a trial and error basis until the total tons over a certain stretch of time were equal.

Thus the calibration results for the above tests can be summarized as follows for:

- the correlation results:

Test:	Y-intercept	Slope
Test 1	1714.00	17.38
Test 2	1485.10	23.75

Table 6.7: Table of calibration constants from correlation results

- the cross validation

Test:	Y-intercept	Slope
Test 1	2214.40	9.80
Test 2	1680.00	22.73

Table 6.8: Table of calibration constants from cross validation tests

It is obvious that considerable problems exist in calibrating such an instrument. Questions are posed how such an instrument could be calibrated in practise by using belt cuts. For such applications it is suggested that the instrument is calibrated after performing a large number of perturbation tests and thus taking the average to filter all the noise out. In deciding which speed to use to calibrate the system to it is suggested that the system is calibrated for 50% speed setpoint as this seemed to be the average around which the speed was fluctuating in a test situation.

Care must thus be taken when calibrating this instrument for varying speed applications.

CHAPTER 7:
SUMMARY AND CONCLUSIONS

7.1 SUMMARY

Massflow rates need to be measured on various process paths for good plant management. Conventional weightometers are however too expensive to warrant a widespread installation in mineral extraction processes.

This study was initiated to investigate more economic means to measure the massflow on conveyor belts. A specific objective of the project was to develop a low cost unit that should be easy to install on existing conveyor belts. The possible deterioration in accuracy over conventional methods was recognized, but in a widespread installation the effective precision of the low-cost unit could be enhanced by a mass smoothing package.

The technique employed to measure the massflow on a conveyor belt, firstly determines the absolute mass on a conveyor belt and then deduces the massflow by multiplying this mass by the speed of the belt.

The method to measure massflow in this investigation was adapted from a technique which determines the inertia of a system by means of a 'run-down' test. The direct analogy between mass in a translational frame of reference and inertia in a rotational frame of reference was established. The run-down test in

the translational frame of reference consists of switching the power to the system off and then studying the speed response of the system.

The original run-down test needed further adaptations to serve as an on-line mass estimator. These included using an energy model rather than a power model to measure the mass, because this reduced the amount of noise on the speed and power signals. In the energy model the energy converted to kinetic energy was proportional to the mass transported by the system. Thus the energy model could also be used to measure the mass from the starting response of a system, if the power response was measured as well.

By imposing a perturbation onto the system, the system was forced to slow down and then speed up again, and thus energy was converted to kinetic energy. The mass on the conveyor belt could be determined, by logging the speed and power responses during this perturbation, and thus determining the kinetic energy.

A few shortcomings were noted with this technique in that the power delivered *to* the motor was not proportional to the power delivered *by* the motor over a whole range of loads. This was due to the efficiency characteristics of the motor which are neither constant nor linear with speed. The power delivered by the motor could however be assumed proportional to the power supplied to the motor for loads above 40% of full load, as it was shown that both the efficiency and the power factor could be assumed constant at such loads.

A second shortcoming was that the exact load-force characteristics in terms of mass and speed of the conveyor belt had to be known to fully describe the model. It is difficult to determine such a

relationship in practice. Preliminary test showed, however, that the accuracy did not deteriorate if the load force was approximated by a constant value, provided the massflow was determined using the step-up response. The load force was hence established under steady state conditions before every test .

The measurement of speed requires that a tacho generator has to be specially installed on the conveyor belt. This is a disadvantage as it adds to the overall cost of the instrument and it makes the instrument difficult to install. Preliminary tests also showed that the speed needs not be measured, if the step-up response is used to determine the mass on the conveyor belt.

The basic statistics involved in calibrating such an instrument were also investigated. The basic procedure of calibration consists of the quantification and removal of the bias on a set of measurement data. It further includes the quantification of the imprecision associated with a measurement. For this project, the calibration was done by correlating the estimates obtained from the above technique against measurements from another reference massflow meter of higher accuracy.

In that way the bias could be determined by fitting a linear function through the correlation data points to yield the calibration coefficients in terms of y-intercept and slope.

The imprecision associated with a set of measurement is the spread in the readings on either side of the best fit line. The wider this spread the lower the accuracy. Thus statistical models were be used to describe the scatter of such readings, assuming that the scatter is of a random nature. Error bounds were

described, between which a certain percentage of the readings lie. The standard measure of imprecision used in industry is the 'standard deviation' which denotes the bounds between which 68% of the readings lie. In this study the bounds between which 95% of the readings lie has been chosen, which is equal to a error band twice the width of the standard deviation. Thus the results of imprecision in this study look worse than those normally quoted for accuracy. This is however only due to the percentage error bounds being chosen differently.

The error can not be assumed constant over a range of readings, as measurements near the centre of range of readings are described in much more detail than measurements near the extremes of the range of readings. The error bounds can thus not be assumed as a parallel line to the best fit line, but it should be modelled with an elliptical function, which will yield less error near the mean of all readings. The error thus stated will always be the width of the 95% confidence band near the mean of the range of readings, expressed as a percentage of full scale.

Dynamical effect had to be accounted for as well. This was done by filtering the data appropriately.

The software was written to implement the above mass measuring technique and to calibrate the system appropriately to a reference massflow reading. This software was also to serve as a PC based prototype instrument which was to be used on a mine. Options had to be included to test the technique for various test conditions.

The software can be divided into three groups of software packages. The first group consisted of a real-time response measuring package (DLOG), a

graphical response display package (GRAF) and a massflow correlation package obtaining the speed and power responses manually from a file (MFCM). This group of packages was used to validate the technique and to perform preliminary test. The second group consisted of a massflow correlation package, which obtained the speed and power responses automatically from the actual plant in an on-line fashion (MFCA). This package was used to calibrate the prototype instrument technique against a reference massflow meter. The last group consisted of the actual prototype massflow meter (MF), which included the finally proposed technique to measure the massflow on a horizontal conveyor belt.

The massflow measuring technique was then tested firstly in a laboratory environment and then in a true industrial environment.

The tests performed on a laboratory test rig included validating the technique for different modes of operation. Two ways to perturb the speed of the system to produce a change in kinetic energy were investigated. The first uses a Variable Speed Drive (VSD) to perturb the speed setpoint of the system while the second uses Solid State Relays to cut the power to the system at a specific instance in time.

A VSD has the advantage that the speed can relatively easy be perturbed as the frequency of the VSD can be altered by adding a negative voltage to the frequency control setpoint signal. An added advantage is that the system needs not be stopped totally, but can also just be slowed down.

As one of the objectives to this study was that the final instrument be a low cost instrument, the use of a VSD alone for the application of measuring massflow

would not be justified, as the price of a VSD lies in the same region as a conventional weightometer. The use of the technique based on a VSD is thus only justified for applications where such a device is installed already for other purposes. The additional cost related to installing such a massflow measuring device can thus be considered as low cost.

The test performed for the system being perturbed with a VSD comprised investigating the system for different size speed steps applied to the system, and investigating the calibration coefficients for the system running at various initial speed setpoints.

Test results from the system being perturbed with a VSD showed that the best accuracy could be obtained for the system running at full speed and then being perturbed to stop. The error associated with the reading equalled 5.3%. Applying smaller steps this accuracy decreased slowly to 24.1% when a step of only 25% was applied.

Running the VSD at different speed setpoints yielded calibration coefficients that changed, as the speed setpoint changed. This created the problem that the calibration is different for each speed that the system is operating at if the speed is varying. One solution to the problem would be to model the calibration coefficients with respect to speed, which would however complicate the calibration process. A second solution proposed that the results be filtered by a low pass filter with a long time constant. In this way all dynamic changes in the speed setpoint were filtered out.

Perturbing the system with SSRs would be a more economic option for those conveyor belts where the use of a VSD is not justified. Disadvantages however

included that the system could not be perturbed as easily as in the case of the VSD. The system could only be controlled to run or switched off. The period for which the system was switched off could however be controlled. As the system required some time to stop after the power was cut, the system could be switched on again before it actually stopped. In this way the influence of the perturbation on the system could be reduced.

When the system was operated with SSRs the precision was not as good as in the case of the system being perturbed with a VSD. The imprecision for the system being forced to stop totally equalled 20.7% for the massflow being determined from the step-down response. For the case of the power being cut only for a short duration of time and thus speeding the system up again the imprecision equalled 22.1%. It was noted that the imprecision of the measurements as determined from the step-up response were unacceptably large in comparison to the results from the step-down response.

Generally it can be stated that the accuracy for the system being perturbed with SSRs was not as good as that for the system being perturbed with a VSD. However the lower cost factor of the technique using SSRs made it a viable option to consider for applications where a VSD was not installed.

Due to the higher accuracies obtained from the system using a VSD for the perturbation, this technique was investigated in more depth.

Two aspects were of special importance: Firstly the cross validation of data and the total tons recorded for a specific period of time and secondly the effect of filtering. The cross-validation was done by choosing a test, which took a couple of hours. This

test was then divided into five equal time intervals. One of these time intervals was used to calibrate the system appropriately. The calibration coefficients from this calibration were then applied to the other intervals thereby simulating the practical operation of the instrument. The error was measured by integrating the massflow signal over that specific period of time, and then comparing the total tons recorded for that period of time against the reference mass measurement for the same period of time.

Results from such a cross validation being performed on a set of data measured in the laboratory environment resulted in the error between the estimated tons recorded and the actual tons recorded being between 0% and 3%. An extreme case of the error being 20% occurred at very low massflow rates, which thus did not have much effect in relation to the total amount of tons recorded. The error also decreased if the total tons were recorded over longer periods of time.

Filtering the measurement data reduced the noise level on the measurements. As the dynamics of the massflow estimation system were similar to that of the reference massflow meter, the filtering did not have much effect.

Following this initial validation of the massflow measurement technique a prototype was proposed, which was then implemented on a mine. The technique of using the VSD was proposed, because of the superior accuracies obtained where using this technique. Thus it was assumed in the implementation that a VSD was already installed. The instrument that was proposed consisted of a micro-processor based 'Smart Box', a power transducer and an adding circuit. The total cost of this instrument (excluding the price of a VSD)

was determined to be R3,410.- in July 1990 prices, which indicates a considerable saving against conventional weightometers ranging from R 17,000.-.

The tests performed on the mine always assumed a full perturbation, i.e. the system was forced to stop. It also had to be kept in mind that the reference massflow meter on the process stream was separated by a crusher to the horizontal conveyor belt. Thus the dynamics of the crusher had to be accounted for. This was done by filtering the results with a filter that had a time constant longer than that of the crusher and the measurement devices.

The system was first investigated for a constant initial speed application. The filter time constant chosen equalled 600 seconds, which was determined from experimental test. A longer time constant did not bring any further improvement. Direct correlation results indicated correlation coefficients of between 0.92 and 0.97 for different initial setpoints. A figure for the imprecision of the system would not have yielded any results due to the dynamics of the crusher. The test did however indicate that the calibration constants varied for different initial setpoints of speed.

The cross validation results gave an indication of the accuracy of the massflow estimation technique. Performing the test in a similar fashion as for the laboratory test rig, the error between estimated and actual massflow varied between 0% and 24% for specific periods of time. Extremes also occurred during the cross validation, but these cases had little effect on the overall error as the massflow rates measured during these periods were small. The errors improved to a range of 0% to 13% if the tons recorded were considered for longer periods of time.

The system being operated under controlled speed was of special interest. A direct correlation between the actual and the estimated massflow rates showed less correlation than before. This indicated a loss of correlation due to the inconsistent calibration coefficients at different initial speed setpoints. A way to overcome this problem was to filter the results more. This resulted in the dynamical changes being lost, but the error results improving. The filter time constant chosen equalled 6000 seconds.

Cross Validation results indicated the errors for specific time intervals to lie between 0% and 17%. This error improved to lie between 0% and 11% if the total tons recorded for a specific period of time was logged for a longer period of time. Care has to be taken however when calibrating such a system which has such a long time constant.

7.2 CONCLUSIONS

In conclusion the following can be stated:

- The technique to measure the massflow by determining the kinetic energy converted by the system during speed perturbation was proved to yield sensible results.

- The application of using the VSD yielded better results than the using SSRs. The high cost of a VSD limits this application to conveyor belts where a VSD is installed already.

- The accuracies obtained from constant speed applications were better than the ones obtained from varying speed applications. This problem could however be solved by filtering the massflow signal appropriately.

Specifically, for an industrial application two options exist:

- Where a VSD is installed the mass can be estimated to an accuracy of 13% over a three hour average in a constant speed application. Instantaneous massflows can be determined to an accuracy of 12% (assuming a 68% error band and assuming a filter with a 10 minute time constant). In a case where the speed is varying the mass can be determined to an accuracy of 11% over a three hour average.

- If a VSD is not installed the SSR provides an economic alternative. Here the massflow can be determined to an accuracy of 11% in a

laboratory environment (assuming a 68% error band).

The above results make this technique viable for inclusions on a plant wide scale.

7.3 RECOMMENDATIONS FOR FUTURE WORK

Following the conclusions of this thesis the following recommendations are made:

- Construct a micro-processor based prototype instrument as suggested in chapter 5.

- Install such an instrument on a mine and test it for extended periods of time.

- Use the results from such a instrument and feed them into a mass balancing package, to improve the resulting accuracy of the measurement.

List of References

[1] - Considine and Ross - **Handbook of Applied Instrumentation**; McGraw-Hill; New York, 1964

[2] - Dierks, K C A - **An Investigation into the possibility to determine the Inertia of an Induction Motor**; B.Sc. Thesis; University of Cape Town, 1988

[3] - Dierks, K C A - **Patent: 'The Measurement of Massflow on a Conveyor Belt'**; Johannesburg, 1989

[4] - Doebelin - **Measurement Systems: Application and Design**; McGraw-Hill; New York, 1966

[5] - Dover, A T - **Theory and Practice of Alternating Currents**; 3rd Edition; Sir Isaac Pitman & Sons Ltd.; London, 1965

[6] - EEE417 Course Notes on **Control Engineering II**, (Prof. M. Braae); University of Cape Town; 1988

[7] - Halliday, Resnick - **Fundamentals of Physics, 2nd Edition**; Wiley; New York, 1981

[8] - Horowitz, Hill - **The Art of Electronics**, Cambridge University Press; Cambridge, 1981

[9] - Leonard, W - **Control of Electrical Drives**; Springer Verlag; Berlin, 1985

[10] - **Linear 1 Databook** - National Semiconductor Corporation; 1988 Ed.; pg. 2-482

[11] - Miller, Rupert G jr. - **Simultaneous Statistical Inference**; McGraw-Hill; New York, 1966

[12] - Mitchell, J R - **How to weigh bulk solid materials**;

Chemical Engineer (vol: 85, no.24); Oct 1978

[13] - Potgieter J H, Joseph I - **Unique Conveyor Massflow measurement and Mass Balancing Technique**; 1st IFAC Workshop on Applied Measurements in Mineral and Metallurgical Processing; Sabi Sabi, R.S.A., 1988

[14] - Say, M G - **Performance and Design of Alternating Current Machines**; 3rd Edition; Sir Isaac Pitman & Sons Ltd.; London 1965

[15] - **Turbo Pascal Version 4.0** - Owner's Handbook; Borland International; Scotts Valley, U.S.A., 1987

[16] - Volk, W - **Engineering Statistics with a Programmable Calculator**; McGraw-Hill; New York, 1982

Bibliography

[1] - Colijn, H - **Weighing and Proportioning of Bulk Solids**; First Edition; Trans Tech Publications; 1975

[2] - Considine and Ross - **Handbook of Applied Instrumentation**; McGraw-Hill; New York, 1964

[3] - Dierks, K C A - **An Investigation into the possibility to determine the Inertia of an Induction Motor**; B.Sc. Thesis; University of Cape Town, 1988

[4] - Dierks, K C A - **Patent: 'The Measurement of Massflow on a Conveyor Belt'**; Johannesburg, 1989

[5] - Doebelin - **Measurement Systems: Application and Design**; McGraw-Hill; New York, 1966

[6] - Dover, A T - **Theory and Practice of Alternating Currents**; 3rd Edition; Sir Isaac Pitman & Sons Ltd.; London, 1965

[7] - EEE 417 Course Notes on **Control Engineering II**, (Prof. M. Braae); University of Cape Town, 1988

[8] - EEE 331 Course Notes on **Energy Utilization** (Prof S.G. McLaren); University of Cape Town, 1987

[9] - EEE 328 Course Notes on **Energy Conversion** (Dr R.E.Neubauer); University of Cape Town, 1987

[10] - Geigy, J R - **Documenta Geigy Scientific Tables**; 6th Edition; S.A., 1962

[11] - Halliday, Resnick - **Fundamentals of Physics, 2nd Edition**; Wiley; New York, 1981

[12] - Horowitz, Hill - **The Art of Electronics**, Cambridge University Press; Cambridge, 1981

[13] - Isemann, R (editor) - **Identification and System Parameter Estimation, Volume 1**; IFAC 1979

[14] - Keeping, E S - **Introduction to Statistical Interference**; D. van Nostrand Company Inc.; 1962

[15] - Leonard, W - **Control of Electrical Drives**; Springer Verlag; Berlin, 1985

[16] - **Linear 1 Databook** - National Semiconductor Corporation; 1988 Ed.; pg. 2-482

[17] - Miller, Rupert G jr. - **Simultaneous Statistical Inference**; McGraw-Hill; New York, 1966

[18] - Mitchell, J R - **How to weigh bulk solid materials**; Chemical Engineer (vol: 85, no.24); Oct 1978

[19] - Pearson E S, Hartley H O (editors) - **Biomedical Tables for Statistics, Volume 1**; Cambridge University Press; Cambridge, 1962

[20] - **Physic Notes for EEE 104W course**; University of Cape Town, 1985

[21] - **Physic Notes for EEE 222W course**; University of Cape Town, 1985

[22] - Potgieter J H, Joseph I - **Unique Conveyor Massflow measurement and Mass Balancing Technique**; 1st IFAC Workshop on Applied Measurements in Mineral and Metallurgical Processing; Sabi Sabi, R.S.A., 1988

[23] - Say, M G - **Performance and Design of Alternating Current Machines**; 3rd Edition; Sir Isaac Pitman &

Sons Ltd.; London 1965

[24] - Stevens, R E - **Electrical Machines and Power Electronics**; Van Nostrand Reinhold; Berkshire, 1984

[25] - Richards, R J - **An Introduction to Dynamics and Control**; Longman Scientific and Technical; Essex, 1979

[26] - **Turbo Pascal Version 4.0** - Owner's Handbook; Borland International; Scotts Valley, U.S.A., 1987

[27] - Volk, W - **Engineering Statistics with a Programmable Calculator**; McGraw-Hill; New York, 1982

APPENDIX A: INCORPORATING THE LOAD CHARACTERISTICS IN THE MASSFLOW ESTIMATION ALGORITHM

As mentioned before the load force F_l will be modelled with a higher order polynomial with regard to speed v and mass M on the CB:

$$F_l = F_0 + a*v + b*v^2 + c*M + d*M^2$$

The parameter F_0, a, b, c and d can be determined from a multi-linear regression between v,M and F_l for different speeds and masses on the CB.

To model the load force of the system the speed and the mass on the CB need to be known. The mass on the CB is, however, not known, and thus the terms 'c*M' and 'd*M^2' can not be determined. It is also assumed that the term F_0 is not constant with time, and hence this term will also be regarded as unknown. The coefficients of speed a and b are however assumed constant with time. Collecting all the unknowns into one variable F_u the above equation becomes:

$$F_l = F_u + a*v + b*v^2$$

The test sequence and the timing used in this procedure is the same as described in section 2.5. F_l can be determined as before in a initial period $t_0<=t<=t_1$. Thus F_u can be

calculated if the speed of the system is known.

Hence the energy converted into frictional energy has to be determined:

$$E_f = \int F_1(m,v) * v(t)\ dt$$

$$E_f = \int (F_u + a*v + b*v^2) * v(t)\ dt$$

$$E_f = \int F_u*v(t)\ dt + \int a*v^2(t)\ dt + \int b*v^3(t)\ dt$$

$$E_f = F_u* x_{12} + a* \int v^2(t)dt + b* \int v^3(t)dt$$

The integrals of v^2 and v^3 can only be determined numerically from the speed signal as an analytical solution does not exist. The distance x_{12} covered in a time span $t_1<=t<=t_2$ can be obtained by integrating the speed signal.

The equations to determine the mass read as follow:

- for the step-down response:

$$M_{dn} = \frac{(F_u * x_{12} + a * \int_{t1}^{t2} v^2 dt + b * \int_{t1}^{t2} v^3 dt)}{1/2\ (v_1^2 - v_0^2)}$$

- and from the step-up response

$$M_{up} = \frac{\int_{t3}^{t4} P_d\ dt - (F_u * x_{12} + a * \int_{t3}^{t4} v^2 dt + b * \int_{t3}^{t4} v^3 dt)}{1/2\ (v_0^2 - v_1^2)}$$

APPENDIX B:

TABLE OF F-DISTRIBUTION NUMBERS

(Reproduced from J.R.Geigy; **Documanta Geigy, Scientific Tables, 6th Edition**; S.A. 1962)

Upper Significance Limits of the *F*-Distribution[1] $P = 0.05$

$P = P(\text{right}) = \text{integral between } F \text{ and infinity}$

ν_2 \ ν_1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	22	24	26	28	30	40	50	60	80	100
1	161.44	200	216	225	230	234	237	239	241	242	243	244	245	245	246	246	247	248	248	248	249	249	250	250	250	251	252	252	252	253
2	18.51	19.0	19.2	19.2	19.3	19.3	19.4	19.4	19.4	19.4	19.4	19.4	19.4	19.4	19.4	19.4	19.4	19.4	19.4	19.4	19.5	19.5	19.5	19.5	19.5	19.5	19.5	19.5	19.5	19.5
3	10.13	9.55	9.28	9.12	9.01	8.94	8.89	8.85	8.81	8.79	8.76	8.74	8.73	8.71	8.70	8.69	8.68	8.67	8.67	8.66	8.65	8.64	8.63	8.62	8.62	8.59	8.58	8.57	8.56	8.55
4	7.71	6.94	6.59	6.39	6.26	6.16	6.09	6.04	6.00	5.96	5.94	5.91	5.89	5.87	5.86	5.84	5.83	5.82	5.81	5.80	5.79	5.77	5.76	5.75	5.75	5.72	5.70	5.69	5.67	5.66
5	6.61	5.79	5.41	5.19	5.05	4.95	4.88	4.82	4.77	4.74	4.70	4.68	4.66	4.64	4.62	4.60	4.59	4.58	4.57	4.56	4.54	4.53	4.52	4.50	4.50	4.46	4.44	4.43	4.41	4.41
6	5.99	5.14	4.76	4.53	4.39	4.28	4.21	4.15	4.10	4.06	4.03	4.00	3.98	3.96	3.94	3.92	3.91	3.90	3.88	3.87	3.86	3.84	3.83	3.82	3.81	3.77	3.75	3.74	3.72	3.71
7	5.59	4.74	4.35	4.12	3.97	3.87	3.79	3.73	3.68	3.64	3.60	3.57	3.55	3.53	3.51	3.49	3.48	3.47	3.46	3.44	3.43	3.41	3.40	3.38	3.38	3.34	3.32	3.30	3.29	3.27
8	5.32	4.46	4.07	3.84	3.69	3.58	3.50	3.44	3.39	3.35	3.31	3.28	3.26	3.24	3.22	3.20	3.19	3.17	3.16	3.15	3.13	3.12	3.10	3.09	3.08	3.04	3.02	3.01	2.99	2.97
9	5.12	4.26	3.86	3.63	3.48	3.37	3.29	3.23	3.18	3.14	3.10	3.07	3.05	3.03	3.01	2.99	2.97	2.96	2.95	2.94	2.92	2.90	2.89	2.87	2.86	2.83	2.80	2.79	2.77	2.76
10	4.96	4.10	3.71	3.48	3.33	3.22	3.14	3.07	3.02	2.98	2.94	2.91	2.89	2.85	2.86	2.85	2.83	2.81	2.80	2.78	2.77	2.75	2.74	2.72	2.71	2.70	2.66	2.64	2.62	2.60
11	4.84	3.98	3.59	3.36	3.20	3.09	3.01	2.95	2.90	2.85	2.82	2.79	2.76	2.74	2.72	2.70	2.69	2.68	2.66	2.65	2.63	2.61	2.60	2.58	2.57	2.53	2.51	2.49	2.47	2.46
12	4.75	3.89	3.49	3.26	3.11	3.00	2.91	2.85	2.80	2.75	2.72	2.69	2.66	2.64	2.62	2.60	2.58	2.57	2.56	2.54	2.52	2.51	2.49	2.48	2.47	2.43	2.40	2.38	2.36	2.35
13	4.67	3.81	3.41	3.18	3.03	2.92	2.83	2.77	2.71	2.67	2.63	2.60	2.58	2.55	2.53	2.51	2.50	2.48	2.47	2.46	2.44	2.42	2.41	2.39	2.38	2.34	2.31	2.30	2.27	2.26
14	4.60	3.74	3.34	3.11	2.96	2.85	2.76	2.70	2.65	2.60	2.57	2.53	2.51	2.48	2.46	2.44	2.43	2.41	2.40	2.39	2.37	2.36	2.35	2.33	2.32	2.27	2.24	2.22	2.20	2.19
15	4.54	3.68	3.29	3.06	2.90	2.79	2.71	2.64	2.59	2.54	2.51	2.48	2.45	2.42	2.40	2.38	2.37	2.35	2.34	2.33	2.31	2.29	2.28	2.27	2.25	2.20	2.18	2.16	2.14	2.12
16	4.49	3.63	3.24	3.01	2.85	2.74	2.66	2.59	2.54	2.49	2.46	2.42	2.40	2.37	2.35	2.33	2.32	2.30	2.29	2.28	2.25	2.24	2.22	2.21	2.20	2.15	2.12	2.11	2.08	2.07
17	4.45	3.59	3.20	2.96	2.81	2.70	2.61	2.55	2.49	2.45	2.41	2.38	2.35	2.33	2.31	2.29	2.27	2.26	2.24	2.23	2.22	2.19	2.17	2.16	2.15	2.10	2.08	2.06	2.04	2.02
18	4.41	3.55	3.16	2.93	2.77	2.66	2.58	2.51	2.46	2.41	2.37	2.34	2.31	2.29	2.27	2.25	2.23	2.22	2.20	2.19	2.17	2.15	2.13	2.12	2.11	2.06	2.04	2.02	1.99	1.98
19	4.38	3.52	3.13	2.90	2.74	2.63	2.54	2.48	2.42	2.38	2.34	2.31	2.28	2.26	2.23	2.21	2.20	2.18	2.17	2.16	2.13	2.11	2.10	2.08	2.07	2.03	2.00	1.98	1.96	1.94
20	4.35	3.49	3.10	2.87	2.71	2.60	2.51	2.45	2.39	2.35	2.31	2.28	2.25	2.22	2.20	2.18	2.17	2.15	2.14	2.12	2.10	2.08	2.07	2.05	2.04	2.01	1.99	1.97	1.95	1.93
21	4.32	3.47	3.07	2.84	2.68	2.57	2.49	2.42	2.37	2.32	2.28	2.25	2.22	2.20	2.18	2.16	2.14	2.12	2.11	2.10	2.07	2.05	2.04	2.02	2.01	1.96	1.94	1.92	1.90	1.88
22	4.30	3.44	3.05	2.82	2.66	2.55	2.46	2.40	2.34	2.30	2.26	2.23	2.20	2.17	2.15	2.13	2.11	2.10	2.08	2.07	2.05	2.03	2.02	2.00	1.99	1.94	1.91	1.90	1.88	1.86
23	4.28	3.42	3.03	2.80	2.64	2.53	2.44	2.37	2.32	2.27	2.24	2.20	2.18	2.15	2.13	2.11	2.09	2.08	2.06	2.05	2.02	2.00	1.99	1.97	1.96	1.91	1.88	1.86	1.84	1.82
24	4.26	3.40	3.01	2.78	2.62	2.51	2.42	2.36	2.30	2.25	2.21	2.18	2.15	2.13	2.11	2.09	2.07	2.05	2.04	2.03	2.00	1.98	1.97	1.95	1.95	1.89	1.86	1.84	1.82	1.80
25	4.24	3.39	2.99	2.76	2.60	2.49	2.40	2.34	2.28	2.24	2.20	2.16	2.14	2.11	2.09	2.07	2.05	2.04	2.02	2.01	1.98	1.96	1.95	1.93	1.93	1.87	1.84	1.82	1.80	1.78
26	4.23	3.37	2.98	2.74	2.59	2.47	2.39	2.32	2.27	2.22	2.18	2.15	2.12	2.09	2.07	2.05	2.03	2.02	2.00	1.99	1.97	1.95	1.93	1.91	1.90	1.85	1.82	1.80	1.78	1.76
27	4.21	3.35	2.96	2.73	2.57	2.46	2.37	2.31	2.25	2.20	2.17	2.13	2.10	2.08	2.06	2.04	2.02	2.00	1.99	1.97	1.95	1.93	1.91	1.90	1.88	1.84	1.81	1.79	1.77	1.74
28	4.20	3.34	2.95	2.71	2.56	2.45	2.36	2.29	2.24	2.19	2.15	2.12	2.09	2.06	2.04	2.02	2.00	1.99	1.97	1.96	1.94	1.92	1.91	1.89	1.87	1.82	1.79	1.77	1.75	1.73
29	4.18	3.33	2.93	2.70	2.55	2.43	2.35	2.28	2.22	2.18	2.14	2.10	2.08	2.05	2.03	2.01	1.99	1.98	1.96	1.95	1.92	1.91	1.88	1.88	1.85	1.81	1.77	1.75	1.73	1.71
30	4.17	3.32	2.92	2.69	2.53	2.42	2.33	2.27	2.21	2.16	2.13	2.09	2.06	2.04	2.01	1.99	1.98	1.96	1.95	1.94	1.91	1.89	1.87	1.86	1.84	1.79	1.76	1.74	1.71	1.70
32	4.15	3.29	2.90	2.67	2.51	2.40	2.31	2.24	2.19	2.14	2.10	2.07	2.04	2.01	1.99	1.97	1.95	1.94	1.92	1.91	1.88	1.86	1.85	1.85	1.82	1.77	1.74	1.71	1.69	1.67
34	4.13	3.28	2.88	2.65	2.49	2.36	2.29	2.23	2.17	2.12	2.08	2.05	2.02	1.99	1.97	1.95	1.93	1.92	1.90	1.89	1.86	1.84	1.82	1.82	1.80	1.75	1.71	1.69	1.66	1.65
36	4.11	3.26	2.87	2.63	2.48	2.36	2.28	2.21	2.15	2.11	2.07	2.03	2.00	1.98	1.95	1.93	1.92	1.90	1.88	1.87	1.85	1.82	1.81	1.79	1.78	1.73	1.69	1.67	1.64	1.62
38	4.10	3.24	2.85	2.62	2.46	2.35	2.26	2.19	2.14	2.09	2.05	2.02	1.99	1.96	1.94	1.92	1.90	1.88	1.87	1.85	1.83	1.81	1.79	1.77	1.76	1.71	1.68	1.65	1.62	1.61
40	4.08	3.23	2.84	2.61	2.45	2.34	2.25	2.18	2.12	2.08	2.04	2.00	1.97	1.95	1.92	1.90	1.89	1.87	1.85	1.84	1.81	1.79	1.77	1.76	1.74	1.69	1.66	1.64	1.61	1.59
42	4.07	3.22	2.83	2.59	2.44	2.32	2.24	2.17	2.11	2.06	2.03	1.99	1.96	1.93	1.91	1.89	1.87	1.86	1.84	1.83	1.80	1.78	1.76	1.74	1.73	1.68	1.65	1.62	1.59	1.57
44	4.06	3.21	2.82	2.58	2.43	2.31	2.23	2.16	2.10	2.05	2.01	1.98	1.95	1.92	1.90	1.88	1.86	1.84	1.83	1.81	1.79	1.77	1.75	1.73	1.72	1.67	1.63	1.61	1.58	1.56
46	4.05	3.20	2.81	2.57	2.42	2.30	2.22	2.15	2.09	2.04	2.00	1.97	1.94	1.91	1.89	1.87	1.85	1.83	1.82	1.80	1.78	1.76	1.74	1.72	1.71	1.65	1.62	1.60	1.57	1.55
48	4.04	3.19	2.80	2.57	2.41	2.29	2.21	2.14	2.08	2.03	1.99	1.96	1.93	1.91	1.88	1.86	1.84	1.82	1.81	1.79	1.77	1.75	1.73	1.71	1.70	1.64	1.61	1.59	1.56	1.54
50	4.03	3.18	2.79	2.56	2.40	2.29	2.20	2.13	2.07	2.03	1.99	1.95	1.92	1.89	1.87	1.85	1.83	1.81	1.80	1.78	1.76	1.74	1.72	1.70	1.69	1.63	1.60	1.58	1.54	1.52
60	4.00	3.15	2.76	2.53	2.37	2.25	2.17	2.10	2.04	1.99	1.95	1.92	1.89	1.86	1.84	1.82	1.80	1.78	1.76	1.75	1.72	1.70	1.68	1.66	1.65	1.59	1.56	1.53	1.50	1.48
70	3.98	3.13	2.74	2.50	2.35	2.23	2.14	2.07	2.02	1.97	1.93	1.89	1.86	1.84	1.81	1.79	1.77	1.75	1.74	1.72	1.70	1.67	1.65	1.64	1.64	1.57	1.53	1.50	1.47	1.45
80	3.96	3.11	2.72	2.49	2.33	2.21	2.13	2.06	2.00	1.95	1.91	1.88	1.84	1.82	1.79	1.77	1.75	1.73	1.72	1.70	1.68	1.65	1.63	1.62	1.62	1.54	1.51	1.48	1.45	1.43
90	3.95	3.10	2.71	2.47	2.32	2.20	2.11	2.04	1.99	1.94	1.90	1.86	1.83	1.80	1.78	1.76	1.74	1.72	1.70	1.69	1.66	1.64	1.62	1.60	1.59	1.53	1.49	1.46	1.43	1.41
100	3.94	3.09	2.70	2.46	2.31	2.19	2.10	2.03	1.97	1.93	1.89	1.85	1.82	1.79	1.77	1.75	1.73	1.71	1.69	1.68	1.65	1.63	1.61	1.59	1.57	1.52	1.48	1.45	1.41	1.39
125	3.92	3.07	2.68	2.44	2.29	2.17	2.08	2.01	1.96	1.91	1.87	1.83	1.80	1.77	1.75	1.73	1.71	1.69	1.67	1.66	1.63	1.60	1.58	1.56	1.55	1.50	1.46	1.43	1.39	1.36
150	3.90	3.06	2.66	2.43	2.27	2.16	2.07	2.00	1.94	1.89	1.85	1.82	1.79	1.76	1.74	1.72	1.69	1.67	1.66	1.64	1.62	1.60	1.57	1.55	1.53	1.48	1.46	1.41	1.37	1.34
200	3.89	3.04	2.65	2.42	2.26	2.14	2.06	1.98	1.93	1.88	1.84	1.80	1.77	1.74	1.72	1.69	1.67	1.66	1.64	1.62	1.60	1.57	1.55	1.53	1.52	1.46	1.41	1.39	1.35	1.32
300	3.87	3.03	2.63	2.40	2.24	2.13	2.04	1.97	1.91	1.86	1.82	1.78	1.75	1.72	1.70	1.68	1.66	1.64	1.62	1.61	1.58	1.55	1.53	1.51	1.50	1.43	1.39	1.36	1.32	1.30
500	3.86	3.01	2.62	2.39	2.23	2.12	2.03	1.96	1.90	1.85	1.81	1.77	1.74	1.71	1.69	1.66	1.64	1.62	1.61	1.59	1.56	1.54	1.52	1.51	1.49	1.42	1.38	1.34	1.30	1.28
1000	3.85	3.00	2.61	2.38	2.22	2.11	2.02	1.95	1.89	1.84	1.80	1.76	1.73	1.70	1.68	1.65	1.63	1.61	1.60	1.58	1.55	1.53	1.51	1.49	1.47	1.41	1.36	1.33	1.29	1.26

1) Values from VAN DER WAERDEN, B. L., *Mathematische Statistik*, Springer, Berlin, 1957, page 340. Reprinted by kind permission of the author and publishers.

Upper Significance Limits of the *F*-Distribution[1] $P = 0.01$

$P = P$ (right) = integral between F and infinity

ν_2 \ ν_1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	22	24	26	28	30	40	50	60	80	100
2	98.50	99.0	99.2	99.2	99.3	99.3	99.4	99.4	99.4	99.4	99.4	99.4	99.4	99.4	99.4	99.4	99.4	99.4	99.4	99.4	99.5	99.5	99.5	99.5	99.5	99.5	99.5	99.5	99.5	99.5
3	34.12	30.8	29.5	28.7	28.2	27.9	27.7	27.5	27.3	27.2	27.1	27.1	27.0	26.9	26.9	26.9	26.8	26.8	26.7	26.7	26.6	26.6	26.6	26.5	26.5	26.4	26.4	26.3	26.3	26.2
4	21.20	18.0	16.7	16.0	15.5	15.2	15.0	14.8	14.7	14.5	14.4	14.4	14.3	14.2	14.2	14.2	14.2	14.1	14.0	14.0	14.0	13.9	13.9	13.9	13.8	13.7	13.7	13.7	13.6	13.6
5	16.26	13.3	12.1	11.4	11.0	10.7	10.5	10.3	10.2	10.1	9.96	9.89	9.82	9.77	9.72	9.68	9.64	9.61	9.58	9.55	9.51	9.47	9.43	9.40	9.38	9.29	9.24	9.20	9.16	9.13
6	13.75	10.9	9.78	9.15	8.75	8.47	8.26	8.10	7.98	7.87	7.79	7.72	7.66	7.60	7.56	7.52	7.48	7.45	7.42	7.40	7.35	7.31	7.28	7.25	7.23	7.14	7.09	7.06	7.01	6.99
7	12.25	9.55	8.45	7.85	7.46	7.19	6.99	6.84	6.72	6.62	6.54	6.47	6.41	6.36	6.31	6.27	6.24	6.21	6.18	6.16	6.11	6.07	6.04	6.02	5.99	5.91	5.86	5.82	5.78	5.75
8	11.26	8.65	7.59	7.01	6.63	6.37	6.18	6.03	5.91	5.81	5.73	5.67	5.61	5.56	5.52	5.48	5.44	5.41	5.38	5.36	5.32	5.28	5.25	5.22	5.20	5.12	5.07	5.03	4.99	4.96
9	10.56	8.02	6.99	6.42	6.06	5.80	5.61	5.47	5.35	5.26	5.18	5.11	5.05	5.00	4.96	4.92	4.89	4.86	4.83	4.81	4.77	4.73	4.70	4.67	4.65	4.57	4.52	4.48	4.44	4.42
10	10.04	7.56	6.55	5.99	5.64	5.39	5.20	5.06	4.94	4.85	4.77	4.71	4.65	4.60	4.56	4.52	4.49	4.46	4.43	4.41	4.36	4.33	4.30	4.27	4.25	4.17	4.12	4.08	4.04	4.01
11	9.65	7.21	6.22	5.67	5.32	5.07	4.89	4.74	4.63	4.54	4.46	4.40	4.34	4.29	4.25	4.21	4.18	4.15	4.12	4.10	4.06	4.02	3.99	3.96	3.94	3.86	3.81	3.78	3.73	3.71
12	9.33	6.93	5.95	5.41	5.06	4.82	4.64	4.50	4.39	4.30	4.22	4.16	4.10	4.05	4.01	3.97	3.94	3.91	3.88	3.86	3.82	3.78	3.75	3.72	3.70	3.62	3.57	3.54	3.49	3.47
13	9.07	6.70	5.74	5.21	4.86	4.62	4.44	4.30	4.19	4.10	4.02	3.96	3.91	3.86	3.82	3.78	3.75	3.72	3.69	3.66	3.62	3.59	3.56	3.53	3.51	3.43	3.38	3.34	3.30	3.27
14	8.86	6.51	5.56	5.04	4.69	4.46	4.28	4.14	4.03	3.94	3.86	3.80	3.75	3.70	3.66	3.62	3.59	3.56	3.53	3.51	3.46	3.43	3.40	3.37	3.35	3.27	3.22	3.18	3.14	3.11
15	8.68	6.36	5.42	4.89	4.56	4.32	4.14	4.00	3.89	3.80	3.73	3.67	3.61	3.56	3.52	3.49	3.45	3.42	3.40	3.37	3.33	3.29	3.26	3.24	3.21	3.13	3.08	3.05	3.00	2.98
16	8.53	6.23	5.29	4.77	4.44	4.20	4.03	3.89	3.78	3.69	3.62	3.55	3.50	3.45	3.41	3.37	3.34	3.31	3.28	3.26	3.22	3.18	3.15	3.12	3.10	3.02	2.97	2.93	2.89	2.86
17	8.40	6.11	5.18	4.67	4.34	4.10	3.93	3.79	3.68	3.59	3.52	3.46	3.40	3.35	3.31	3.27	3.24	3.21	3.18	3.16	3.12	3.08	3.05	3.03	3.00	2.92	2.87	2.83	2.79	2.76
18	8.29	6.01	5.09	4.58	4.25	4.01	3.84	3.71	3.60	3.51	3.43	3.37	3.32	3.27	3.23	3.19	3.16	3.13	3.10	3.08	3.05	3.00	2.97	2.94	2.92	2.84	2.78	2.75	2.70	2.68
19	8.18	5.93	5.01	4.50	4.17	3.94	3.77	3.63	3.52	3.43	3.36	3.30	3.24	3.19	3.15	3.12	3.08	3.05	3.03	3.00	2.96	2.92	2.89	2.87	2.84	2.76	2.71	2.67	2.63	2.60
20	8.10	5.85	4.94	4.43	4.10	3.87	3.70	3.56	3.46	3.37	3.29	3.23	3.18	3.13	3.09	3.05	3.02	2.99	2.96	2.94	2.90	2.86	2.83	2.80	2.78	2.69	2.64	2.61	2.56	2.54
21	8.02	5.78	4.87	4.37	4.04	3.81	3.64	3.51	3.40	3.31	3.24	3.17	3.12	3.07	3.03	2.99	2.96	2.93	2.90	2.88	2.84	2.80	2.77	2.74	2.72	2.64	2.58	2.55	2.50	2.48
22	7.95	5.72	4.82	4.31	3.99	3.76	3.59	3.45	3.35	3.26	3.18	3.12	3.07	3.02	2.98	2.94	2.91	2.88	2.85	2.83	2.78	2.75	2.72	2.69	2.67	2.58	2.53	2.50	2.45	2.42
23	7.88	5.66	4.76	4.26	3.94	3.71	3.54	3.41	3.30	3.21	3.14	3.07	3.02	2.97	2.93	2.89	2.86	2.83	2.80	2.78	2.74	2.70	2.67	2.64	2.62	2.54	2.48	2.45	2.40	2.37
24	7.82	5.61	4.72	4.22	3.90	3.67	3.50	3.36	3.26	3.17	3.09	3.03	2.98	2.93	2.89	2.85	2.82	2.79	2.76	2.74	2.70	2.66	2.63	2.60	2.58	2.49	2.44	2.40	2.36	2.33
25	7.77	5.57	4.68	4.18	3.86	3.63	3.46	3.32	3.22	3.13	3.06	2.99	2.94	2.89	2.85	2.81	2.78	2.75	2.72	2.69	2.66	2.62	2.59	2.56	2.54	2.45	2.40	2.36	2.32	2.29
26	7.72	5.53	4.64	4.14	3.82	3.59	3.42	3.29	3.18	3.09	3.02	2.96	2.90	2.86	2.81	2.78	2.74	2.72	2.69	2.66	2.62	2.58	2.55	2.53	2.50	2.41	2.36	2.33	2.28	2.25
27	7.68	5.49	4.60	4.11	3.78	3.56	3.39	3.26	3.15	3.06	2.99	2.93	2.87	2.82	2.78	2.75	2.71	2.68	2.66	2.63	2.59	2.55	2.52	2.49	2.47	2.38	2.33	2.29	2.25	2.22
28	7.64	5.45	4.57	4.07	3.75	3.53	3.36	3.23	3.12	3.03	2.96	2.90	2.84	2.79	2.75	2.72	2.68	2.65	2.63	2.60	2.56	2.52	2.49	2.46	2.44	2.35	2.30	2.26	2.22	2.19
29	7.60	5.42	4.54	4.04	3.73	3.50	3.33	3.20	3.09	3.00	2.93	2.87	2.81	2.77	2.73	2.69	2.66	2.63	2.60	2.57	2.53	2.50	2.47	2.44	2.41	2.33	2.27	2.23	2.19	2.16
30	7.56	5.39	4.51	4.02	3.70	3.47	3.30	3.17	3.07	2.98	2.91	2.84	2.79	2.74	2.70	2.66	2.63	2.60	2.57	2.55	2.51	2.47	2.44	2.41	2.39	2.30	2.25	2.21	2.16	2.13
32	7.50	5.34	4.46	3.97	3.65	3.43	3.26	3.13	3.02	2.93	2.86	2.80	2.74	2.70	2.66	2.62	2.58	2.55	2.53	2.50	2.46	2.42	2.39	2.36	2.34	2.25	2.20	2.16	2.11	2.08
34	7.44	5.29	4.42	3.93	3.61	3.39	3.22	3.09	2.98	2.89	2.82	2.76	2.70	2.66	2.62	2.58	2.55	2.51	2.49	2.46	2.42	2.38	2.35	2.32	2.30	2.21	2.16	2.12	2.07	2.04
36	7.40	5.25	4.38	3.89	3.57	3.35	3.18	3.05	2.95	2.86	2.79	2.72	2.67	2.62	2.58	2.54	2.51	2.48	2.45	2.43	2.38	2.35	2.32	2.29	2.26	2.17	2.12	2.08	2.03	2.00
38	7.35	5.21	4.34	3.86	3.54	3.32	3.15	3.02	2.92	2.83	2.75	2.69	2.64	2.59	2.55	2.51	2.48	2.45	2.42	2.40	2.35	2.32	2.28	2.26	2.23	2.14	2.09	2.05	2.00	1.97
40	7.31	5.18	4.31	3.83	3.51	3.29	3.12	2.99	2.89	2.80	2.73	2.66	2.61	2.56	2.52	2.48	2.45	2.42	2.39	2.37	2.33	2.29	2.26	2.23	2.20	2.11	2.06	2.02	1.97	1.94
42	7.28	5.15	4.29	3.80	3.49	3.27	3.10	2.97	2.86	2.78	2.70	2.64	2.59	2.54	2.50	2.46	2.43	2.40	2.37	2.34	2.30	2.26	2.23	2.20	2.18	2.09	2.03	1.99	1.94	1.91
44	7.25	5.12	4.26	3.78	3.47	3.24	3.08	2.95	2.84	2.75	2.68	2.62	2.56	2.52	2.47	2.44	2.40	2.37	2.35	2.32	2.28	2.24	2.21	2.18	2.15	2.06	2.01	1.97	1.92	1.89
46	7.22	5.10	4.24	3.76	3.44	3.22	3.06	2.93	2.82	2.73	2.66	2.60	2.54	2.50	2.45	2.42	2.38	2.35	2.33	2.30	2.26	2.22	2.19	2.16	2.13	2.04	1.99	1.95	1.90	1.86
48	7.19	5.08	4.22	3.74	3.43	3.20	3.04	2.91	2.80	2.72	2.64	2.58	2.53	2.48	2.44	2.40	2.37	2.33	2.31	2.28	2.24	2.20	2.17	2.14	2.12	2.02	1.97	1.93	1.88	1.84
50	7.17	5.06	4.20	3.72	3.41	3.19	3.02	2.89	2.79	2.70	2.63	2.56	2.51	2.46	2.42	2.38	2.35	2.32	2.29	2.27	2.22	2.18	2.15	2.12	2.10	2.01	1.95	1.91	1.86	1.82
55	7.12	5.01	4.16	3.68	3.37	3.15	2.98	2.85	2.75	2.66	2.59	2.53	2.47	2.42	2.38	2.34	2.31	2.28	2.25	2.23	2.18	2.14	2.11	2.08	2.06	1.97	1.91	1.87	1.81	1.78
60	7.08	4.98	4.13	3.65	3.34	3.12	2.95	2.82	2.72	2.63	2.56	2.50	2.44	2.39	2.35	2.31	2.28	2.25	2.22	2.20	2.15	2.12	2.08	2.05	2.03	1.94	1.88	1.84	1.78	1.75
65	7.04	4.95	4.10	3.62	3.31	3.09	2.93	2.80	2.69	2.61	2.53	2.47	2.42	2.37	2.33	2.29	2.26	2.23	2.20	2.18	2.13	2.09	2.06	2.03	2.01	1.92	1.86	1.82	1.76	1.73
70	7.01	4.92	4.08	3.60	3.29	3.07	2.91	2.78	2.67	2.59	2.51	2.45	2.40	2.35	2.31	2.27	2.23	2.20	2.17	2.15	2.11	2.07	2.03	2.01	1.98	1.89	1.83	1.78	1.73	1.70
80	6.96	4.88	4.04	3.56	3.26	3.04	2.87	2.74	2.64	2.55	2.48	2.42	2.36	2.31	2.27	2.23	2.20	2.17	2.14	2.12	2.07	2.03	2.00	1.97	1.94	1.85	1.79	1.75	1.69	1.66
90	6.93	4.85	4.01	3.54	3.23	3.01	2.84	2.72	2.61	2.52	2.45	2.39	2.33	2.29	2.24	2.21	2.17	2.14	2.11	2.09	2.04	2.00	1.97	1.94	1.92	1.82	1.76	1.72	1.66	1.62
100	6.90	4.82	3.98	3.51	3.21	2.99	2.82	2.69	2.59	2.50	2.43	2.37	2.31	2.26	2.22	2.19	2.15	2.12	2.09	2.07	2.02	1.98	1.94	1.92	1.89	1.80	1.73	1.69	1.63	1.60
125	6.84	4.78	3.94	3.47	3.17	2.95	2.79	2.66	2.55	2.47	2.39	2.33	2.28	2.23	2.19	2.15	2.11	2.08	2.05	2.03	1.98	1.94	1.91	1.88	1.85	1.76	1.69	1.65	1.59	1.55
150	6.81	4.75	3.92	3.45	3.14	2.92	2.76	2.63	2.53	2.44	2.37	2.31	2.25	2.20	2.16	2.12	2.09	2.06	2.03	2.00	1.96	1.92	1.88	1.85	1.83	1.73	1.66	1.62	1.56	1.52
200	6.76	4.71	3.88	3.41	3.11	2.89	2.73	2.60	2.50	2.41	2.34	2.27	2.22	2.17	2.13	2.09	2.06	2.02	2.00	1.97	1.93	1.89	1.85	1.82	1.79	1.69	1.63	1.58	1.52	1.48
300	6.72	4.68	3.85	3.38	3.08	2.86	2.70	2.57	2.47	2.38	2.31	2.24	2.19	2.14	2.10	2.06	2.03	1.99	1.97	1.94	1.89	1.85	1.82	1.79	1.76	1.66	1.59	1.55	1.48	1.44
500	6.69	4.65	3.82	3.36	3.05	2.84	2.68	2.55	2.44	2.36	2.28	2.22	2.17	2.12	2.07	2.04	2.00	1.97	1.94	1.92	1.87	1.83	1.79	1.76	1.74	1.63	1.56	1.52	1.45	1.41
1000	6.66	4.63	3.80	3.34	3.04	2.82	2.66	2.53	2.43	2.34	2.27	2.20	2.15	2.10	2.06	2.02	1.98	1.95	1.92	1.90	1.85	1.81	1.77	1.74	1.72	1.61	1.54	1.50	1.43	1.38

1) Values from VAN DER WAERDEN, B. L., *Mathematische Statistik*, Springer, Berlin, 1957, page 342. Reprinted by kind permission of the author and publishers.

Appendix C:
Listings of Software

C.1 INDEX TO SOFTWARE ROUTINES:

Subroutine Name	Source Filename	Source Listing Page Number
ADC_READ	DT2801_4	247
AREA	MATH	232
AUTOCORRSCALE	PLOTGRAF	242
AUTORESPSCALE	PLOTGRAF	239
CHGCORRSCALE	PLOTGRAF	243
CHGFREQ	MATH	235
CHGPARA	MODEL	315
CHGSCALE	PLOTGRAF	239
CHGSTRING	FILEHAND	215
CHGTITLE	FILEHAND	216
CHKDIRECTORY	FILEHAND	217
CHKFILENAME	FILEHAND	217
CHKOVERWRITE	FILEHAND	218
CHKPULSETIMING	DT2801_4	259
CHKRANDOMTIMING	DT2801_4	261
CHKSTEPTIMING	DT2801_4	257
COPYPARA	MODEL	314
COPYRESPDATA	FILEHAND	220
CORRGRAPH	PLOTGRAF	241
CUBICAREA	MATH	232
DAC_WRITE	DT2801_4	248
DETMASSFLOW	DETMF	249
DETRESP	MODEL	312
DISREGARD_UP_TO_STEP	MATH	234
DISPLAYINIT	PLOTGRAF	243
DISPLAYNO	PLOTGRAF	244
FILENAME	FILEHAND	216
FILTER	MATH	233
FITTIME00	MODEL	308
FITTIME11	MODEL	309
FITTIME21	MODEL	309
FITTIME22	MODEL	310

Subroutine Name	Source Filename	Source Listing Page Number
FITZ1	MODEL	310
FITZ2	MODEL	311
GAUSS_N	MATH	228
GAUSS_1	MATH	229
GRAFCOMP	PLOTGRAF	240
GRAPHINIT	PLOTGRAF	236
GRAPHSETUP	PLOTGRAF	237
HIGHLIGHT	TYPES	211
INITDATA	DLOGD	266
INITMENU0	MFCMD	282
INITMENU1	DCONVD	329
INITMENU1	DETFRICD	324
INITMENU1	DETMFD	254
INITMENU1	DLOGD	267
INITMENU1	FITD	332
INITMENU1	GRAFD	271
INITMENU1	MFCAD	292
INITMENU1	MFCMD	282
INITMENU1	MFD	299
INITMENU1	MODELD	315
INITMENU1	PLOTGRAD	245
INITMENU11	MFCMD	283
INITMENU12	MFCMD	283
INITMENU2	DETFRICD	324
INITMENU2	DETMFD	255
INITMENU2	DLOGD	267
INITMENU2	GRAFD	272
INITMENU2	MFCAD	293
INITMENU2	MFCMD	284
INITMENU2	MFD	300
INITMENU2	MODELD	316
INITMENU2	PLOTGRAD	246
INITMENU3	DETFRICD	325
INITMENU3	MFCMD	284
INITMENU4	MFCMD	285
INITMENU41	MFCMD	285
INITMODEL	MODEL	311
INITPARA	MODEL	313
INITPARA00	MODELD	316
INITPARA11	MODELD	317
INITPARA21	MODELD	317
INITPARA22	MODELD	318
INITVAR	MFCMD	281
INITVAR	DETFRICD	323
INITVAR1	MFCAD	291
INITVAR1	MFD	298
INITVAR2	MFCAD	292
INITVAR2	MFD	299

Subroutine Name	Source Filename	Source Listing Page Number
LABELS	PLOTGRAF	240
LOADCONST	FILEHAND	223
LOADCONSTM	FILEHAND	224
LOADCORRDATA	FILEHAND	220
LOADLOOPDATA	FILEHAND	224
LOADMFDATA	FILEHAND	222
LOADRESPDATA	FILEHAND	218
LSE	ESTIMATE	303
LSE2	ESTIMATE	303
MASSBIN	DETMF	250
MASSFLOWBELT	DETMF	250
MASSFLOWNUC	DETMF	250
MATRIXMULT	MATH	226
MEANCALC	MATH	230
MFCALI	DETMF	252
MFSTATS	DETMF	251
NELM	ESTIMATE	304
NORM	MATH	234
ONLINE_MEANSDEV	MATH	230
OFFSET	MATH	233
PLOTDATALIN	PLOTGRAF	238
PLOTDATAPIX	PLOTGRAF	238
POSITION	FILEHAND	214
PULSE	DATALOG	258
PULSERESPONSE	DATALOG	258
QUADAREA	MATH	232
RANDOMRESPONSE	DATALOG	260
READINT	TYPES	212
READREAL	TYPES	212
REGRESS	MATH	231
RNDOM	DATALOG	260
SAVECONST	FILEHAND	223
SAVECORRDATA	FILEHAND	221
SAVELOOPDATA	FILEHAND	225
SAVEMFDATA	FILEHAND	222
SAVERESPDATA	FILEHAND	219
SELECTPARA	MODEL	314
SEMI_MEANSDEV	MATH	230
SET_UP	DT2801_4	249
SET_UP_U1	ESTIMATE	301
SET_UP_U2C	ESTIMATE	302
SET_UP_U2R	ESTIMATE	302
SET_UP_Y	ESTIMATE	301

Subroutine Name	Source Filename	Source Listing Page Number
STEP	DATALOG	256
STEPRESPONSE	DATALOG	256
SUMERRSQR	MODEL	312
TIMEDIFF	TYPES	213
TPO	MATH	233
UtU	MATH	227
UtY	MATH	227
WRITEMENU	TYPES	212

C.2 SUMMARY OF SOFTWARE ROUTINES:

C.2.1 General and Basic Menu Routines:

Subroutine Name	Source Filename	Description
Highlight	Types	Highlights a string of text on a text screen
Writemenu	Types	Writes a text menu on a text screen
Readint	Types	Reads an integer number from the keyboard
Readreal	Types	Reads a real number from the keyboard
Timediff	Types	Calculates the time-difference between two samples from the Dos-clock

C.2.2 Filehandling Routines:

Subroutine Name	Source Filename	Description
Position	Filehand	Finds the position of the first free space in a string
Chgstring	Filehand	Edit a string from the keyboard
Filename	Filehand	Enter a filename from the keyboard
Chgtitle	Filehand	Enter a title from the keyboard
Chkdirectory	Filehand	Check if directory exists

Subroutine Name	Source Filename	Description
Chkfilename	Filehand	Check if file exists
Chkoverwrite	Filehand	Checks if file may be overwritten
Loadrespdata	Filehand	Load response data from file
Saverespdata	Filehand	Save response data to a file
Copyrespdata	Filehand	Copy response data from one record to a next
Loadcorrdata	Filehand	Load a massflow correlation data file from disc
Savecorrdata	Filehand	Save a massflow correlation data file to disc
Loadmfdata	Filehand	Load a massflow data set from a file on disc
Savemfdata	Filehand	Save a massflow data set to a file on disc
Loadconst	Filehand	Load calibration constants
Saveconst	Filehand	Save calibration constants
Loadconstm	Filehand	Load calibration constants for reference massflow meter
Loadloopdata	Filehand	Load no of wait loops required for data capture routine
Saveloopdata	Filehand	Saves no of wait loops required to disc

C.2.3 Mathematics Routines:

Subroutine Name	Source Filename	Description
Matrixmult	Math	Matrix multiplies two matrices
UtU	Math	Determines product of matrix U and the transpose of U
UtY	Math	Determines product of matrix U and Vector Y
Gauss_N	Math	Gauss Reduction with N solution vectors
Gauss_1	Math	Gauss reduction with 1 solution vector
Online_meansdev	Math	Determines mean and standard deviation in an on-line fashion
Semi_meansdev	Math	Determines mean and standard deviation of a small set of data in an on-line fashion
Meancalc	Math	Determines the mean and standard deviation of a set of data
Regress	Math	Perform a linear regression between two sets of data
Area	Math	Determines the area under a response
Quadarea	Math	Determines the area under the square of a response
Cubicarea	Math	Determines the area under the cubic of a response

Subroutine Name	Source Filename	Description
Tpo	Math	Determines a number raised to the power of n
Filter	Math	Filters a real time data response
Offset	Math	Subtracts the bias from a real time response
Norm	Math	Normalizes a real time response with respect to the step applied
Disregard_up_to_step	Math	Disregards all samples of a real time response no to the step
Chgfreq	Math	Changes the frequency of a real time response

C.2.4 Graphics Routines:

Subroutine Name	Source Filename	Description
Graphinit	Plotgraf	Determines and initializes the Graphics Adapter
Graphsetup	Plotgraf	Sets screen for a real time response
Plotdatalin	Plotgraf	Plots a response in form of lines
Plotdatapix	Plotgraf	Plots a response in form of pixels
Autorespscale	Plotgraf	Automatically scales a real time response
Chgscale	Plotgraf	Change scales of response graph manually
Labels	Plotgraf	Insert legends for each channel plotted
Grafcomp	Plotgraf	Plot two graphs on the same screen to compare them
Corrgraph	Plotgraf	Set up Screen for a correlation graph and plot correlation graph
Autocorrscale	Plotgraf	Automatically scales a correlation graph
Chgcorrscale	Plotgraf	Change scale of correlation graph manually
Displayinit	Plotgraf	Initializes display of Massflow meter
Displayno	Plotgraf	Write number in MFM display

C.2.5 Datalogging Routines:

Subroutine Name	Source Filename	Description
ADC_read	Dt2801_4	Reads a Voltage from a A/D converter
DAC_write	Dt2801_4	Sends a Voltage from a D/A converter
Set_up	Dt2801_4	Sets A/D and D/A card up for Data logging
Step	Datalog	Basic procedure to perform step test
Stepresponse	Datalog	Routine to set up and perform step test
Chksteptiming	Datalog	Calibrate timing cycle of step test
Pulse	Datalog	Basic procedure to perform pulse test
Pulseresponse	Datalog	Routine to set up and perform pulse test
Chkpulsetiming	Datalog	Calibrate timing cycle of pulse test
Rndom	Datalog	Basic procedure to perform random sequence test
Randomresponse	Datalog	Routine to set up and perform a random sequence test
Chkrandomtiming	Datalog	Calibrate timing cycle random sequence test

C.2.6 Specialized Massflow Meter Routines:

Subroutine Name	Source Filename	Description
Detmassflow	Detmf	Determine massflow from kinetic energy response of a pulse test
Massflownuc	Detmf	Calibrate reading from Nuclear Weightometer
Massbin	Detmf	Calibrate reading of mass in hopper bin
Massflowbelt	Detmf	Determine reference massflow from difference of mass in the hopper bin
Mfstats	Detmf	Determine statistical data, eg. averages
Mfcali	Detmf	Specify Calibration factors

C.2.7 Specialized System Identification Routines:

Subroutine Name	Source Filename	Description
Set_up_U1	Estimate	Sets up U matrix for LSE Algorithm (pole positions)
Set_up_Y	Estimate	Sets up Y vector for LSE Algorithm (pole positions)
Set_up_U2r	Estimate	Sets up U matrix for LSE Algorithm (zero positions, real poles)
Set_up_U2c	Estimate	Sets up U matrix for LSE Algorithm (zero positions, complex poles)
LSE	Estimate	Performs L.S.E.
LSE2	Estimate	Performs L.S.E. to determine transfer function of a measured response
Nelm	Estimate	Determines transfer function by using Nelm algorithm
Fittime00	Model	Determines linear response for a given set of parameters
Fittime11	Model	Determines first order response for a given set of parameters
Fittime21	Model	Determines second order response with real poles for a given set of parameters
Fittime22	Model	Determines second order response with complex poles for a given set of parameters

Subroutine Name	Source Filename	Description
Fitz1	Model	Determines first order response in sampled data domain for a given set of parameters
Fitz2	Model	Determines second order response in sampled data domain for a given set of parameters
Initmodel	Model	Specify which type of model to use
Detresp	Model	Determine theoretical response of model
Summerrsqr	Model	Determine sum of errors squared between theoretical and measured response
Chgpara	Model	Change parameters of transfer function manually
Initpara	Model	Initialize parameters of transfer function to default values
Selectpara	Model	Select certain parameter to regress on and fix others
Copypara	Model	Copy parameters for use in Nelm algorithm

C.2.8 Data Routines:

Subroutine Name	Source Filename	Description
Initdata	Dlogd	Initialize variables for Dlog
Initmenu1	Dlogd	Initialize Main menu for Dlog
Initmenu2	Dlogd	Initialize Change Format menu for Dlog
Initmenu1	Grafd	Initialize Main menu for Graf
Initmenu2	Grafd	Initialize Change File to Display menu for Graf
Initvar	Mfcmd	Initialize variables for Mfcm
Initmenu0	Mfcmd	Initialize Main menu for Mfcm
Initmenu1	Mfcmd	Initialize menu for Manual Data Acquisition for Mfcm
Initmenu11	Mfcmd	Initialize Variables Initialization menu for Mfcm
Initmenu12	Mfcmd	Initialize Filename Initialization menu for Mfcm
Initmenu2	Mfcmd	Initialize Results menu for Mfcm
Initmenu3	Mfcmd	Initialize Transfer Files menu for Mfcm
Initmenu4	Mfcmd	Initialize Data Processing menu for Mfcm
Initmenu41	Mfcmd	Initialize Graphical Results menu for Mfcm

Subroutine Name	Source Filename	Description
Initvar1	Mfcad	Initialize variables for Mfca
Initvar2	Mfcad	Initialize data logging variables for Mfca
Initmenu1	Mfcad	Initialize Main menu for Mfca
Initmenu2	Mfcad	Initialize Variables Initialization menu for Mfca
Initvar1	Mfd	Initialize variables for Mf
Initvar2	Mfd	Initialize data logging variables for Mf
Initmenu1	Mfd	Initialize Main menu for Mf
Initmenu2	Mfd	Initialize Variables Initialization menu for Mf
Initvar	Detfricd	Initialize variables for Detfric
Initmenu1	Detfricd	Initialize Main menu for Detfric
Initmenu2	Detfricd	Initialize Variables Initialization menu for Detfric
Initmenu3	Detfricd	Initialize Filenames and Title Initialization menu for Detfric
Initmenu1	Dconvd	Initialize Main menu for Dconv
Initmenu1	Fitd	Initialize Main menu for Fit
Initmenu1	Plotgrad	Initialize Change Scales menu for 2 channels for Plotgraf

Subroutine Name	Source Filename	Description
Initmenu2	Plotgrad	Initialize Change Scales menu for 1 channel for Plotgraf
Initmenu1	Modeld	Initialize Model Specifications menu for Model
Initmenu1	Modeld	Initialize Parameter Specifications menu for Model
Initpara00	Modeld	Initialize parameters for linear model for Model
Initpara11	Modeld	Initialize parameters for first order model for Model
Initpara21	Modeld	Initialize parameters for second order model with real poles for Model
Initpara22	Modeld	Initialize parameters for second order model with complex poles for Model

C.2.9 Main Programs:

Program Name	Source Filename	Description
Dlog	Dlog	Logg a real time response
Graf	Graf	Plot a real time response on screen
Mfcm	Mfcm	Manual massflow data correlation package
Mfca	Mfca	Automatic massflow data correlation package
MF	Mf	Massflow meter prototype
Detfric	Detfric	Determine friction characteristics of system
Dconv	Dconv	Apply data conversion to response data
Fit	Fit	Identify transfer function of a response

C.3 LISTINGS OF SOFTWARE:

C.3.1 Types.pas

```
{*************************************************************************}
{***   UNIT PROCEDURE LIBRARY WHICH DEFINES VARIOUS TYPES AND RECORDS  ***}
{***   FILE: TYPES.PAS                                                 ***}
{*************************************************************************}

UNIT types;

{*************************************************************************}
                      INTERFACE
{*************************************************************************}

   USES crt;

   CONST
      maxsamp = 1000;        {maximum dimension for matrices}
      massflowmax = 500;     {maximum number of massflow samples posssible}
      masscorrmax = 500;     {maximum number of correlation samples possible}
      respdatamax = 501;     {maximum number of samples for a real time response}
      respdatachannel = 6;   {maximum number of channels to be sampled for a real time response}

   TYPE
      { Types used for Filehand }
      labeltype = array [1..respdatachannel] of string[10];
      screenmenu = array [1..24] of string[80];         {contains one menu for the screen}

      response = array [1..respdatachannel,0..respdatamax] of integer;     {data of a real time response}
      responsedata = record
                     nsamples:integer;     {no of samples in this response - max of respdatamax}
                     nochannel:integer;    {no of channels sampled - max of respdatachannel}
                     whenstep:integer;     {sample at which step was applied}
                     step:real;            {size of step in volts}
                     frequency:real;       {sampling frequency in Hertz}
                     labels:labeltype;     {labels for the different channels sampled}
                     title:string[50];     {title of the response}
                     data:response;        {actaul response data}
                  end;


      { Types used in Model }
      parameterdata = record
                       no:integer;                      {no of parameters neede for the model}
                       para:array[1..10] of real;       {actual value of parameter}
                       vari:array[1..10] of integer;    {flag to indicate if parameter is to be
                                                         regressed on or not}
                    end;
      modeldata = record
                  order:integer;    {value indicates order of model: 0-integrating response, 1-1st order,
                                     2-2nd order}
                  typ:integer;      {value indicates if model contains real poles[1] or complex poles[2]}
                  what:string[50];  {title of model}
               end;

      { Types used in Math }
      matrixbig = record
                  rows,cols:integer;    {no of rows and columns in matrix - limited to a 5xmaxsamp matrix}
                  a:array[1..maxsamp,1..5]of real;   {contains actual entries of matrix}
               end;
      matrixsml = record
                  rows,cols:integer;    {no of rows and columns in matrix - limited to a maximum of 10x10}
                  a:array[1..10,1..10]of real;       {contains actual entries of matrix}
               end;
      vectorbig = record
                  elements:integer;     {no of elements in vector - limited by maxsamp}
                  v:array[1..maxsamp]of real;        {contains actual entries of vector}
               end;
      vectorsml = record
                  elements:integer;     {no of elements in vector - limited to maximum of 10}
                  v:array[1..10]of real;     {actual entries of matrix}
               end;

      { Types used in Plotgraf }
      responsescale = record
                       ymax,ymin:integer;   {maximum and minimum of y-scale}
                       tmax,tmin:real;      {maximum and minimum of horizontal t-scale}
                    end;
      corrscale = record
                  ymin,ymax,xmin,xmax:array[1..2]of real;   {maximum and minimum of y- and x-scale}
               end;
```

```
    { Types used in Detmass }
    correlationdata = record
                        data:array[1..4,1..masscorrmax] of real;
                                                {actual data arrays for which the correlation
                                                 is to be checked}
                        nsamples,nzero:integer;  {no of samples in total and no of samples taken
                                                  for initializing the system}
                        title:string[50];        {title of correlation data}
                      end;
    reqressiondata = record
                       slope,yconst,error,r,ymean,xmean,yerror,C1,C2:array[1..2] of real;
                       {information about best fit for a linear function: slope, y-intercept,
                        error of estimate at mean, correlation coefficient, mean of y and x values,
                        standard error of y value, constants C1 and C20 to determine the confidence band}
                     end;

    constant = record
               speed,power,dh,offset,slope,error,Fa,Fb:real;
                    {calibration constants of m/s/count for speed, W/count for power,
                     length of conveyor belt,
                     zero offset and slope of calibration graph, error associated with massflow reading,
                     friction constants proportional to speed and square of speed}
            end;
    massflowconst = record
                      vincl,dh,di,dch1,dch2,mass,nuc:real;
                       {calibration constants for speed of inclined conveyor belt, length of horizontal
                        and inclined conveyor belt, equivalent length of the chutes, kg/count for mass
                        in the hopper bin, t/hr/count for the nuclear weightometer}
                    end;
    time = record
           ssup,ssdn,stepdn,stepup,ssend:integer;
                     {time instances when steady state for input being high starts (- ssup),
                      steady state for input being low (- ssdn), instance of step down (- stepdn),
                      and step up (- stepup), and instance when last sample is (-ssend).
                      Sequence: ssup-stepdn-ssdn-stepup-ssend }
      end;

    dostime = record
              hour,min,sec,sec100:word;  {record to contain a sample from the dos-clock}
          end;

    { Types used in Massflow }
    massflowdata = record
                     data:array[1..massflowmax]of real;        {array of actual massflow samples}
                     time:array[1..massflowmax]of dostime;     {array of time samples indicating when
                                                                sample was taken}
                     nsamples:integer;                         {no of samples}
                     error:real;                               {error associated with reading}
                     title:string[50];                         {title of massflow test}
                   end;


   PROCEDURE Highlight(mode:integer);
   PROCEDURE writemenu(screen:screenmenu);
   PROCEDURE readint(x,y,nopos:integer;VAR n:integer);
   PROCEDURE readreal(x,y,nopos:integer;VAR n:real);
   PROCEDURE timediff(t1,t2:dostime;VAR dt:longint);

{*************************************************************************}
                  IMPLEMENTATION
{*************************************************************************}

{*************************************************************************}
{N**   MODULE NAME       :  HIGHLIGHT                                  ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE         :  highlights a string of text written to the ***}
{***                         screen in a flashing mode (mode=1) or a    ***}
{***                         normal mode (mode=0). If the output is al- ***}
{***                         ready in the highlighted mode the output is***}
{***                         set to normal                              ***}
{S**   CALL SEQUENCE     :  HIGHLIGHT(mode)                            ***}
{I**   INPUT PARAMETERS  :  mode=1 => flashing output; mode=0 => normal***}
{O**   OUTPUT PARAMETERS :  none                                       ***}
{G**   GLOBAL VARIABLES  :  none                                       ***}
{M**   MODULES CALLED    :  none                                       ***}
{E**   ERROR CONDITIONS  :  none                                       ***}
{C**   COMMENTS          :  none                                       ***}
{*************************************************************************}

PROCEDURE Highlight(mode:integer);

BEGIN
   if mode=1 then mode:=128 else mode:=0;
   textbackground(7);
   textcolor(0+mode);
END;
```

```
{*************************************************************************}
{N**   MODULE NAME        :  WRITEMENU                                    ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          :  writes the menu contained in an array of     ***}
{***                         strings to the screen                        ***}
{S**   CALL SEQUENCE      :  WRITEMENU(arrayofstrings)                    ***}
{I**   INPUT PARAMETERS   :  array of strings                             ***}
{O**   OUTPUT PARAMETERS  :  none                                         ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  none                                         ***}
{E**   ERROR CONDITIONS   :  none                                         ***}
{C**   COMMENTS           :  none                                         ***}
{*************************************************************************}

PROCEDURE writemenu(screen:screenmenu);

  VAR
     i:integer;

   BEGIN
      for i:=1 to 24 do writeln(screen[i]);
   END;


{*************************************************************************}
{N**   MODULE NAME        :  READINT                                      ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          :  Reads a string supposed to be an integer     ***}
{***                         from the keyboard and converts the string    ***}
{***                         to an integer                                ***}
{S**   CALL SEQUENCE      :  READINT(x,y,nopos,integernumber)             ***}
{I**   INPUT PARAMETERS   :  position on screen (x,y), no of positions    ***}
{***                         allocated for the number                     ***}
{O**   OUTPUT PARAMETERS  :  actual integer                               ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  Crt                                          ***}
{E**   ERROR CONDITIONS   :  if the string entered is not an integer      ***}
{C**   COMMENTS           :  if <ENTER> is keyed in, the old number is    ***}
{***                         accepted                                     ***}
{*************************************************************************}

PROCEDURE readint(x,y,nopos:integer;VAR n:integer);

   VAR
      i,code:integer;
      readstring:string[40];

   BEGIN
      code:=0;
      repeat
         readstring:='';
         gotoXY(x,y);for i:=1 to nopos do write(' ');
         gotoXY(x,y);readln(readstring);
{$R-}
         if readstring<>'' then val(readstring,n,code);
{$R+}
         if code<>0 then
            begin
               Sound(1000);delay(50);nosound;
            end;
      until (code=0) or (readstring='');
   END;


{*************************************************************************}
{N**   MODULE NAME        :  READREAL                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          :  Reads a string supposed to be a real         ***}
{***                         from the keyboard and converts the string    ***}
{***                         to a real number                             ***}
{S**   CALL SEQUENCE      :  READREAL(x,y,nopos,realnumber)               ***}
{I**   INPUT PARAMETERS   :  position on screen (x,y), no of positions    ***}
{***                         allocated for number                         ***}
{O**   OUTPUT PARAMETERS  :  actual real                                  ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  Crt                                          ***}
{E**   ERROR CONDITIONS   :  If string entered is not a real             ***}
{C**   COMMENTS           :  If <ENTER> is keyed in the old Number is     ***}
{***                         accepted                                     ***}
{*************************************************************************}

PROCEDURE readreal(x,y,nopos:integer;VAR n:real);

   VAR
      i,code:integer;
      readstring:string[40];
```

```
   BEGIN
      code:=0;
      repeat
         readstring:='';
         gotoXY(x,y);for i:=1 to nopos do write(' ');
         gotoXY(x,y);readln(readstring);
{$R-}
         if readstring<>'' then val(readstring,n,code);
{$R+}
         if code<>0 then
            begin
               Sound(1000);delay(50);nosound;
            end;
      until (code=0) or (readstring='');
   END;


{*****************************************************************************}
{N**   MODULE NAME        :  TIMEDIFF                                         ***}
{***   ------------------------------------------------------------------    ***}
{A**   PROCEDURE          :  determines the time difference in msec's    ***}
{***                          between two time samples from the dos clock***}
{S**   CALL SEQUENCE      :  TIMEDIFF(first time sample, second time     ***}
{***                          sample, time difference)                    ***}
{I**   INPUT PARAMETERS   :  two time samples as taken from the DOS      ***}
{***                          clock (Type: dostime; see TYPES)            ***}
{O**   OUTPUT PARAMETERS  :  time difference in msec's                    ***}
{G**   GLOBAL VARIABLES   :  None                                         ***}
{M**   MODULES CALLED     :  None                                         ***}
{E**   ERROR CONDITIONS   :  None                                         ***}
{C**   COMMENTS           :                                               ***}
{*****************************************************************************}

PROCEDURE timediff(t1,t2:dostime;VAR dt:longint);

   BEGIN
      if t1.min>t2.min then begin t2.min:=t2.min+60; t2.hour:=t2.hour-1; if t2.hour=-1 then t2.hour:=23;end;
      if t1.sec>t2.sec then begin t2.sec:=t2.sec+60; t2.min:=t2.min-1; end;
      if t1.sec100>t2.sec100 then begin t2.sec100:=t2.sec100+100; t2.sec:=t2.sec-1; end;

      t2.hour:=t2.hour-t1.hour;
      t2.min:=t2.min-t1.min;
      t2.sec:=t2.sec-t1.sec;
      t2.sec100:=t2.sec100-t1.sec100;

      dt:=t2.hour*60;
      dt:=(dt+t2.min)*60;
      dt:=(dt+t2.sec)*100;
      dt:=(dt+t2.sec100)*10;
   END;


{*****************************************************************************}
{H**   REVISION HISTORY:                                                     ***}
{***   VERSION      BY        DATE       COMMENTS                           ***}
{***     1.0       KCAD     26-02-90                                        ***}
{***                                                                        ***}
{***   UNITEND  TYPES.PAS                                                   ***}
{*****************************************************************************}
END.
```

C.3.2 Filehand.pas

```
{*************************************************************************}
{***   UNIT PROCEDURE LIBRARY TO HANDLE DATA FILES                     ***}
{***   FILE:FILEHAND.PAS                                               ***}
{*************************************************************************}

UNIT filehand;

{*************************************************************************}
                        INTERFACE
{*************************************************************************}

   USES Crt,Types;

   TYPE
      str50=string[50];

   FUNCTION position(name:str50):integer;
   FUNCTION chgstring(name,clear:str50;pos,x,y:integer):str50;
   PROCEDURE filename(VAR name:str50;id:string);
   FUNCTION chgtitle(what:str50;strin:str50):str50;
   PROCEDURE chkdirectory(VAR name:str50;VAR flag:boolean);
   PROCEDURE chkfilename(VAR name:str50;VAR flag:boolean);
   PROCEDURE chkoverwrite(VAR name:str50;VAR flag:boolean);
   PROCEDURE loadresponsedata(VAR name:str50; VAR loaddata:responsedata);
   PROCEDURE saveresponsedata(VAR name:str50;storedata:responsedata);
   PROCEDURE copyresponsedata(data1:responsedata;VAR data2:responsedata);
   PROCEDURE loadcorrdata(VAR name:str50;VAR corrdata:correlationdata;VAR regdata:regressiondata);
   PROCEDURE savecorrdata(VAR name:str50;corrdata:correlationdata;regdata:regressiondata);
   PROCEDURE loadmfdata(VAR name:str50;VAR data:massflowdata);
   PROCEDURE savemfdata(VAR name:str50;data:massflowdata);
   PROCEDURE Loadconst(VAR Csp:constant);
   PROCEDURE Saveconst(VAR Csp:constant);
   PROCEDURE Loadconstm(VAR Cm:massflowconst);
   PROCEDURE loadloopdata(VAR noloop:longint);
   PROCEDURE saveloopdata(noloop:longint);

{*************************************************************************}
                        IMPLEMENTATION
{*************************************************************************}

{*************************************************************************}
{N**   MODULE NAME       : POSITION                                     ***}
{***   ------------------------------------------------------------    ***}
{A**   FUNCTION          : finds the position of the first free space   ***}
{***                        in a character string                       ***}
{S**   CALL SEQUENCE     : POSITION(string)                             ***}
{I**   INPUT PARAMETERS  : string to be investigated                    ***}
{O**   OUTPUT PARAMETERS : position of first space                       ***}
{G**   GLOBAL VARIABLES  : none                                         ***}
{M**   MODULES CALLED    : none                                         ***}
{E**   ERROR CONDITIONS  : none                                         ***}
{C**   COMMENTS          : none                                         ***}
{*************************************************************************}


FUNCTION position(name:str50):integer;

   VAR
      pos:integer;
      endid:string[1];

   BEGIN
      pos:=50;
      repeat
         pos:=pos-1;
         endid:=copy(name,pos,1);
      until (endid<>'') or (pos=1);

      if pos<>1 then pos:=pos+1;

      position:=pos;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : CHGSTRING                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   FUNCTION           : edits a string by taking the appropiate       ***}
{***                        action as defined by the user from the key-   ***}
{***                        board                                         ***}
{S**   CALL SEQUENCE       : CHGSTRING(string,clearing string,position,    ***}
{***                        x position,y position)                        ***}
{I**   INPUT PARAMETERS   : name of string, clearing string, current      ***}
{***                        position of cursor in string and x,y posi-    ***}
{***                        tion of most left hand entry of the string    ***}
{***                        on the screen                                 ***}
{O**   OUTPUT PARAMETERS  : updated string                                ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Crt                                           ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : clearing string is a string consisting of     ***}
{***                        spaces to clear the allocated space on the    ***}
{***                        screen. The length of the clearing string is***}
{***                        equal to that of the actual string            ***}
{*************************************************************************}

FUNCTION chgstring(name,clear:str50;pos,x,y:integer):str50;


  VAR
    posc:integer;
    key:char;

  BEGIN
    HIGHLIGHT(0);
    posc:=POSITION(clear);
    gotoXY(x,y);write(clear);
    gotoXY(x,y);write(name);
    gotoXY(x+pos-1,y);
    key:=readkey;

    while key<>#13 do
      begin
        if key=#0 then
          case readkey of
            #75:pos:=pos-1;          {left arrow pressed}
            #77:pos:=pos+1;          {right arrow pressed}
            #83:delete(name,pos,1); {delete pressed}
          end
        else if key=#8 then
          begin
            pos:=pos-1;              {backspace pressed}
            delete(name,pos,1);
          end

        else
          begin
            if pos<posc then         {any other key pressed}
              begin
                insert(key,name,pos);
                pos:=pos+1;
                delete(name,posc,1);
              end
            else
              begin
                sound(1000);delay(250);nosound;
              end;
          end;

        gotoXY(x,y);write(clear);
        gotoXY(x,y);write(name);
        gotoXY(x+pos-1,y);key:=readkey;
      end;
    chgstring:=name;
    normvideo;
  END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : FILENAME                                      ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : creates a filename which can be entered and ***}
{***                        edited one by one                            ***}
{S**   CALL SEQUENCE      : FILENAME(string containing filename,         ***}
{***                        extension)                                   ***}
{I**   INPUT PARAMETERS   : string containing default subdirectory and  ***}
{***                        name of file, extension of filename          ***}
{O**   OUTPUT PARAMETERS  : edited filename                               ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED     : Crt                                          ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           : if extension is entered as '', the routine  ***}
{***                        will detect if the filename already contains***}
{***                        a extension or it will assume no extension   ***}
{*************************************************************************}

PROCEDURE filename(VAR name:str50;id:string);

  VAR
    okay:char;
    pos:integer;
    clear:string[50];
    test:string[1];

  BEGIN
    okay:='y';
    clear:='                                                  ';

    repeat
      pos:=POSITION(name);
      test:=copy(name,pos-4,1);
      if test='.' then
        begin
          pos:=pos-4;
          if id<>'' then delete(name,pos,4);
        end;
      insert(id,name,pos);

      clrscr;gotoXY(20,10);writeln('filename: ');
      name:=CHGSTRING(name,clear,pos,30,10);
      clrscr;gotoXY(20,10);writeln('filename: ',name);
      gotoXY(37,12);write('  okay? [y] ');okay:=readkey;
      if okay=#13 then okay:='y';writeln(okay);
    until okay='y';
  END;


{*************************************************************************}
{N**   MODULE NAME        : CHGTITLE                                      ***}
{***   ------------------------------------------------------------      ***}
{A**   FUNCTION           : edits a previously entered title string one ***}
{***                        by one                                       ***}
{S**   CALL SEQUENCE      : CHGTITLE(explainstring,titlestring)          ***}
{I**   INPUT PARAMETERS   : Explanation of the string,title string       ***}
{O**   OUTPUT PARAMETERS  : edited title string                          ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED     : Crt                                          ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           : example for explain string: 'Title: '        ***}
{*************************************************************************}

FUNCTION chgtitle(what:str50;strin:str50):str50;

  VAR
    okay:char;
    pos1,pos2:integer;
    clear:string[50];

  BEGIN
    okay:='y';
    clear:='                                                  ';

    repeat
      pos1:=POSITION(strin);
      pos2:=POSITION(what);

      clrscr;gotoXY(5,10);writeln(what,' ');
      strin:=CHGSTRING(strin,clear,pos1,5+pos2+2,10);
      clrscr;gotoXY(5,10);writeln(what,':  ',strin);
      gotoxy(22,12);write('  okay? [y] ');okay:=readkey;
      if okay=#13 then okay:='y';writeln(okay);
    until okay='y';

    chgtitle:=strin;
  END;
```

```
{*************************************************************************}
{N**   MODULE NAME        :  CHKDIRECTORY                                 ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          :  checks if the directory defined for a new    ***}
{***                          file or a file to be read from is valid,    ***}
{***                          i.e. that the directory defined exists      ***}
{S**   CALL SEQUENCE      :  CHKDIRECTORY(filename,flag)                  ***}
{I**   INPUT PARAMETERS   :  filename                                     ***}
{O**   OUTPUT PARAMETERS  :  flag indicating whether to continue         ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  Crt,Types                                    ***}
{E**   ERROR CONDITIONS   :  if directory does not exist                  ***}
{C**   COMMENTS           :  This check is done by writing a dummy file   ***}
{***                          to the defined directory. Should this       ***}
{***                          action be successful the dummy file is      ***}
{***                          erased again                                ***}
{*************************************************************************}

PROCEDURE chkdirectory(VAR name:str50;VAR flag:boolean);

  VAR
    opt:char;
    disc:text;
    pos:integer;
    tempname:str50;

  BEGIN
    repeat
      flag:=false;

      pos:=POSITION(name);
      tempname:=name;
      delete(tempname,pos-4,4);

      assign(disc,tempname);
{$I-}
      rewrite(disc);
{$I+}

      if IOResult=0 then
        begin
          flag:=true;
          close(disc);
          erase(disc);
        end
      else
        begin
          sound(1000);delay(250);nosound;
          HIGHLIGHT(1);gotoXY(25,20);writeln(' Directory does not exist ');normvideo;
          gotoXY(15,22);writeln('Press <ENTER> to re-enter directory or <ESC> to Escape');
          opt:=readkey;
          if opt=#13 then FILENAME(name,'');
        end;
    until (opt=#27) or (flag=true);
  END;

{*************************************************************************}
{N**   MODULE NAME        :  CHKFILENAME                                  ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          :  checks if the name defined for a file to be***}
{***                          read is valid, i.e. that the file exists    ***}
{S**   CALL SEQUENCE      :  CHKFILENAME(filename,flag)                   ***}
{I**   INPUT PARAMETERS   :  filename                                     ***}
{O**   OUTPUT PARAMETERS  :  flag indicating whether to continue         ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  Crt,Types                                    ***}
{E**   ERROR CONDITIONS   :  if filename does not exiat                   ***}
{C**   COMMENTS           :  none                                         ***}
{*************************************************************************}

PROCEDURE chkfilename(VAR name:str50;VAR flag:boolean);

  VAR
    opt:char;
    disc:text;
```

```
   BEGIN
      repeat
         flag:=false;
         assign(disc,name);
{$I-}
         reset(disc);
{$I+}

         if IOResult=0 then
            begin
               flag:=true;
               close(disc);
            end
         else
            begin
               sound(1000);delay(250);nosound;
               HIGHLIGHT(1);gotoXY(27,20);writeln(' File does not exist ');normvideo;
               gotoXY(12,22);writeln('Press <ENTER> to enter new filename or <ESC> to Escape');
               opt:=readkey;
               if opt=#13 then FILENAME(name,'');
            end;
      until (opt=#27) or (flag=true);
   END;


{*************************************************************************}
{N**   MODULE NAME       :  CHKOVERWRITE                                  ***}
{***   ------------------------------------------------------------       ***}
{A**   PROCEDURE         :  checks if the name defined for a new file     ***}
{***                        does already exist and if it does exist       ***}
{***                        whether to overwrite it                       ***}
{S**   CALL SEQUENCE     :  CHKOVERWRITE(filename,flag)                   ***}
{I**   INPUT PARAMETERS  :  filename                                       ***}
{O**   OUTPUT PARAMETERS :  flag indicating whether to overwrite          ***}
{G**   GLOBAL VARIABLES  :  none                                           ***}
{M**   MODULES CALLED    :  Crt,Types                                     ***}
{E**   ERROR CONDITIONS  :  if filename does already exist                 ***}
{C**   COMMENTS          :  none                                          ***}
{*************************************************************************}

PROCEDURE chkoverwrite(VAR name:str50;VAR flag:boolean);

   VAR
      opt:char;
      disc:text;

   BEGIN
      repeat
         flag:=false;
         assign(disc,name);
{$I-}
         reset(disc);
{$I+}

         if IOResult=0 then
            begin
               sound(1000);delay(250);nosound;
               clrscr;HIGHLIGHT(1);gotoXY(25,20);writeln(' File does already exist ');normvideo;
               gotoXY(25,22);write('Press <ENTER> to Overwrite');
               gotoXY(31,23);write('<ESC>   to Escape');
               gotoXY(31,24);write('<R>     to Re-enter filename');
               opt:=readkey;
               if opt=#13 then flag:=true;
               if (opt='r') or (opt='R') then FILENAME(name,'');

               close(disc);
            end
         else flag:=true;
      until (opt=#27) or (flag=true);
   END;


{*************************************************************************}
{N**   MODULE NAME       :  LOADRESPDATA                                  ***}
{***   ------------------------------------------------------------       ***}
{A**   PROCEDURE         :  loads a file of data in the RESPONSEDATA      ***}
{***                        record format                                 ***}
{S**   CALL SEQUENCE     :  LOADRESPDATA(filename,datarecord)             ***}
{I**   INPUT PARAMETERS  :  filename                                       ***}
{O**   OUTPUT PARAMETERS :  data record                                   ***}
{G**   GLOBAL VARIABLES  :  none                                           ***}
{M**   MODULES CALLED    :  Crt,Types                                     ***}
{E**   ERROR CONDITIONS  :  if file does not exist                        ***}
{C**   COMMENTS          :  details about RESPONSEDATA data record in     ***}
{***                        unit TYPES                                    ***}
{*************************************************************************}

PROCEDURE loadresponsedata(VAR name:str50; VAR loaddata:responsedata);

   VAR
      i,j,code:integer;
      disc:text;
      datastring:string[5];
      message:string[80];
      flag:boolean;
```

```
BEGIN
    CHKDIRECTORY(name,flag);
    if flag=true then CHKFILENAME(name,flag);

    if flag=true then
       with loaddata do
          begin
             message:='Please wait - Loading file: ';
             insert(name,message,29);
             clrscr;HIGHLIGHT(0);gotoXY(10,10);writeln(message);normvideo;

             assign(disc,name);
             reset(disc);

             readln(disc,nsamples,i,nochannel,whenstep,step);
             readln(disc,frequency);
             for i:=1 to nochannel do read(disc,labels[i]);
             readln(disc);
             readln(disc,title);
             for i:=0 to (nsamples-1) do
                begin
                   for j:=1 to nochannel do
                      begin
                         read(disc,datastring);
                         val(datastring,data[j,i],code);
                      end;
                   readln(disc);
                end;

             close(disc);
          end;
END;


{*************************************************************************}
{N**   MODULE NAME       :  SAVERESPDATA                                  ***}
{***   ----------------------------------------------------------------   ***}
{A**   PROCEDURE         :  stores a data record in the RESPONSEDATA      ***}
{***                        recordon disc                                  ***}
{S**   CALL SEQUENCE     :  SAVERESPDATA(filename,datarecord)             ***}
{I**   INPUT PARAMETERS  :  filename, data record                        ***}
{O**   OUTPUT PARAMETERS :  none                                         ***}
{G**   GLOBAL VARIABLES  :  none                                         ***}
{M**   MODULES CALLED    :  Crt,Types                                     ***}
{E**   ERROR CONDITIONS  :  if file already exists                       ***}
{C**   COMMENTS          :  details about RESPONSEDATA data record in     ***}
{***                        unit TYPES                                     ***}
{*************************************************************************}

PROCEDURE saveresponsedata(VAR name:str50;storedata:responsedata);

   VAR
      i,j:integer;
      disc:text;
      message:string[80];
      flag:boolean;

   BEGIN
      CHKDIRECTORY(name,flag);
      if flag=true then CHKOVERWRITE(name,flag);

      if flag=true then
         with storedata do
            begin
               message:=' Please wait - Saving file: ';
               insert(name,message,28);
               clrscr;HIGHLIGHT(0);gotoXY(10,10);writeln(message);normvideo;

               assign(disc,name);
               rewrite(disc);

               writeln( disc,nsamples:5,'    ',1,'    ',nochannel,'    ',whenstep,'    ',step:10:5);
               writeln(disc,frequency:10:4);
               for i:= 1 to nochannel do write(disc,labels[i]:10);
               writeln(disc);
               writeln(disc,title);
               for i:=0 to (nsamples-1) do
                  begin
                     for j:=1 to nochannel do write(disc,data[j,i]:5);
                     writeln(disc);
                  end;

               close(disc);
            end;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        :  COPYRESPDATA                                 ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          :  copies one data record in the RESPONSEDATA ***}
{***                          format into another data record            ***}
{S**   CALL SEQUENCE      :  COPYRESPDATA(data1,data2)                    ***}
{I**   INPUT PARAMETERS   :  data1 = data record to be copied FROM       ***}
{O**   OUTPUT PARAMETERS  :  data2 = data record to be copied TO         ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  Types                                        ***}
{E**   ERROR CONDITIONS   :  none                                         ***}
{C**   COMMENTS           :  details about RESPONSEDATA data record in  ***}
{***                          unit TYPES                                 ***}
{*************************************************************************}

PROCEDURE copyresponsedata(data1:responsedata;VAR data2:responsedata);

   VAR
      i,j:integer;

  BEGIN
     data2.nsamples:=data1.nsamples;
     data2.nochannel:=data1.nochannel;
     data2.whenstep:=data1.whenstep;
     data2.step:=data1.step;
     data2.frequency:=data1.frequency;
     for i:=1 to data1.nochannel do
        data2.labels[i]:=data1.labels[i];
     data2.title:=data1.title;
     for i:=0 to (data1.nsamples-1) do
         for j:=1 to data1.nochannel do
              data2.data[j,i]:=data1.data[j,i];
   END;


{*************************************************************************}
{N**   MODULE NAME        :  LOADCORRDATA                                 ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          :  loads a file containing resulting data from***}
{***                          a linear regression and the data which was ***}
{***                          used in the regression process, i.e.the    ***}
{***                          data for which the correlation was checked ***}
{S**   CALL SEQUENCE      :  LOADCORRDATA(filename,correlation data      ***}
{***                          record, regression data record)            ***}
{I**   INPUT PARAMETERS   :  filename                                     ***}
{O**   OUTPUT PARAMETERS  :  record containing the regression data (i.e.***}
{***                          results from the regression process) and a ***}
{***                          record containing the data used in the reg-***}
{***                          ression process (i.e. correlation Data)    ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  Crt,Types                                    ***}
{E**   ERROR CONDITIONS   :  if file does not exist                       ***}
{C**   COMMENTS           :  for details about correlation and           ***}
{***                          regression data record see the unit TYPES  ***}
{*************************************************************************}

PROCEDURE loadcorrdata(VAR name:str50;VAR corrdata:correlationdata;VAR regdata:regressiondata);

   VAR
      i,j,code:integer;
      disc:text;
      datastring:string[7];
      dum:string[3];
      message:string[80];
      flag:boolean;


   BEGIN
      CHKDIRECTORY(name,flag);
      if flag=true then CHKFILENAME(name,flag);

      if flag=true then
         with corrdata,regdata do
            begin
               message:=' Please Wait - Loading File: ';
               insert(name,message,29);
               clrscr;HIGHLIGHT(0);gotoXY(10,10);writeln(message);normvideo;

               assign(disc,name);
               reset(disc);

               readln(disc,title);
               readln(disc,nsamples,nzero);
               readln(disc,slope[1],slope[2]);
               readln(disc,yconst[1],yconst[2]);
               readln(disc,error[1],error[2]);
               readln(disc,r[1],r[2]);
               readln(disc,yerror[1],yerror[2]);
               readln(disc,xmean[1],xmean[2]);
               readln(disc,ymean[1],ymean[2]);
               readln(disc,C1[1],C1[2]);
               readln(disc,C2[1],C2[2]);
               readln(disc);
```

```
            for i:=1 to nsamples do
               begin
                  for j:=1 to 3 do
                     begin
                        read(disc,datastring,dum);
                        val(datastring,data[j,i],code);
                     end;
                  readln(disc);
               end;

            close(disc);
         end;
   END;

{*************************************************************************}
{N**   MODULE NAME       :  SAVECORRDATA                                  ***}
{***   ------------------------------------------------------------------  ***}
{A**   PROCEDURE         :  saves the resulting data from a linear        ***}
{***                        regression and the data which was used in     ***}
{***                        the regression process, i.e. the data for     ***}
{***                        which the correlation was checked to a file   ***}
{S**   CALL SEQUENCE     :  SAVECORRDATA(filename,correlation data         ***}
{***                        record, regression data record)               ***}
{I**   INPUT PARAMETERS  :  filename,                                      ***}
{O**                       record containing the regression data (i.e.   ***}
{***                       results from the regression process) and a    ***}
{***                       record containing the data used in the reg-   ***}
{***                       ression process (i.e. correlation Data)       ***}
{O**   OUTPUT PARAMETERS :  none                                          ***}
{G**   GLOBAL VARIABLES  :  none                                          ***}
{M**   MODULES CALLED    :  Crt,Types                                     ***}
{E**   ERROR CONDITIONS  :  If file already exists                       ***}
{C**   COMMENTS          :  for details about correlation and             ***}
{***                       regression data record see the unit TYPES     ***}
{*************************************************************************}

PROCEDURE savecorrdata(VAR name:str50;corrdata:correlationdata;regdata:regressiondata);

   VAR
      i,j:integer;
      disc:text;
      message:string[80];
      flag:boolean;

   BEGIN
      CHKDIRECTORY(name,flag);
      if flag=true then CHKOVERWRITE(name,flag);

      if flag=true then
         with corrdata,regdata do
            begin

               message:=' Please Wait - Saving File: ';
               insert(name,message,28);
               clrscr;HIGHLIGHT(0);gotoXY(10,10);writeln(message);normvideo;

               assign(disc,name);
               rewrite(disc);

               writeln(disc,title);
               writeln(disc,nsamples,'   ',nzero);
               writeln(disc,slope[1]:12:5,'   ',slope[2]:12:5,'   :Slope');
               writeln(disc,yconst[1]:12:5,'   ',yconst[2]:12:5,'   :Y-intercept');
               writeln(disc,error[1]:12:5,'   ',error[2]:12:5,'   :Error at mean');
               writeln(disc,r[1]:12:5,'   ',r[2]:12:5,'   :Correlation Coefficient');
               writeln(disc,yerror[1]:12:5,'   ',yerror[2]:12:5,'   :Standard Error in Y-Estimate');
               writeln(disc,xmean[1]:12:5,'   ',xmean[2]:12:5,'   :X-mean');
               writeln(disc,ymean[1]:12:5,'   ',ymean[2]:12:5,'   :Y-mean');
               writeln(disc,C1[1]:12:10,'   ',C1[2]:12:10,'   :Constant 1');
               writeln(disc,C2[1]:12:5,'   ',C2[2]:12:5,'   :Constant 2');
               writeln(disc);

               for i:=1 to nsamples do
                  begin
                     for j:=1 to 3 do
                        write(disc,data[j,i]:7:2,'   ');
                     writeln(disc);
                  end;

               close(disc);
            end;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        :  LOADMFDATA                                   ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          :  loads massflow data from file                ***}
{S**   CALL SEQUENCE      :  LOADMFDATA(filename, massflow data record    ***}
{I**   INPUT PARAMETERS   :  filename                                     ***}
{O**   OUTPUT PARAMETERS  :  data record containing massflow data        ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  Crt,Types                                    ***}
{E**   ERROR CONDITIONS   :  if file does not exists                      ***}
{C**   COMMENTS           :  for more information about the massflow      ***}
{***                          data record see TYPES                      ***}
{*************************************************************************}

PROCEDURE loadmfdata(VAR name:str50;VAR data:massflowdata);

  VAR
     i,code:integer;
     disc:text;
     hourstr,minstr,secstr:string[2];
     dumstr:string[1];
     message:string[80];
     flag:boolean;

 BEGIN
     CHKDIRECTORY(name,flag);
     if flag=true then CHKFILENAME(name,flag);

     if flag=true then
        with data do
           begin
              message:='Please Wait - Loading File: ';
              insert(name,message,29);
              clrscr;HIGHLIGHT(0);gotoXY(10,10);writeln(message);normvideo;

              assign(disc,name);
              reset(disc);

              readln(disc,title);
              readln(disc);
              readln(disc,nsamples,error);
              readln(disc);
              readln(disc);
              readln(disc);
              for i:=1 to nsamples do
                 begin
                    readln(disc,data[i],dumstr,dumstr,hourstr,dumstr,minstr,dumstr,secstr);
                    val(hourstr,time[i].hour,code);
                    val(minstr,time[i].min,code);
                    val(secstr,time[i].sec,code);
                 end;

              close(disc);
           end;
  END;


{*************************************************************************}
{N**   MODULE NAME        :  SAVEMFDATA                                   ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          :  saves massflow data on file                  ***}
{S**   CALL SEQUENCE      :  SAVEMFDATA(filename, massflow data record    ***}
{I**   INPUT PARAMETERS   :  filename, data record containing massflow   ***}
{***                          data                                       ***}
{O**   OUTPUT PARAMETERS  :  none                                         ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  Crt,Types                                    ***}
{E**   ERROR CONDITIONS   :  if file already exists                       ***}
{C**   COMMENTS           :  for more information about the massflow      ***}
{***                          data record see TYPES                      ***}
{*************************************************************************}

PROCEDURE savemfdata(VAR name:str50;data:massflowdata);

  VAR
     i:integer;
     disc:text;
     message:string[80];
     flag:boolean;

  BEGIN
     CHKDIRECTORY(name,flag);
     if flag=true then CHKOVERWRITE(name,flag);

     if flag=true then
        with data do
           begin
              message:=' Please Wait - Saving File: ';
              insert(name,message,28);
              clrscr;HIGHLIGHT(0);gotoXY(10,10);writeln(message);normvideo;

              assign(disc,name);
              rewrite(disc);
```

```
            writeln(disc,'Title: ',title);
            writeln(disc,'No of Samples:      Error [t/hr]:');
            writeln(disc,nsamples:4,'                ',error:8:2);
            writeln(disc);
            writeln(disc,'  Massflow    Time');
            writeln(disc,'  [t/hr]      [hrs:min:sec]');
            for i:=1 to nsamples do
               writeln(disc,'  ',data[i]:8:2,' ',time[i].hour:2,':', time[i].min:2,':',time[i].sec:2);

            close(disc);
         end;
   END;


{*************************************************************************}
{N**   MODULE NAME       : LOADCONST                                      ***}
{***   ------------------------------------------------------------       ***}
{A**   PROCEDURE         : loads the constants used in the massflow       ***}
{***                       determination process                          ***}
{S**   CALL SEQUENCE     : LOADCONST(const)                               ***}
{I**   INPUT PARAMETERS  : none                                           ***}
{O**   OUTPUT PARAMETERS : data record containing speed, power, cali-     ***}
{***                       bration and friction constants                 ***}
{G**   GLOBAL VARIABLES  : none                                           ***}
{M**   MODULES CALLED     : none                                           ***}
{E**   ERROR CONDITIONS   : none                                           ***}
{C**   COMMENTS          : for details about the data record see TYPES ***}
{*************************************************************************}

PROCEDURE Loadconst(VAR Csp:constant);

   VAR
      disc:text;

   BEGIN
      assign(disc,'const.dat');
      reset(disc);

      with Csp do
         begin
            readln(disc,speed);
            readln(disc,power);
            readln(disc,dh);
            readln(disc,offset);
            readln(disc,slope);
            readln(disc,error);
            readln(disc,Fa);
            readln(disc,Fb);

            close(disc);
         end;
   END;


{*************************************************************************}
{N**   MODULE NAME       : SAVECONST                                      ***}
{***   ------------------------------------------------------------       ***}
{A**   PROCEDURE         : saves the constants used in the massflow       ***}
{***                       determination process                          ***}
{S**   CALL SEQUENCE     : SAVECONST(const)                               ***}
{I**   INPUT PARAMETERS  : none                                           ***}
{O**   OUTPUT PARAMETERS : data record containing speed, power, cali-     ***}
{***                       bration and friction constants                 ***}
{G**   GLOBAL VARIABLES  : none                                           ***}
{M**   MODULES CALLED     : none                                           ***}
{E**   ERROR CONDITIONS   : none                                           ***}
{C**   COMMENTS          : for details about the data record see TYPES ***}
{*************************************************************************}

PROCEDURE Saveconst(VAR Csp:constant);

   VAR
      disc:text;

   BEGIN
      assign(disc,'const.dat');
      rewrite(disc);

      with Csp do
         begin
            writeln(disc,speed);
            writeln(disc,power);
            writeln(disc,dh);
            writeln(disc,offset);
            writeln(disc,slope);
            writeln(disc,error);
            writeln(disc,Fa);
            writeln(disc,Fb);

            close(disc);
         end;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : LOADCONSTM                                    ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : loads the constants used in the massflow      ***}
{***                        determining process                           ***}
{S**   CALL SEQUENCE      : LOADCONST(massflowconst)                      ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : data record containing massflow constants     ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : none                                          ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about the data record see TYPES   ***}
{*************************************************************************}

PROCEDURE Loadconstm(VAR Cm:massflowconst);

   VAR
      disc:text;

   BEGIN
      assign(disc,'constm.dat');
      reset(disc);

      with Cm do
         begin
            readln(disc,mass);
            readln(disc,vincl);
            readln(disc,dh);
            readln(disc,di);
            readln(disc,dch1);
            readln(disc,dch2);
            readln(disc,nuc);
         end;

      close(disc);

   END;


{*************************************************************************}
{N**   MODULE NAME        :  LOADLOOPDATA                                 ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          :  loads the no of wait loops to be executed    ***}
{***                        in the datalogging process                    ***}
{S**   CALL SEQUENCE      :  LOADLOOPDATA(noloop)                         ***}
{I**   INPUT PARAMETERS   :  none                                          ***}
{O**   OUTPUT PARAMETERS  :  no of waitloops                               ***}
{G**   GLOBAL VARIABLES   :  none                                          ***}
{M**   MODULES CALLED     :  none                                          ***}
{E**   ERROR CONDITIONS   :  none                                          ***}
{C**   COMMENTS           :  none                                         ***}
{*************************************************************************}

PROCEDURE loadloopdata(VAR noloop:longint);

   VAR
      disc:text;

  BEGIN
        assign(disc,'timedata.dat');
  {$I-}
        reset(disc);
  {$I+}
        if IOResult<>0 then noloop:=0
        else
           begin
              reset(disc);
              readln(disc,noloop);
              close(disc);
           end;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        :  SAVELOOPDATA                                 ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          :  stores the no of waitloops to be executed    ***}
{***                         in the datalogging process                   ***}
{S**   CALL SEQUENCE      :  SAVELOOPDATA(noloop)                         ***}
{I**   INPUT PARAMETERS   :  noloop                                       ***}
{O**   OUTPUT PARAMETERS  :  none                                         ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  none                                         ***}
{E**   ERROR CONDITIONS   :  none                                         ***}
{C**   COMMENTS           :  none                                         ***}
{*************************************************************************}

PROCEDURE saveloopdata(noloop:longint);

   VAR
      disc:text;
      opt:char;

   BEGIN
      assign(disc,'timedata.dat');
      rewrite(disc);
      writeln(disc,noloop);
      close(disc);
   END;


{*************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0       KCAD     26-02-90                                     ***}
{***                                                                     ***}
{***   UNITEND FILEHAND.PAS                                              ***}
{*************************************************************************}
END.
```

C.3.3 Math.pas

```
{*************************************************************************}
{***   UNIT PROCEDURE LIBRARY TO PERFORM CERTAIN MATHEMATICAL          ***}
{***        MANIPULATIONS                                              ***}
{***   FILE: MATH.PAS                                                  ***}
{*************************************************************************}

UNIT math;

{*************************************************************************}
                        INTERFACE
{*************************************************************************}

   USES crt,types;

   VAR
      U:matrixbig;
      Y:vectorbig;

   PROCEDURE matrixmult(A,B:matrixsml;VAR C:matrixsml);
   PROCEDURE UtU(VAR B:matrixsml);
   PROCEDURE UtY(VAR B:vectorsml);
   PROCEDURE gauss_N(A:matrixsml;VAR B:matrixsml);
   PROCEDURE gauss_1(A:matrixsml;VAR B:vectorsml);
   PROCEDURE online_meansdev(value,NT:real;VAR mean,stddev,state:real);
   PROCEDURE semi_meansdev(n:integer;value:real;VAR mean,stddev,state1,state2:real);
   PROCEDURE meancalc(data:responsedata;ch,start,range:integer;VAR mean:integer;VAR dev:real);
   PROCEDURE regress(corrdata:correlationdata;VAR regdata:regressiondata);
   FUNCTION area(data:responsedata;ch,start,range:integer):real;
   FUNCTION quadarea(data:responsedata;ch,start,range:integer):real;
   FUNCTION cubicarea(data:responsedata;ch,start,range:integer):real;
   FUNCTION tpo(number:real;n:integer):real;
   PROCEDURE filter(breakfreq:real;ch:integer;VAR data:responsedata);
   PROCEDURE offset(VAR data:responsedata);
   PROCEDURE norm(VAR data:responsedata);
   PROCEDURE disregard_up_to_step(VAR data:responsedata);
   PROCEDURE chgfreq(newfreq:real;VAR data:responsedata);



{*************************************************************************}
                        IMPLEMENTATION
{*************************************************************************}

{*************************************************************************}
{N**   MODULE NAME       : MATRIXMULT                                   ***}
{***   ------------------------------------------------------------    ***}
{A**   FUNCTION          : matrix multiplies two matrices, i.e.C=A*B    ***}
{S**   CALL SEQUENCE     : MATRIXMULT(A,B,C)                            ***}
{I**   INPUT PARAMETERS  : A and B of type Matrixsml                    ***}
{O**   OUTPUT PARAMETERS : C of type Matrixsml                         ***}
{G**   GLOBAL VARIABLES  : none                                        ***}
{M**   MODULES CALLED    : Crt, Types                                   ***}
{E**   ERROR CONDITIONS  : matrices uncompatible                       ***}
{C**   COMMENTS          : for info about Matrixsml see TYPES           ***}
{*************************************************************************}

PROCEDURE matrixmult(A,B:matrixsml;VAR C:matrixsml);

   VAR
      i,j,k:integer;
      sum:real;


   BEGIN
      if A.cols<>B.rows then
         begin
            clrscr;HIGHLIGHT(1);gotoXY(20,10);
            writeln('!!! matrices uncompatible !!!');normvideo;
            gotoXY(20,15);writeln('press <ENTER> to continue');readln;
         end

      else
         begin
            for i:=1 to A.rows do
               for j:=1 to A.cols do
                  begin
                     sum:=0;
                     for k:=1 to A.cols do sum:=sum + A.a[i,k]*B.a[k,j];
                     C.a[i,j]:=sum;
                  end;

           C.cols:=B.cols;
           C.rows:=A.rows;
       end;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : UtU                                          ***}
{***   ------------------------------------------------------------      ***}
{A**   FUNCTION           : matrix multiplies the transpose of a matrixU***}
{***                        with the matrix U itself i.e. B=Ut*U         ***}
{S**   CALL SEQUENCE      : UtU(B)                                       ***}
{I**   INPUT PARAMETERS   : U of type Matrixbig                          ***}
{O**   OUTPUT PARAMETERS  : B (type matrixsml)                           ***}
{G**   GLOBAL VARIABLES   : U                                           ***}
{M**   MODULES CALLED     : Types                                        ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : for details about matrixbig and matrixsml    ***}
{***                        see TYPES                                    ***}
{*************************************************************************}

PROCEDURE UtU(VAR B:matrixsml);

   VAR
      i,j,k:integer;
      sum:real;


   BEGIN
      for i:=1 to U.cols do
         for j:=1 to U.cols do
            begin
               sum:=0;
               for k:=1 to U.rows do sum:=sum + U.a[k,j]*U.a[k,i];
               B.a[i,j]:=sum;
            end;

      B.cols:=U.cols;
      B.rows:=U.cols;
   END;


{*************************************************************************}
{N**   MODULE NAME        : UtY                                          ***}
{***   ------------------------------------------------------------      ***}
{A**   FUNCTION           : matrix multiplies the transpose of a matrixU***}
{***                        with the vector Y i.e. B=Ut*Y                ***}
{S**   CALL SEQUENCE      : UtY(B)                                       ***}
{I**   INPUT PARAMETERS   : U of type Matrixbig and Y of type Vectorbig ***}
{O**   OUTPUT PARAMETERS  : B (type vectorsml)                           ***}
{G**   GLOBAL VARIABLES   : U and Y                                      ***}
{M**   MODULES CALLED     : Crt, Types                                   ***}
{E**   ERROR CONDITIONS   : matrix U uncompatible with vector Y         ***}
{C**   COMMENTS           : for details about Matrixbig, Vectorbig and   ***}
{***                        Vectorsml see TYPES                          ***}
{*************************************************************************}

PROCEDURE UtY(VAR B:vectorsml);

   VAR
      i,j,k:integer;
      sum:real;


   BEGIN
      if U.rows<>Y.elements then
         begin
            clrscr;HIGHLIGHT(1);gotoXY(10,10);
            writeln('!!! matrix U uncompatible with vector Y !!!');normvideo;
            gotoXY(10,15);writeln('press <ENTER> to continue');readln;
         end

      else
         begin
           for i:=1 to U.cols do
               begin
                  sum:=0;
                  for k:=1 to U.rows do sum:=sum + Y.v[k]*U.a[k,i];
                  B.v[i]:=sum;
               end;

            B.elements:=U.cols;
         end;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : GAUSS_N                                       ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : solves a set of simultaneous equations with ***}
{***                        N solution vectors, using the Gauss          ***}
{***                        Reduction process with Partial Pivoting for ***}
{***                        accuracy                                     ***}
{S**   CALL SEQUENCE      : GAUSS_n(A,B)                                  ***}
{I**   INPUT PARAMETERS   : coefficients matrix A, solution matrix B     ***}
{***                        (both of type matrixsml)                     ***}
{O**   OUTPUT PARAMETERS  : Gauss reduced solution matrix B (matrixsml) ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED     : Crt, Types                                    ***}
{E**   ERROR CONDITIONS   : - if coefficient matrix not square          ***}
{***                        - dimensions of solution vector do not match***}
{***                          dimensions of coefficient matrix          ***}
{C**   COMMENTS           : - the solution vectors are represented by a ***}
{***                          matrix                                     ***}
{***                        - for details about matrixsml see TYPES     ***}
{*************************************************************************}

PROCEDURE gauss_N(A:matrixsml;VAR B:matrixsml);

   VAR
      n,nn,i,j,k,largindex:integer;
      largcoef,temp,factor:real;


   BEGIN
         if (A.rows<>A.cols)  then
         begin
            clrscr;HIGHLIGHT(1);gotoXY(15,10);writeln('!!! coefficient matrix not square !!!');normvideo;
            gotoXY(15,15);writeln('press <ENTER> to continue');readln;
         end

      else if (A.rows<>B.rows) then
         begin
            clrscr;HIGHLIGHT(1);
            gotoXY(10,10);writeln('!!! dimensions of solution matrix do not match
                                   coefficient matrix !!!');
            normvideo;
            gotoXY(10,15);writeln('press <ENTER> to continue');readln;
         end

      else
         begin
            n:=A.cols;
            nn:=B.cols;

            for i:=1 to n-1 do
               begin
                  largcoef:=A.a[i,i];     {look for highest leading Coefficient}
                  largindex:=i;
                  for j:=i+1 to n do
                     if abs(A.a[j,i]) > largcoef then
                        begin
                           largcoef:=abs(A.a[j,i]);
                           largindex:=j;
                        end;

                  if largindex<>i then   {interchange rows if necessary}
                     begin
                        for j:=i to n do
                           begin
                              temp:=A.a[largindex,j];
                              A.a[largindex,j]:=A.a[i,j];
                              A.a[i,j]:=temp;
                           end;
                        for j:=1 to nn do
                           begin
                              temp:=B.a[largindex,j];
                              B.a[largindex,j]:=B.a[i,j];
                              B.a[i,j]:=temp;
                           end;
                     end;


                  for j:=i+1 to n do     {subtract multiples from row i from lower rows}
                     begin
                        factor:=A.a[j,i]/A.a[i,i];
                        for k:=i+1 to n do A.a[j,k]:=A.a[j,k] - factor*A.a[i,k];
                        for k:=1 to nn do B.a[j,k]:=B.a[j,k] - factor*B.a[i,k];
                     end;
               end;


            for k:=1 to nn do B.a[n,k]:=B.a[n,k]/A.a[n,n];     {back substitution}
               for i:=n-1 downto 1 do
                  for k:=1 to nn do
                     begin
                        factor:=B.a[i,k];
                        for j:=i+1 to n do factor:=factor - A.a[i,j]*B.a[j,k];
                        B.a[i,k]:=factor/A.a[i,i];
                     end;
         end;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : GAUSS_1                                       ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : solves a set of simultaneous equations using***}
{***                        the Gauss reduction process with partial     ***}
{***                        pivoting for accuracy                        ***}
{S**   CALL SEQUENCE      : GAUSS_1(A,B)                                  ***}
{I**   INPUT PARAMETERS   : coefficients matrix A (type matrixsml)       ***}
{***                        solution vector B (type vectorsml)           ***}
{O**   OUTPUT PARAMETERS  : Gauss reduced solution vector B (vectorsml)  ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED     : Crt, Types                                    ***}
{E**   ERROR CONDITIONS   : - if coefficient matrix not square           ***}
{***                        - dimensions of solution vector do not match***}
{***                          dimensions of coefficient matrix           ***}
{C**   COMMENTS           : - for details about matrixsml see TYPES       ***}
{*************************************************************************}

PROCEDURE gauss_1(A:matrixsml;VAR B:vectorsml);

   VAR
      n,i,j,k,largindex:integer;
      largcoef,temp,factor:real;


   BEGIN
      if (A.rows<>A.cols)  then
         begin
            clrscr;HIGHLIGHT(1);gotoXY(15,10);writeln('!!! coefficient matrix not square !!!');normvideo;
            gotoXY(15,15);writeln('press <ENTER> to continue');readln;
         end

      else if (A.rows<>B.elements) then
         begin
            clrscr;HIGHLIGHT(1);
            gotoXY(10,10);writeln('!!! dimensions of solution vector do not match
                                   coefficient matrix !!!');normvideo;
            gotoXY(10,15);writeln('press <ENTER> to continue');readln;
         end

      else
         begin
            n:=A.cols;

            for i:=1 to n-1 do
               begin
                  largcoef:=A.a[i,i];     {look for highest leading coefficient}
                  largindex:=i;
                  for j:=i+1 to n do
                     if abs(A.a[j,i]) > largcoef then
                        begin
                           largcoef:=abs(A.a[j,i]);
                           largindex:=j;
                        end;

                  if largindex<>i then   {interchange rows if necessary}
                     begin
                        for j:=i to n do
                           begin
                              temp:=A.a[largindex,j];
                              A.a[largindex,j]:=A.a[i,j];
                              A.a[i,j]:=temp;
                           end;
                        temp:=B.v[largindex];
                        B.v[largindex]:=B.v[i];
                        B.v[i]:=temp;
                     end;


                  for j:=i+1 to n do     {subtract multiples from row i from lower rows}
                     begin
                        factor:=A.a[j,i]/A.a[i,i];
                        for k:=i+1 to n do A.a[j,k]:=A.a[j,k] - factor*A.a[i,k];
                        B.v[j]:=B.v[j] - factor*B.v[i];
                     end;
               end;


            B.v[n]:=B.v[n]/A.a[n,n];     {back substitution}
            for i:=n-1 downto 1 do
               begin
                  factor:=B.v[i];
                  for j:=i+1 to n do factor:=factor - A.a[i,j]*B.v[j];
                  B.v[i]:=factor/A.a[i,i];
               end;
         end;
   END;
```

```
{************************************************************************}
{N**   MODULE NAME        : ONLINE_MEANSDEV                              ***}
{***   ------------------------------------------------------------     ***}
{A**   PROCEDURE          : determines the mean and standard deviation   ***}
{***                        in an on-line fashion using a recursive mean***}
{***                        and standard deviation estimator, making use***}
{***                        of a state to prevent numeric overflows      ***}
{S**   CALL SEQUENCE      : ONLINE_MEANSDEV(value,alpha,mean,sdev,state)***}
{I**   INPUT PARAMETERS   : new value,NT,mean,std.deviation,state        ***}
{O**   OUTPUT PARAMETERS  : updated mean, std.deviation and state        ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED      : none                                         ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           : - a:=exp(-1/T); T=time constant of detector  ***}
{***                          => NT:=a/(1-a)                            ***}
{***                        - state = dum of value squared / NT; Initial***}
{***                          condition = mean of last five values^2     ***}
{************************************************************************}

PROCEDURE online_meansdev(value,NT:real;VAR mean,stddev,state:real);

   BEGIN
      mean:=(NT*mean + value) / (NT+1);
      state:=(NT*state + sqr(value)) / (NT+1);
      stddev:=sqrt(NT*(state - sqr(mean))/(NT-1));
   END;


{************************************************************************}
{N**   MODULE NAME        : SEMI_MEANSDEV                                ***}
{***   ------------------------------------------------------------     ***}
{A**   PROCEDURE          : determines the mean and standard deviation   ***}
{***                        in an on-line fashion, but not for large     ***}
{***                        number of values as numeric overflows occur  ***}
{S**   CALL SEQUENCE      : SEMI_MEANSDEV(n,value,mean,sdev,states 1+2)  ***}
{I**   INPUT PARAMETERS   : no of values,new value,mean,std.dev,states   ***}
{O**   OUTPUT PARAMETERS  : updated mean, std. deviation and states 1+2  ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED      : none                                         ***}
{E**   ERROR CONDITIONS   : for many values a numeric overflow will      ***}
{***                        occur. Thus rather use online_meansdev       ***}
{C**   COMMENTS           : state1 = sum of values; state2 = sum of      ***}
{***                        values squared. Initial conditions = 0       ***}
{************************************************************************}

PROCEDURE semi_meansdev(n:integer;value:real;VAR mean,stddev,state1,state2:real);

   BEGIN
      state1:=state1+sqr(value);
      state2:=state2+value;
      mean:=state2/n;
      stddev:=sqrt((state1 - 2*mean*state2 + n*sqr(mean))/(n-1));
   END;


{************************************************************************}
{N**   MODULE NAME        :  MEANCALC                                    ***}
{***   ------------------------------------------------------------     ***}
{A**   PROCEDURE          :  calculates the mean and standard deviation  ***}
{***                          of a specified range of numbers            ***}
{S**   CALL SEQUENCE      :  MEANCALC(data,ch,start,range,mean,dev)      ***}
{I**   INPUT PARAMETERS   :  data record, channel to calculate mean from***}
{***                          , starting pointer and range over which to ***}
{***                          calculate the mean                         ***}
{O**   OUTPUT PARAMETERS  :  mean and standard deviation                 ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED      :  Types                                       ***}
{E**   ERROR CONDITIONS   :  none                                         ***}
{C**   COMMENTS           :  details about the data record in TYPES      ***}
{************************************************************************}

PROCEDURE meancalc(data:responsedata;ch,start,range:integer;VAR mean:integer;
                   VAR dev:real);

   VAR
      i:integer;
      dum:real;

   BEGIN
      dum:=0;
      for i:=start to (start+range) do
         dum:=dum+(data.data[ch,i]/range);

      mean:=round(dum);

      dev:=0;
      for i:=start to (start+range) do
      begin
         dum:=data.data[ch,i]-mean;
         dev:=dev+sqr(dum)/(range-1);
       end;
      dev:=sqrt(dev);
   END;
```

```
{************************************************************************}
{N**   MODULE NAME        : REGRESS                                      ***}
{***   ------------------------------------------------------------   ***}
{A**   PROCEDURE          : performs a linear regression on two arrays  ***}
{***                        of data and determines the constants for a  ***}
{***                        95% confidence band                         ***}
{S**   CALL SEQUENCE      : REGRESS(corrdata,regdata)                   ***}
{I**   INPUT PARAMETERS   : data record (type correlationdata) on which ***}
{***                        regression is to be performed               ***}
{O**   OUTPUT PARAMETERS  : regression results (type regressiondata)    ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : TYPES                                       ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : details about regressiondata and correlation***}
{***                        data in TYPES                               ***}
{************************************************************************}

PROCEDURE regress(corrdata:correlationdata;VAR regdata:regressiondata);

   VAR
      sumx,sumx2,sumy,sumy2,sumxy,dx,dy,dxy,d,Fi:real;
      F:array[1..3,1..20] of real;
      i,j,n,nn:integer;

   BEGIN
      F[1,1]:=10;F[2,1]:=15;F[3,1]:=4.96;   {F-distribution factors for a 95% confidence band}
      F[1,2]:=15;F[2,2]:=20;F[3,2]:=4.54;
      F[1,3]:=20;F[2,3]:=25;F[3,3]:=4.35;
      F[1,4]:=25;F[2,4]:=30;F[3,4]:=4.24;
      F[1,5]:=30;F[2,5]:=40;F[3,5]:=4.17;
      F[1,6]:=40;F[2,6]:=50;F[3,6]:=4.08;
      F[1,7]:=50;F[2,7]:=60;F[3,7]:=4.03;
      F[1,8]:=60;F[2,8]:=70;F[3,8]:=4.00;
      F[1,9]:=70;F[2,9]:=80;F[3,9]:=3.98;
      F[1,10]:=80;F[2,10]:=90;F[3,10]:=3.96;
      F[1,11]:=90;F[2,11]:=100;F[3,11]:=3.95;
      F[1,12]:=100;F[2,12]:=125;F[3,12]:=3.94;
      F[1,13]:=125;F[2,13]:=150;F[3,13]:=3.92;
      F[1,14]:=150;F[2,14]:=200;F[3,14]:=3.90;
      F[1,15]:=200;F[2,15]:=300;F[3,15]:=3.89;
      F[1,16]:=300;F[2,16]:=500;F[3,16]:=3.87;
      F[1,17]:=500;F[2,17]:=1000;F[3,17]:=3.86;
      F[1,18]:=1000;F[2,18]:=1000;F[3,18]:=3.85;

      n:=corrdata.nsamples;
      for j:=1 to 2 do
         begin
            sumx:=0; sumx2:=0;
            sumy:=0; sumy2:=0;
            sumxy:=0;

            with corrdata do
               for i:=1 to n do
                  begin
                     sumx:= sumx + data[1,i];
                     sumx2:= sumx2 + sqr(data[1,i]);
                     sumy:= sumy + data[j+1,i];
                     sumy2:= sumy2 + sqr(data[j+1,i]);
                     sumxy:= sumxy + data[j+1,i]*data[1,i];
                  end;

            dx:= sumx2 - sqr(sumx)/n;
            dy:= sumy2 - sqr(sumy)/n;
            dxy:= sumxy - sumx*sumy/n;
            d:=dx*dy - sqr(dxy);

            with regdata do
               begin
                  ymean[j]:=sumy/n;
                  xmean[j]:=sumx/n;

                  slope[j]:=dxy/dx;
                  yconst[j]:=(sumx2*sumy - sumxy*sumx) / (n*dx);
                  r[j]:=dxy / sqrt(dx*dy);
                  yerror[j]:=sqrt(d/(dx*(n-2)));

                  if n-2<=10 then Fi:=4.96
                  else if n-2>=1000 then Fi:=3.85
                  else
                     begin
                        i:=0;
                        nn:=-1;
                        repeat
                           i:=i+1;
                           if F[2,i]>n-2 then nn:=i;
                        until nn=i;
                        Fi:=(1/F[1,nn]-1/(n-2)) / (1/F[1,nn]-1/F[2,nn])
                            * (F[3,nn+1]-F[3,nn])
                            + F[3,nn];
                     end;

                  C1[j]:=Fi*sqr(yerror[j])/dx;
                  C2[j]:=Fi*sqr(yerror[j]) * (1+1/n);
                  error[j]:=sqrt(abs(C2[j]/(C1[j]-sqr(slope[j]))));
               end;
         end;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        :  AREA                                         ***}
{***   ------------------------------------------------------------      ***}
{A**   FUNCTION           :  determines the area under a response over    ***}
{***                         a specified range using the trapezoidal     ***}
{***                         rule                                        ***}
{S**   CALL SEQUENCE      :  AREA(data,ch,start,range)                    ***}
{I**   INPUT PARAMETERS   :  data record,channel to determine area from,***}
{***                         first sample of range, range in samples     ***}
{O**   OUTPUT PARAMETERS  :  area under curve                            ***}
{G**   GLOBAL VARIABLES   :  none                                        ***}
{M**   MODULES CALLED     :  Types                                       ***}
{E**   ERROR CONDITIONS   :  none                                        ***}
{C**   COMMENTS           :  details about data record in TYPES          ***}
{*************************************************************************}

FUNCTION area(data:responsedata;ch,start,range:integer):real;

   VAR
      i:integer;
      dt,sum:real;

   BEGIN
      dt:=1/data.frequency;
      sum:=0;

      for i:=(start+1) to (start+range) do
         sum:=sum+(data.data[ch,i]+data.data[ch,i-1])*dt/2;

      area:=sum;
   END;


{*************************************************************************}
{N**   MODULE NAME        :  QUADAREA                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   FUNCTION           :  determines the area under the quadratic of ***}
{***                         a response over a specified range using the***}
{***                         trapezoidal rule                            ***}
{S**   CALL SEQUENCE      :  QUADAREA(data,ch,start,range)                ***}
{I**   INPUT PARAMETERS   :  data record,channel to determine area from,***}
{***                         first sample of range, range in samples     ***}
{O**   OUTPUT PARAMETERS  :  area under quadratic of curve                ***}
{G**   GLOBAL VARIABLES   :  none                                        ***}
{M**   MODULES CALLED     :  Types                                       ***}
{E**   ERROR CONDITIONS   :  none                                        ***}
{C**   COMMENTS           :  details about data record in TYPES          ***}
{*************************************************************************}

FUNCTION quadarea(data:responsedata;ch,start,range:integer):real;

   VAR
      i:integer;
      dt,sum:real;

   BEGIN
      dt:=1/data.frequency;
      sum:=0;

      for i:=(start+1) to (start+range) do
         sum:=sum+(TPO(data.data[ch,i],2)+TPO(data.data[ch,i-1],2))*dt/2;
      quadarea:=sum;
   END;


{*************************************************************************}
{N**   MODULE NAME        :  CUBICAREA                                    ***}
{***   ------------------------------------------------------------      ***}
{A**   FUNCTION           :  determines the area under the cubic of       ***}
{***                         a response over a specified range using the***}
{***                         trapezoidal rule                            ***}
{S**   CALL SEQUENCE      :  CUBICAREA(data,ch,start,range)               ***}
{I**   INPUT PARAMETERS   :  data record,channel to determine area from,***}
{***                         first sample of range, range in samples     ***}
{O**   OUTPUT PARAMETERS  :  area under cubic of curve                    ***}
{G**   GLOBAL VARIABLES   :  none                                        ***}
{M**   MODULES CALLED     :  Types                                       ***}
{E**   ERROR CONDITIONS   :  none                                        ***}
{C**   COMMENTS           :  details about data record in TYPES          ***}
{*************************************************************************}

FUNCTION cubicarea(data:responsedata;ch,start,range:integer):real;

   VAR
      i:integer;
      dt,sum:real;

   BEGIN
      dt:=1/data.frequency;
      sum:=0;

      for i:=(start+1) to (start+range) do
         sum:=sum+(TPO(data.data[ch,i],3)+TPO(data.data[ch,i-1],3))*dt/2;
      cubicarea:=sum;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        :  TPO                                          ***}
{***   ------------------------------------------------------------      ***}
{A**   FUNCTION           :  determines value of a number raised to the ***}
{***                          power n                                     ***}
{S**   CALL SEQUENCE      :  TPO(number,n)                                ***}
{I**   INPUT PARAMETERS   :  number to be raised, dimension of power(n) ***}
{O**   OUTPUT PARAMETERS  :  number^n                                     ***}
{G**   GLOBAL VARIABLES   :  none                                        ***}
{M**   MODULES CALLED     :  none                                        ***}
{E**   ERROR CONDITIONS   :  none                                        ***}
{C**   COMMENTS           :  none                                        ***}
{*************************************************************************}

FUNCTION tpo(number:real;n:integer):real;

   VAR
      i:integer;
      product:real;

   BEGIN
      product:=1;
      for i:=1 to n do
         product:=product*number;

      tpo:=product;
   END;

{*************************************************************************}
{N**   MODULE NAME        :  FILTER                                       ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          :  filters a response in the time domain using***}
{***                          a first order filter                        ***}
{S**   CALL SEQUENCE      :  FILTER(breakfrequency,channel,datarecord)    ***}
{I**   INPUT PARAMETERS   :  break frequency of filter, channel of data ***}
{***                          recorf to filter, data record              ***}
{O**   OUTPUT PARAMETERS  :  filtered data record                         ***}
{G**   GLOBAL VARIABLES   :  None                                        ***}
{M**   MODULES CALLED     :  Types                                       ***}
{E**   ERROR CONDITIONS   :  None                                        ***}
{C**   COMMENTS           :  - details about data record in TYPES        ***}
{*************************************************************************}

PROCEDURE filter(breakfreq:real;ch:integer;VAR data:responsedata);

   VAR
       i,initpoint:integer;
       factor,initcond:real;

   BEGIN
      initcond:=0;
      initpoint:=trunc(4*data.frequency/breakfreq);
      if initpoint>data.nsamples then initpoint:=data.nsamples;
      for i:=0 to initpoint do initcond:=initcond+data.data[ch,i]/(initpoint+1);

      factor:=exp(-breakfreq/data.frequency);

      data.data[ch,0]:=round(data.data[ch,0]*(1-factor)+initcond*factor);
      for i:=1 to (data.nsamples-1) do
         data.data[ch,i]:=round(data.data[ch,i]*(1-factor)+data.data[ch,i-1]*factor);
   END;


{*************************************************************************}
{N**   MODULE NAME        :  OFFSET                                       ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          :  subtracts the bias from a response           ***}
{S**   CALL SEQUENCE      :  OFFSET(datarecord)                           ***}
{I**   INPUT PARAMETERS   :  data record                                  ***}
{O**   OUTPUT PARAMETERS  :  processed data record                        ***}
{G**   GLOBAL VARIABLES   :  none                                        ***}
{M**   MODULES CALLED     :  Types,Crt                                   ***}
{E**   ERROR CONDITIONS   :  none                                        ***}
{C**   COMMENTS           :  - offset is subtracted from all channels    ***}
{***                          - details about data record in TYPES       ***}
{*************************************************************************}

PROCEDURE offset(VAR data:responsedata);

   VAR
      i,j,meanfin,meaninit:integer;
      devinit,devfin:real;
      error:char;

   BEGIN
      error:='n';
      if data.whenstep<3 then
         begin
            normvideo;clrscr;gotoXY(10,10);HIGHLIGHT(1);writeln('!!! ERROR IN OFFSET !!!');
            gotoXY(10,12);HIGHLIGHT(0);writeln('TOO LITTLE SAMPLES BEFORE STEP');
            gotoXY(10,16);writeln('Press any Key to continue');
            repeat until keypressed;readln;
            error:='y';
         end;
```

```
      for j:=1 to data.nochannel do
         begin
            if error='y' then meaninit:=data.data[j,0]
            else MEANCALC(data,j,0,(data.whenstep-1),meaninit,devinit);

            MEANCALC(data,j,(data.nsamples-data.whenstep),(data.whenstep-1),meanfin,devfin);

            if data.step>0 then
               begin
                  for i:=0 to data.nsamples-1 do
                     data.data[j,i]:=round(data.data[j,i]-meaninit);
                  meanfin:=round(meanfin-meaninit);
                  meaninit:=0;
               end
            else
               begin
                  for i:=0 to data.nsamples-1 do
                     data.data[j,i]:=round(data.data[j,i]-meanfin);
                  meaninit:=round(meaninit-meanfin);
                  meanfin:=0;
               end;

            data.data[j,data.nsamples]:=meanfin;
            data.data[j,0]:=meaninit;
         end;
   END;


{*************************************************************************}
{N**   MODULE NAME        :  NORM                                         ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          :  normalizes a response in respect to the      ***}
{***                         step applied to it                           ***}
{S**   CALL SEQUENCE      :  NORM(datarecord)                             ***}
{I**   INPUT PARAMETERS   :  data record                                  ***}
{O**   OUTPUT PARAMETERS  :  normalized data record                       ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  Types                                        ***}
{E**   ERROR CONDITIONS   :  none                                         ***}
{C**   COMMENTS           :  - all channels are normalized                ***}
{***                         - details about data record in TYPES         ***}
{*************************************************************************}

PROCEDURE norm(VAR data:responsedata);

   VAR
      i,j:integer;

   BEGIN
      for j:=1 to data.nochannel do
         for i:=0 to data.nsamples-1 do data.data[j,i]:=round(data.data[j,i]/data.step);
      data.step:=1;
   END;


{*************************************************************************}
{N**   MODULE NAME        :  DISREGARD_UP_TO_STEP                         ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          :  disregards all samples up to the step        ***}
{S**   CALL SEQUENCE      :  DISREGARD_UP_TP_STEP(datarecord)             ***}
{I**   INPUT PARAMETERS   :  data record                                  ***}
{O**   OUTPUT PARAMETERS  :  processed data record                       ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  Types                                        ***}
{E**   ERROR CONDITIONS   :  none                                         ***}
{C**   COMMENTS           :  - samples before step disregarded for all    ***}
{***                            channels                                  ***}
{***                         - details about data record in TYPES         ***}
{*************************************************************************}

PROCEDURE disregard_up_to_step(VAR data:responsedata);

   VAR
      i,j:integer;

   BEGIN
      for i:= data.whenstep+1 to data.nsamples-1 do
         for j:=1 to data.nochannel do
            data.data[j,i-data.whenstep]:=data.data[j,i];
      data.nsamples:=data.nsamples-data.whenstep;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        :  CHGFREQ                                      ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          :  changes the sampling frequency of the re-    ***}
{***                          sponse and adjusts the dataarray accord-    ***}
{***                          ingly                                       ***}
{S**   CALL SEQUENCE      :  CHGFREQ(newfreq,datarecord)                  ***}
{I**   INPUT PARAMETERS   :  new sampling frequency, data record          ***}
{O**   OUTPUT PARAMETERS  :  processed data record                        ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  Types                                        ***}
{E**   ERROR CONDITIONS   :  none                                         ***}
{C**   COMMENTS           :  - frequency changed for all Channels         ***}
{***                          - details about data record in TYPES        ***}
{*************************************************************************}

PROCEDURE chgfreq(newfreq:real;VAR data:responsedata);

   VAR
      count,i,j,k:integer;

   BEGIN
      count:=round(data.frequency/newfreq);
      data.frequency:=data.frequency/count;
      data.nsamples:=trunc(data.nsamples/count);

      i:=0;
      for j:=0 to data.nsamples-1 do
         begin
            for k:=1 to data.nochannel do
               data.data[k,j]:=data.data[k,i];
            i:=i+count;
         end;
   END;


{*************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0       KCAD     27-02-90                                     ***}
{***                                                                     ***}
{***   UNITEND MATH.PAS                                                  ***}
{*************************************************************************}
END.
```

C.3.4 Plotgraf.pas

```
{*************************************************************************}
{***   UNIT PROCEDURE LIBRARY TO PREPARE GRAPHS AND TO PLOT GRAPHS     ***}
{***   FILE: PLOTGRAF.PAS                                              ***}
{*************************************************************************}

UNIT plotgraf;

{*************************************************************************}
                        INTERFACE
{*************************************************************************}

   USES crt,drivers,fonts,graph,types,filehand,plotgrad;

   PROCEDURE graphinit;
   PROCEDURE graphsetup(data:responsedata;limits:responsescale;channel:integer;title:string);
   PROCEDURE plotdatalin(data:responsedata;ch:integer;limits:responsescale);
   PROCEDURE plotdatapix(data:responsedata;ch:integer;limits:responsescale);
   PROCEDURE autorespscale(data:responsedata;VAR limits:responsescale;ch:integer);
   PROCEDURE chgscale(VAR limits:responsescale);
   PROCEDURE labels(data:responsedata;channel,mark:integer;explain:string;limits:responsescale);
   PROCEDURE grafcomp(data:responsedata;limits:responsescale);
   PROCEDURE corrgraph(data:correlationdata;regdata:regressiondata;limits:corrscale;ch:integer);
   PROCEDURE autocorrscale(data:correlationdata;regdata:regressiondata;VAR limits:corrscale);
   PROCEDURE chgcorrscale(VAR limits:corrscale);
   PROCEDURE displayinit(title1,title2:string);
   PROCEDURE displayno(number,error:real);


{*************************************************************************}
                        IMPLEMENTATION
{*************************************************************************}

{*************************************************************************}
{N**   MODULE NAME       : GRAPHINIT                                    ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE         : it registers all graphics screen drivers and***}
{***                       all fonts drivers, aborts the program if a  ***}
{***                       certain driver is not present, else it      ***}
{***                       determines which graphics adaptor is needed ***}
{***                       and initializes the screen accordingly      ***}
{S**   CALL SEQUENCE     : GRAPHINIT                                    ***}
{I**   INPUT PARAMETERS  : none                                         ***}
{O**   OUTPUT PARAMETERS : none                                         ***}
{G**   GLOBAL VARIABLES  : none                                         ***}
{M**   MODULES CALLED     : Drivers, Fonts                               ***}
{E**   ERROR CONDITIONS  : If a driver is not present                   ***}
{C**   COMMENTS          : none                                         ***}
{*************************************************************************}

PROCEDURE Graphinit;

   VAR
      GraphDriver, GraphMode, Error : integer;


   PROCEDURE Abort(Msg : string);
      begin
         Writeln(Msg, ': ', GraphErrorMsg(GraphResult));
         Halt(1);
      end;


   BEGIN
      { Register all the drivers }
      if RegisterBGIdriver(@CGADriverProc) < 0 then
      Abort('CGA');
      if RegisterBGIdriver(@EGAVGADriverProc) < 0 then
      Abort('EGA/VGA');
      if RegisterBGIdriver(@HercDriverProc) < 0 then
      Abort('Herc');
      if RegisterBGIdriver(@ATTDriverProc) < 0 then
      Abort('AT&T');
      if RegisterBGIdriver(@PC3270DriverProc) < 0 then
      Abort('PC 3270');


      { Register all the fonts }
      if RegisterBGIfont(@GothicFontProc) < 0 then
      Abort('Gothic');
      if RegisterBGIfont(@SansSerifFontProc) < 0 then
      Abort('SansSerif');
      if RegisterBGIfont(@SmallFontProc) < 0 then
      Abort('Small');
      if RegisterBGIfont(@TriplexFontProc) < 0 then
      Abort('Triplex');
```

```
{*************************************************************************}
{N**   MODULE NAME        : PLOTDATALIN                                   ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : plots the response data of a real time        ***}
{***                        response graph using lines inbetween the      ***}
{***                        individual data points                        ***}
{S**   CALL SEQUENCE      : PLOTDATALIN(data record,channel,limits)       ***}
{I**   INPUT PARAMETERS   : response data record,channel to plot, limits***}
{***                        of graph                                      ***}
{O**   OUTPUT PARAMETERS  : none                                          ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Graph, Types                                  ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about data record and limits see***}
{***                        TYPES                                         ***}
{*************************************************************************}

PROCEDURE plotdatalin(data:responsedata;ch:integer;limits:responsescale);

   VAR
      k,y,t,yold,told,tmax,sampmin,sampmax:integer;

   BEGIN
      sampmin:=round(limits.tmin*data.frequency);
      sampmax:=round(limits.tmax*data.frequency);

      told:=0;
      yold:=getmaxY-60-round(((data.data[ch,sampmin]-limits.ymin)/
                              (limits.ymax-limits.ymin)*(getmaxY-70)));
      tmax:=round(getmaxX-100);

      for k:= (sampmin+1) to sampmax do
         if k<data.nsamples then
            begin
               y:=getmaxY-60-round(((data.data[ch,k]-limits.ymin)/(limits.ymax-limits.ymin)*(getmaxY-70)));
               t:=round((k-sampmin)/(sampmax-sampmin)*tmax);
               line(told,yold,t,y);
               told:=t;yold:=y;
            end;
   END;


{*************************************************************************}
{N**   MODULE NAME        : PLOTDATAPIX                                   ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : plots the response data of a real time        ***}
{***                        response graph using points to plot the       ***}
{***                        individual data points                        ***}
{S**   CALL SEQUENCE      : PLOTDATAPIX(data record,channel, limits)      ***}
{I**   INPUT PARAMETERS   : response data record, channel to plot,limits***}
{***                        of response                                   ***}
{O**   OUTPUT PARAMETERS  : none                                          ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Graph, Types                                  ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about data record and limits see***}
{***                        TYPES                                         ***}
{*************************************************************************}

PROCEDURE plotdatapix(data:responsedata;ch:integer;limits:responsescale);

   VAR
      k,l,y,t,tmax,sampmin,sampmax:integer;

   BEGIN
      sampmin:=round(limits.tmin*data.frequency);
      sampmax:=round(limits.tmax*data.frequency);
      tmax:=round(getmaxX-100);

      for k:= sampmin to sampmax do
         if k<data.nsamples then
            begin
               y:=getmaxY-60-round(((data.data[ch,k]-limits.ymin)/(limits.ymax-limits.ymin)*(getmaxY-70)));
               t:=round((k-sampmin)/(sampmax-sampmin)*tmax);
               putpixel(t,y,15);
            end;
   END;
```

```
{************************************************************************}
{N**   MODULE NAME        : AUTORESPSCALE                                 ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE          : automatically scales the vertical axis of a ***}
{***                        real time response graph and sets the limits***}
{***                        of the time axis to the extremes (i.e.no of ***}
{***                        samples)                                    ***}
{S**   CALL SEQUENCE      : AUTOSCALE(data record,limits)               ***}
{I**   INPUT PARAMETERS   : response data record                        ***}
{O**   OUTPUT PARAMETERS  : limits of response                          ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : Types                                       ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : for details about data record and limits see***}
{***                        TYPES                                       ***}
{************************************************************************}

PROCEDURE autorespscale(data:responsedata;VAR limits:responsescale;ch:integer);

   VAR
      i:integer;

   BEGIN
      limits.ymin:=5000;limits.tmin:=0;limits.ymax:=-5000;
      for i:=0 to (data.nsamples-1) do
         begin
            if limits.ymax<data.data[ch,i] then limits.ymax:=data.data[ch,i];
            if limits.ymin>data.data[ch,i] then limits.ymin:=data.data[ch,i];
         end;

      limits.tmax:=((data.nsamples-1)/data.frequency);
   END;


{************************************************************************}
{N**   MODULE NAME        : CHGSCALE                                      ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE          : changes the scales of a time response graph ***}
{***                        manually                                    ***}
{S**   CALL SEQUENCE      : CHGSCALE(limits)                            ***}
{I**   INPUT PARAMETERS   : limits of response                          ***}
{O**   OUTPUT PARAMETERS  : New limits of response                      ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : Crt, Types, plotgrad                        ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : for details about limits see TYPES          ***}
{************************************************************************}

PROCEDURE chgscale(VAR limits:responsescale);

   VAR
      opt:integer;
      screen:screenmenu;
      option:char;

   BEGIN
      INITMENU1(screen);

      repeat
         option:='c';
         clrscr;
         WRITEMENU(screen);
         gotoXY(45,11);write(limits.tmin:6:3);
         gotoXY(45,12);write(limits.tmax:6:3);
         gotoXY(45,13);write(limits.ymin:6);
         gotoXY(45,14);write(limits.ymax:6);
         gotoXY(5,20);write('Option [',option,']: ');option:=readkey;
         if option=#13 then option:='c';

         opt:=ord(option)-48;

         case option of
         '1':readreal(45,10+opt,8,limits.tmin);
         '2':readreal(45,10+opt,8,limits.tmax);
         '3':readint(45,10+opt,7,limits.ymin);
         '4':readint(45,10+opt,7,limits.ymax);
         end;

      until option='c';
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : LABELS                                        ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          : insert a label in a time response graph       ***}
{S**   CALL SEQUENCE      : LABELS(data record,channel,label,limits)      ***}
{I**   INPUT PARAMETERS   : Response data record, channel to label, no    ***}
{***                        to mark graph with, actual label for expla-   ***}
{***                        nation of the response at the bottom of the   ***}
{***                        graph, limits of response                     ***}
{O**   OUTPUT PARAMETERS  : none                                          ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Graph, Types                                  ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about data record and limits see***}
{***                        TYPES                                         ***}
{*************************************************************************}

PROCEDURE labels(data:responsedata;channel,mark:integer;explain:string;limits:responsescale);

   VAR
      i,x,y:integer;
      s:string[1];
      name:string[25];

   BEGIN
      name:='';

      if limits.tmax>(data.nsamples/data.frequency) then
         x:=8+round(((data.nsamples-1)/data.frequency-limits.tmin)/(limits.tmax-limits.tmin)*
                    round((getmaxX-100)/50)*50)
      else
         x:=15+round(((getmaxX-100)/50)*50);

      y:=round(limits.tmax*data.frequency);
      if y>data.nsamples-1 then y:=data.nsamples-1;
      y:=getmaxY-60-round(((data.data[channel,y]-limits.ymin)/(limits.ymax-limits.ymin)*(getmaxY-70)));

      str(mark:1,s);
      outtextXY(x,y,s);

      y:=getmaxY-26;
      x:=(mark-1)*round((getmaxX-50)/5);
      insert(explain,name,1);
      insert(': ',name,1);
      insert(s,name,1);
      insert('Ch ',name,1);
      outtextXY(x,y,name);
   END;


{*************************************************************************}
{N**   MODULE NAME        : GRAFCOMP                                      ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          : plots channel 1 and 2 of a response data      ***}
{***                        record on the same graph                      ***}
{S**   CALL SEQUENCE      : GRAFCOMP(data,limits)                         ***}
{I**   INPUT PARAMETERS   : Response data record, limits of response      ***}
{O**   OUTPUT PARAMETERS  : none                                          ***}
{G**   GLOBAL VARIABLES   : limits of response                            ***}
{M**   MODULES CALLED     : Crt,Plotgraf,Types                            ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about data record and limits see***}
{***                        TYPES                                         ***}
{*************************************************************************}

PROCEDURE grafcomp(data: responsedata;limits:responsescale);

   BEGIN
      GRAPHSETUP(data,limits,1,data.title);
      PLOTDATALIN(data,1,limits);
      LABELS(data,1,1,data.labels[1],limits);
      PLOTDATALIN(data,2,limits);
      LABELS(data,2,2,data.labels[2],limits);
      Repeat until keypressed;readln;
      closegraph;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : CORRGRAPH                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : plots a graph between two series of data to ***}
{***                         indicate their correlation:                  ***}
{***                         the procedure initializes the graphics       ***}
{***                         screen, plots borders and the axes of the    ***}
{***                         graph, inserts all titles and plots the two ***}
{***                         data arrays against each other               ***}
{S**   CALL SEQUENCE      : CORRGRAPH(correlation data record,regression***}
{***                         data record,limits of graph, channel to plot***}
{I**   INPUT PARAMETERS   : correlation data record, regression data     ***}
{***                         record, limits of graph (type corrscale),    ***}
{***                         channel to plot                              ***}
{O**   OUTPUT PARAMETERS  : none                                         ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED     : Graph, Types                                  ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           : for details about correlation and regression***}
{***                         data record and corrscale see TYPES          ***}
{*************************************************************************}

PROCEDURE corrgraph(data:correlationdata;regdata:regressiondata;limits:corrscale;ch:integer);

  VAR
     errorcode,graphdriver,graphmode,k,i,maxx,maxy,x0,x1,y0,y1,yup,ydn:integer;
     dmaxx,dmaxy,dx,dy,ymin,ymax,xmin,xmax,yfit,y:real;
     s:string[4];
     grandtitle:string[15];

  BEGIN
     GRAPHINIT;

     xmin:=1000000;xmax:=-1000000;

     for i:=1 to data.nsamples do
        begin
           if xmax < data.data[1,i] then xmax:=data.data[1,i];
           if xmin > data.data[1,i] then xmin:=data.data[1,i];
        end;


     ymin:=regdata.yconst[ch]+regdata.slope[ch]*xmin;
     ymax:=regdata.yconst[ch]+regdata.slope[ch]*xmax;


     maxx:=getmaxX;
     maxy:=getmaxY;
     dmaxx:=(maxx-100)/10;
     dmaxy:=(maxy-70)/10;

     grandtitle:='';
     insert(grandtitle,data.title,1);
     settextstyle(triplexfont,horizdir,1);
     settextjustify(centertext,centertext);
     outtextXY(maxx div 2,8,data.title);
     rectangle(0,0,maxx,maxy);
     rectangle(3,20,maxx-3,maxy-20);
     x1:=round(50+dmaxx*10);
     line(50,30,50,maxy-40);line(50,maxy-40,x1,maxy-40);

     settextstyle(smallfont,horizdir,4);
     settextjustify(lefttext,lefttext);
     outtextXY(60,32,'DETERMINED MASSFLOW [t/h]');
     outtextXY(maxx-170,maxy-47,'ACTUAL MASSFLOW [t/h]');

     dy:=(limits.ymax[ch]-limits.ymin[ch])/10;
     for k:=0 to 10 do
        begin
           y1:=round(maxy-40-k*dmaxy);
           line(45,y1,55,y1);
           str((limits.ymin[ch]+dy*k):8:0,s);
           outtextXY(20,y1+5,s);
        end;

     dx:=(limits.xmax[ch]-limits.xmin[ch])/10;
     for k:=0 to 10 do
        begin
           x1:=round(50+k*dmaxx);
           str(limits.xmin[ch]+k*dx:8:0,s);
           line(x1,maxy-42,x1,maxy-37);
           outtextXY(x1-8,maxy-25,s)
        end;

     setviewport(50,20,maxx,maxy,clipoff);

     x0:=round((xmin-limits.xmin[ch])/(limits.xmax[ch]-limits.xmin[ch])*(getmaxX-100));
     x1:=round((xmax-limits.xmin[ch])/(limits.xmax[ch]-limits.xmin[ch])*(getmaxX-100));

     y0:=getmaxY-60-round((ymin-limits.ymin[ch])/(limits.ymax[ch]-limits.ymin[ch])*(getmaxY-70));
     y1:=getmaxY-60-round((ymax-limits.ymin[ch])/(limits.ymax[ch]-limits.ymin[ch])*(getmaxY-70));
     line(x0,y0,x1,y1);
```

```
      settextjustify(centertext,centertext);
      for i:=1 to data.nsamples-1 do
         begin
            x1:=round((data.data[1,i]-limits.xmin[ch])/(limits.xmax[ch]-limits.xmin[ch])*(getmaxX-100));
            y1:=getmaxY-60-round((data.data[ch+1,i]-limits.ymin[ch])/(limits.ymax[ch]-limits.ymin[ch])*
                              (getmaxY-70));
            outtextXY(x1,y1,'*');

            yfit:=regdata.yconst[ch] + regdata.slope[ch]*data.data[1,i];

            y:=yfit + sqrt(regdata.C2[ch] + regdata.C1[ch]*sqr(data.data[1,i]-regdata.xmean[ch]));
            y1:=getmaxY-60-round((y-limits.ymin[ch])/(limits.ymax[ch]-limits.ymin[ch])*(getmaxY-70));
            if i>1 then line(x0,yup,x1,y1);
            yup:=y1;

            y:=yfit - sqrt(regdata.C2[ch] + regdata.C1[ch]*sqr(data.data[1,i]-regdata.xmean[ch]));
            y1:=getmaxY-60-round((y-limits.ymin[ch])/(limits.ymax[ch]-limits.ymin[ch])*(getmaxY-70));
            if i>1 then line(x0,ydn,x1,y1);
            ydn:=y1;

            x0:=x1;
         end;

      readln;
      closegraph;
   END;


{*************************************************************************}
{N**   MODULE NAME        : AUTOCORRSCALE                                 ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          : automatically scales the axes of a correla- ***}
{***                        tion graph                                    ***}
{S**   CALL SEQUENCE      : AUTOCORRSCALE(correlation data record,      ***}
{***                                       regression data record,limits)***}
{I**   INPUT PARAMETERS   : correlation data record, regression data    ***}
{***                        record                                        ***}
{O**   OUTPUT PARAMETERS  : limits of Graph (type corrscale)            ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : Types                                       ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : for details about correlation and regression***}
{***                        data record and corrscale see TYPES         ***}
{*************************************************************************}

PROCEDURE autocorrscale(data:correlationdata;regdata: regressiondata;VAR limits:corrscale);

   VAR
      i,ch:integer;
      y:real;

   BEGIN
      limits.xmin[1]:=1000000;limits.xmax[1]:=-1000000;

      for i:=1 to data.nsamples do
         begin
            if limits.xmax[1] < data.data[1,i] then limits.xmax[1]:=data.data[1,i];
            if limits.xmin[1] > data.data[1,i] then limits.xmin[1]:=data.data[1,i];
         end;

      limits.xmin[2]:=limits.xmin[1]; limits.xmax[2]:=limits.xmax[1];

      for ch:=1 to 2 do
         begin
            limits.ymin[ch]:=regdata.yconst[ch]+regdata.slope[ch]*limits.xmin[ch];
            limits.ymax[ch]:=regdata.yconst[ch]+regdata.slope[ch]*limits.xmax[ch];
         end;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME       : CHGCORRSCALE                                   ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE         : change the scales of a correlation graph       ***}
{S**   CALL SEQUENCE     : CHGCORRSCALE(limits)                           ***}
{I**   INPUT PARAMETERS  : limits of Graph                                ***}
{O**   OUTPUT PARAMETERS : New limits of Graph (type corrscale)           ***}
{G**   GLOBAL VARIABLES  : none                                           ***}
{M**   MODULES CALLED     : Crt, Types, Filehand                           ***}
{E**   ERROR CONDITIONS  : none                                           ***}
{C**   COMMENTS          : for details about corrscale see TYPES          ***}
{*************************************************************************}

PROCEDURE chgcorrscale(VAR limits:corrscale);

   VAR
      opt,i:integer;
      screen:screenmenu;
      option:char;

   BEGIN
      INITMENU2(screen);

      repeat
         option:='c';
         clrscr;
         WRITEMENU(screen);

         for i:=1 to 2 do
            begin
               gotoXY(19+i*19,12);write(limits.xmin[i]:6:3);
               gotoXY(19+i*19,13);write(limits.xmax[i]:6:3);
               gotoXY(19+i*19,14);write(limits.ymin[i]:6:3);
               gotoXY(19+i*19,15);write(limits.ymax[i]:6:3);
            end;
         gotoXY(5,20);write('Option [',option,']: ');option:=readkey;
         if option=#13 then option:='c';

         opt:=ord(option)-48;

         case option of
         '1':for i:=1 to 2 do readreal(19+i*19,11+opt,7,limits.xmin[i]);
         '2':for i:=1 to 2 do readreal(19+i*19,11+opt,7,limits.xmax[i]);
         '3':for i:=1 to 2 do readreal(19+i*19,11+opt,7,limits.ymin[i]);
         '4':for i:=1 to 2 do readreal(19+i*19,11+opt,7,limits.ymax[i]);
         end;

      until option='c';
   END;


{*************************************************************************}
{N**   MODULE NAME       : DISPALYINIT                                    ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE         : initializes the screen to display a 6 digi- ***}
{***                        tal number over the whole screen            ***}
{S**   CALL SEQUENCE     : DISPALYINIT(title1,title2)                     ***}
{I**   INPUT PARAMETERS  : titles to display at top and bottom of       ***}
{***                        screen                                      ***}
{O**   OUTPUT PARAMETERS : none                                          ***}
{G**   GLOBAL VARIABLES  : none                                          ***}
{M**   MODULES CALLED     : Graph                                        ***}
{E**   ERROR CONDITIONS  : none                                          ***}
{C**   COMMENTS          :                                              ***}
{*************************************************************************}

PROCEDURE displayinit(title1,title2:string);

VAR x,y:integer;

   BEGIN
      GRAPHINIT;
      x:=getmaxX;y:=getmaxY;

      setlinestyle(solidln,0,thickwidth);
      rectangle(x div 200,y div 25,x- x div 200,y- y div 25);
      rectangle(x div 50,trunc(y/4.5),x- x div 50,y-trunc(y/4.5));

      settextstyle(triplexfont,horizdir,5);
      settextjustify(centertext,centertext);
      outtextXY(x div 2,y div 8,title1);

      settextstyle(triplexfont,horizdir,3);
      settextjustify(righttext,centertext);
      outtextXY(x div 2,y - y div 7,title2);

      settextjustify(centertext,centertext);
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : DISPALYNO                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : displays a 6 digital number in the middle of***}
{***                         the screen and the associated error at the  ***}
{***                         bottom                                      ***}
{S**   CALL SEQUENCE      : DISPALYNO(number,error)                       ***}
{I**   INPUT PARAMETERS   : number to display and associated error       ***}
{O**   OUTPUT PARAMETERS  : none                                          ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Crt, Graph                                    ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           :                                               ***}
{*************************************************************************}

PROCEDURE displayno(number,error:real);

   VAR
      s:string[6];
      i,x,y,xnew,ynew:integer;

   BEGIN
      x:=getmaxX;y:=getmaxY;

      setviewport(x div 50 + x div 100,trunc(y/4.0),
                  x- x div 50 - x div 100,y- trunc(y/4.0),clipon);
      settextstyle(defaultfont,horizdir,12);

      xnew:=x-2*(x div 50 + x div 100);ynew:=y-2*trunc(y/4.0);
      clearviewport;
      str(number:6:1,s);
      outtextXY(xnew div 2,ynew div 2,s);

      setviewport(x div 2,y- trunc(y/4.5) + y div 50, x - x div 3,y- y div 25 - y div 50,clipon);
      settextstyle(defaultfont,horizdir,2);

      xnew:=x- x div 2  - x div 3; ynew:= trunc(y/4.5) - y div 25 - 2* (y div 50);
      clearviewport;
      str(error:5:1,s);
      outtextXY(xnew div 2,ynew div 2,s);
   END;


{*************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0       KCAD     28-02-90                                     ***}
{***                                                                     ***}
{***   UNITEND  PLOTGRAF.PAS                                             ***}
{*************************************************************************}
END.
```

C.3.5 Plotgrad.pas

```
{*************************************************************************}
{***   UNIT PROCEDURE LIBARY CONTAINING DATAFILES TO BE USED IN        ***}
{***        PLOTGRAF.PAS                                               ***}
{***   FILE: PLOTGRAD.PAS                                              ***}
{*************************************************************************}

UNIT plotgrad;

{*************************************************************************}
                        INTERFACE
{*************************************************************************}

   USES types;

   PROCEDURE initmenu1(VAR menu:screenmenu);
   PROCEDURE initmenu2(VAR menu:screenmenu);

{*************************************************************************}
                        IMPLEMENTATION
{*************************************************************************}

{*************************************************************************}
{N**   MODULE NAME        : INITMENU1                                   ***}
{***   ---------------------------------------------------------------  ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-   ***}
{***                        routine                                     ***}
{S**   CALL SEQUENCE      : INITMENU1(menu)                             ***}
{I**   INPUT PARAMETERS   : none                                        ***}
{O**   OUTPUT PARAMETERS  : menu1 (type screenmenu)                     ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : Types                                       ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : details about screenmenu in TYPES           ***}
{*************************************************************************}

PROCEDURE initmenu1(VAR menu:screenmenu);

   BEGIN
      menu[1]:='╔═══════════════════════════════════════════════════════════════════════╗';
      menu[2]:='║                                                                       ║';
      menu[3]:='║                CHANGE OF SCALES OF THE GRAPH:                         ║';
      menu[4]:='║                                                                       ║';
      menu[5]:='║                                                                       ║';
      menu[6]:='║                                                                       ║';
      menu[7]:='║                          MENU:                                        ║';
      menu[8]:='║                                                                       ║';
      menu[9]:='║                                                                       ║';
     menu[10]:='║                                                                       ║';
     menu[11]:='║                        1 - Time Minimum:                              ║';
     menu[12]:='║                        2 - Time Maximum:                              ║';
     menu[13]:='║                        3 - Y Minimum   :                              ║';
     menu[14]:='║                        4 - Y Maximum   :                              ║';
     menu[15]:='║                                                                       ║';
     menu[16]:='║                                                                       ║';
     menu[17]:='║                        c - Continue                                   ║';
     menu[18]:='║                                                                       ║';
     menu[19]:='╚═══════════════════════════════════════════════════════════════════════╝';
     menu[20]:='';
     menu[21]:='';
     menu[22]:='';
     menu[23]:='';
     menu[24]:='';
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITMENU2                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                        routine                                       ***}
{S**   CALL SEQUENCE      : INITMENU2(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : menu2  (type screenmenu)                      ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : details about screenmenu in TYPES             ***}
{*************************************************************************}

PROCEDURE initmenu2(VAR menu:screenmenu);

  BEGIN
     menu[1]:='╔═══════════════════════════════════════════════════════════════════╗';
     menu[2]:='║                                                                   ║';
     menu[3]:='║          CHANGE OF SCALES OF THE GRAPH:                           ║';
     menu[4]:='║                                                                   ║';
     menu[5]:='║                                                                   ║';
     menu[6]:='║                                                                   ║';
     menu[7]:='║                        MENU:                                      ║';
     menu[8]:='║                                                                   ║';
     menu[9]:='║                                                                   ║';
    menu[10]:='║                                  Channel1:        Channel2:       ║';
    menu[11]:='║                                  =========        =========       ║';
    menu[12]:='║            1 - X minimum:                                         ║';
    menu[13]:='║            2 - X Maximum:                                         ║';
    menu[14]:='║            3 - Y Minimum:                                         ║';
    menu[15]:='║            4 - Y Maximum:                                         ║';
    menu[16]:='║                                                                   ║';
    menu[17]:='║                                                                   ║';
    menu[18]:='║                      c - Continue                                 ║';
    menu[19]:='╚═══════════════════════════════════════════════════════════════════╝';
    menu[20]:='';
    menu[21]:='';
    menu[22]:='';
    menu[23]:='';
    menu[24]:='';
  END;

{*************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0       KCAD     28-02-90                                     ***}
{***                                                                     ***}
{***   UNITEND  PLOTGRAD.PAS                                             ***}
{*************************************************************************}
END.
```

C.3.6 DT2801_4.pas

```
{*************************************************************************}
{***   UNIT PROCEDURE LIBARY FOR DATA TRANSLATION CARD DT2801          ***}
{***   FILE: DT2801_4.PAS                                             ***}
{*************************************************************************}
{ Ref: DT2801 manual}

UNIT DT2801_4;

{*************************************************************************}
                        INTERFACE
{*************************************************************************}


   PROCEDURE adc_read(chan_no:integer;VAR value:integer);
   PROCEDURE dac_write(chan_no,value:integer);
   PROCEDURE set_up;

   CONST
      base_address = $2EC;       {DT2801 uses port $2EC & $2ED}
      comm_wait = $4;            {DT2801 commands}
      write_wait = $2;
      read_wait = $5;
      cstop = $F;
      cclear = $1;
      cerror = $2;
      cadin = $C;
      cdaout = $8;

   VAR
      command_reg, data_reg : integer;     {DT2801 control ports}


{*************************************************************************}
                        IMPLEMENTATION
{*************************************************************************}


{*************************************************************************}
{N**   MODULE NAME        :  ADC_READ                                  ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE          :  Reads the voltage from A/D channel number ***}
{***                          <chan_no>.                               ***}
{***                          Result returned in the integer variable  ***}
{***                          <value> in the range -2048 to 2047 [Counts]***}
{***                          corresponding to the voltage input       ***}
{***                          -10,000 to 9,9951 [Volt]                 ***}
{S**   CALL SEQUENCE      :  ADC_READ(chan_no;value)                   ***}
{I**   INPUT PARAMETERS   :  channel no                                 ***}
{O**   OUTPUT PARAMETERS  :  value                                      ***}
{G**   GLOBAL VARIABLES   :  none                                       ***}
{M**   MODULES CALLED     :  none                                       ***}
{E**   ERROR CONDITIONS   :  none                                       ***}
{C**   COMMENTS           :  none                                       ***}
{*************************************************************************}

PROCEDURE adc_read(chan_no:integer; VAR value:integer);

   VAR
      high,low:byte;

   BEGIN
      repeat until (port [command_reg] and comm_wait) > 0;
      port [command_reg] := cadin;
      repeat until ((port [command_reg] xor write_wait) and write_wait) > 0;
      port [data_reg] := 0;
      repeat until ((port [command_reg] xor write_wait) and write_wait) > 0;
      port [data_reg] := chan_no;
      repeat until (port [command_reg] and read_wait) > 0;
      low := port [data_reg];
      repeat until (port [command_reg] and read_wait) > 0;
      high:=port[data_reg];

      value:=high * 256 + low;
      value:=value-2048;
   END;  {adc_read}
```

```
{*************************************************************************}
{N**   MODULE NAME        :  DAC_WRITE                                    ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          :  Writes a voltage to D/A channel <chan_no>.  ***}
{***                         The value must be an integer                ***}
{***                         0=< value =<4095                            ***}
{S**   CALL SEQUENCE      :  DAC_WRITE(chan_no,value)                     ***}
{I**   INPUT PARAMETERS   :  channel no, value                            ***}
{O**   OUTPUT PARAMETERS  :  none                                         ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  none                                         ***}
{E**   ERROR CONDITIONS   :  none                                         ***}
{C**   COMMENTS           :  none                                         ***}
{*************************************************************************}

PROCEDURE dac_write(chan_no,value:integer);


  VAR
    high, low : byte;

  BEGIN
      {Shift the value}
    value:=value+2048;
      {Impose limits: 0[C]=-10,0000[V], 2048[C]=0,0000[V] & 4095[C]=9,9951[V]}
    if value<0 then value:=0;
    if value>4095 then value:=4095;
      {Find high & low byte values}
    high:=value div 256;
    low:=value - high * 256;
      {Set DT2801 for ADC output}
    repeat until (port [command_reg] and comm_wait) > 0;
    port [command_reg] := cdaout;
      {Set Channel number}
    repeat until ((port [command_reg] xor write_wait) and write_wait) > 0;
    port [data_reg] := chan_no;
      {Output low then high byte}
    repeat until ((port [command_reg] xor write_wait) and write_wait) > 0;
    port [data_reg] := low;
    repeat until ((port [command_reg] xor write_wait) and write_wait) > 0;
    port [data_reg] := high;
  END;  {dac_write}


{*************************************************************************}
{N**   MODULE NAME        :  SET_UP                                       ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          :  To intitilize the DT2801 card and sets       ***}
{***                         channels to zero                            ***}
{S**   CALL SEQUENCE      :  SET-UP                                       ***}
{I**   INPUT PARAMETERS   :  none                                         ***}
{O**   OUTPUT PARAMETERS  :  none                                         ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  none                                         ***}
{E**   ERROR CONDITIONS   :  none                                         ***}
{C**   COMMENTS           :  none                                         ***}
{*************************************************************************}

PROCEDURE set_up;

  VAR
    temp : integer;   {Discarded data}
  BEGIN
      {Set DT2801 port addresses}
    command_reg:= base_address + 1;
    data_reg:= base_address;
      {Stop DT2801 card & Clear data register}
    port [command_reg]:= cstop;
    temp:= port [data_reg];
      {Clear DT2801 card}
    repeat until (port [command_reg] and comm_wait) > 0;
    port [command_reg] := cclear;
      {Zero DAC outputs}
    dac_write(0,0);
    dac_write(1,0);
  END;  {Set_Up}


{*************************************************************************}
{H**   REVISION HISTORY:                                                  ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0         EHF       '86       Written in Turbo 3.0            ***}
{***     2.0        KCAD     24-04-89    Converted to a Turbo Pascal Unit***}
{***                                                                      ***}
{***   UNITEND  DT2801_4.PAS                                              ***}
{*************************************************************************}
END.
```

C.3.7 Detmf.pas

```
{*************************************************************************}
{***   UNIT PROCEDURE LIBRARY TO DETERMINE THE MASSFLOW ON A CONVEYOR  ***}
{***        BELT                                                       ***}
{***   FILE: DETMF.PAS                                                 ***}
{*************************************************************************}

UNIT detmf;

{*************************************************************************}
                        INTERFACE
{*************************************************************************}

   USES Crt,types,math,detmfd,filehand,datalog;

   PROCEDUREdetmassflow(Csp:constant;Ct:time;data:responsedata;mode:integer;VAR massflowdn,massflowup,
                        Vavup:real);
   FUNCTION massflownuc(Cm:massflowconst;data:responsedata):real;
   FUNCTION massbin(Cm:massflowconst;data:responsedata):real;
   PROCEDURE massflowbelt(Cm:massflowconst;VAR mfcorr:correlationdata);
   PROCEDURE mfstats(mfdata:massflowdata);
   PROCEDURE mfcali(VAR Csp:constant;setpt:real;pulselength,outch,mode:integer;Ct:time;
                    respdata:responsedata);


{*************************************************************************}
                        IMPLEMENTATION
{*************************************************************************}

{*************************************************************************}
{N**   MODULE NAME       : DETMASSFLOW                                  ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE         : determines the massflow on a horizontal      ***}
{***                       conveyor belt from the kinetic energy        ***}
{***                       required for a step-up and -down response    ***}
{S**   CALL SEQUENCE     : DETMASSFLOW(Csp,Ct,response datarecord,      ***}
{***                       mode,massflowup,massflowdn,Vavup)            ***}
{I**   INPUT PARAMETERS  : records containing constants about speed and***}
{***                       power responses and timing of the response, ***}
{***                       response data record containing the speed    ***}
{***                       and power response, operating mode           ***}
{O**   OUTPUT PARAMETERS : massflow as determined from step-up and      ***}
{***                       -down responses, average speed of belt       ***}
{G**   GLOBAL VARIABLES  : none                                         ***}
{M**   MODULES CALLED    : Types,Math                                   ***}
{E**   ERROR CONDITIONS  : none                                         ***}
{C**   COMMENTS          : - for details on the response data record    ***}
{***                         see TYPES                                  ***}
{***                       - modes: 1 - VSD, Vdn=0; 2 - VSD, Vdn>0;     ***}
{***                                3 - SSR, Vdn=0; 4 - SSR, Vdn>0      ***}
{***                       - for details about Csp and Ct see TYPES     ***}
{*************************************************************************}

PROCEDURE detmassflow(Csp:constant;Ct:time;data:responsedata;mode:integer;VAR massflowdn,massflowup,
                      Vavup:real);

  VAR
     Essup,Essdn,Edyndn,Edynup,Dssup,Dssdn,Ddyndn,Ddynup,
     V2ssup,V2dyndn,V2dynup,V3ssup,V3dyndn,V3dynup,
     F,Vavdn,Ekindn,Ekinup:real;

  BEGIN
     with Ct,Csp do
        begin
           Essup:=AREA(data,2,ssup,stepdn-ssup)*power; Essdn:=AREA(data,2,ssdn,stepup-ssdn)*power;
           Edyndn:=AREA(data,2,stepdn,ssdn-stepdn)*power; Edynup:=AREA(data,2,stepup,ssend-stepup)*power;
           Dssup:=AREA(data,1,ssup,stepdn-ssup)*speed; Dssdn:=AREA(data,1,ssdn,stepup-ssdn)*speed;
           Ddyndn:=AREA(data,1,stepdn,ssdn-stepdn)*speed; Ddynup:=AREA(data,1,stepup,ssend-stepup)*speed;
           V2ssup:=QUADAREA(data,1,ssup,stepdn-ssup)*TPO(speed,2);
           V2dyndn:=QUADAREA(data,1,stepdn,ssdn-stepdn)*TPO(speed,2);
           V2dynup:=QUADAREA(data,1,stepup,ssend-stepup)*TPO(speed,2);
           V3ssup:=CUBICAREA(data,1,ssup,stepdn-ssup)*TPO(speed,3);
           V3dyndn:=CUBICAREA(data,1,stepdn,ssdn-stepdn)*TPO(speed,3);
           V3dynup:=CUBICAREA(data,1,stepup,ssend-stepup)*TPO(speed,3);
        end;

     F:=(Essup - Csp.Fa*V2ssup - Csp.Fb*V3ssup)/Dssup;

     Vavup:=Dssup*data.frequency/(Ct.stepdn-Ct.ssup);
     if mode=4 then Vavdn:=Ddyndn*2 * data.frequency/(Ct.stepup-Ct.stepdn) - Vavup
        else Vavdn:=Dssdn*data.frequency/(Ct.stepup-Ct.ssdn);

     Ekindn:=Edyndn - (F*Ddyndn + Csp.Fa*V2dyndn + Csp.Fb*V3dyndn);
     Ekinup:=Edynup - (F*Ddynup + Csp.Fa*V2dynup + Csp.Fb*V3dynup);

     massflowdn:=2*Ekindn/(sqr(Vavdn)-sqr(Vavup))  *  Vavup*3.6/Csp.dh;
     massflowup:=2*Ekinup/(sqr(Vavup)-sqr(Vavdn))  *  Vavup*3.6/Csp.dh;
  END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : MASSFLOWNUC                                   ***}
{***   ------------------------------------------------------------      ***}
{A**   FUNCTION           : determines the massflow using the signal      ***}
{***                        from a nuclear weightometer                   ***}
{S**   CALL SEQUENCE      : MASSFLOWNUC(Cm,response datarecord)           ***}
{I**   INPUT PARAMETERS   : record containing calibration constants,      ***}
{***                        response data record containing massflow      ***}
{***                        response                                      ***}
{O**   OUTPUT PARAMETERS  : massflow as determined by weightometer        ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types,Buffer                                  ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for more information about response data and***}
{***                        and massflow constants records see TYPES      ***}
{*************************************************************************}

FUNCTION massflownuc(Cm:massflowconst;data:responsedata):real;

   BEGIN
      massflownuc:=AREA(data,3,0,data.nsamples-1)*data.frequency/(data.nsamples-1)*Cm.nuc;
   END;


{*************************************************************************}
{N**   MODULE NAME        : MASSBIN                                       ***}
{***   ------------------------------------------------------------      ***}
{A**   FUNCTION           : determines the mass in the bin from the       ***}
{***                        signal from the hopper bin                    ***}
{S**   CALL SEQUENCE      : MASSBIN(Cm,response datarecord)               ***}
{I**   INPUT PARAMETERS   : record containing mass calibration constants***}
{***                        ,record containing mass in the bin response   ***}
{O**   OUTPUT PARAMETERS  : mass in bin                                    ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types,Buffer                                  ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about response data record or     ***}
{***                        calibration constants record see TYPES        ***}
{*************************************************************************}

FUNCTION massbin(Cm:massflowconst;data:responsedata):real;

   BEGIN
      massbin:=AREA(data,3,0,data.nsamples)*Cm.mass*data.frequency/data.nsamples;
   END;


{*************************************************************************}
{N**   MODULE NAME        : MASSBELT                                      ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : determines the massflow on the belt from the***}
{***                        mass in the hopper bin                        ***}
{S**   CALL SEQUENCE      : MASSBELT(Cm,nzi,nzf,nsamp,Vav,Masscorr)       ***}
{I**   INPUT PARAMETERS   : record containing mass calibration constants***}
{***                        ,correlation data array                       ***}
{O**   OUTPUT PARAMETERS  : massflow figures as determined from the mass***}
{***                        in the bin                                    ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : none                                          ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about correlation data record or***}
{***                        massflow constants see TYPES                  ***}
{*************************************************************************}

PROCEDURE massflowbelt(Cm:massflowconst;VAR mfcorr:correlationdata);

   VAR
      massinit,massfin,massdiff,factor,Vav:real;
      i:integer;

   BEGIN
      { Determine Mass in Bin Before Tests were Run }
      massinit:=0;
      for i:=1 to mfcorr.nzero do massinit:=massinit+mfcorr.data[1,i]/mfcorr.nzero;
      for i:=1 to mfcorr.nzero do
         begin
            Vav:=mfcorr.data[4,i];
            factor:=Cm.vincl*Cm.dh/(Cm.vincl*(Cm.dh+Cm.dch1)+Vav*(Cm.di+Cm.dch2)) * Vav/Cm.dh  *  3.6;
            mfcorr.data[1,i]:=(massinit-mfcorr.data[1,i])*factor;
         end;

      { Determine Mass in Bin after Tests were Run }
      massfin:=0;
      for i:=(mfcorr.nsamples+1) to (mfcorr.nsamples+mfcorr.nzero) do
         massfin:=massfin+mfcorr.data[1,i]/mfcorr.nzero;

      massdiff:=(massinit-massfin)/(mfcorr.nsamples-mfcorr.nzero+2);

      { Update all Mass Readings in between initial and final Zero-Mass Tests)
      for i:=(mfcorr.nzero+1) to mfcorr.nsamples do
         begin
            massinit:=massinit-massdiff;
            Vav:=mfcorr.data[4,i];
            factor:=Cm.vincl*Cm.dh/(Cm.vincl*(Cm.dh+Cm.dch1)+Vav*(Cm.di+Cm.dch2))  *  Vav/Cm.dh  *  3.6;
            mfcorr.data[1,i]:=(massinit-mfcorr.data[1,i])*factor
         end;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : MFSTATS                                       ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : determines the total, average, maximum and    ***}
{***                        minimum massflow over the last sampling       ***}
{***                        period                                        ***}
{S**   CALL SEQUENCE      : MFSTATS(mfdata)                               ***}
{I**   INPUT PARAMETERS   : record containing massflow data              ***}
{O**   OUTPUT PARAMETERS  : total, average, maximum and minimum massflow***}
{***                        figures                                       ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : detmfd                                        ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about massflow data record see    ***}
{***                        TYPES                                         ***}
{*************************************************************************}

PROCEDURE mfstats(mfdata:massflowdata);

   VAR
      i,initsample,finsample,max,min:integer;
      dt:longint;
      totmass,avmassflow:real;
      menu:screenmenu;
      opt:char;

   BEGIN
      INITMENU1(menu);
      initsample:=1;finsample:=mfdata.nsamples;

      repeat
         max:=initsample;min:=initsample;
         totmass:=0;

         for i:=initsample to finsample do
            begin
               if mfdata.data[i]>mfdata.data[max] then max:=i;
               if mfdata.data[i]<mfdata.data[min] then min:=i;
               mfdata.time[i].sec100:=0;
            end;

         for i:=initsample to (finsample-1) do
            begin
               TIMEDIFF(mfdata.time[i],mfdata.time[i+1],dt);
               totmass:=totmass + (mfdata.data[i]+mfdata.data[i+1])*dt/2/1000/3600;
            end;

         TIMEDIFF(mfdata.time[initsample],mfdata.time[finsample],dt);
         avmassflow:=totmass/dt*1000*3600;

         clrscr;
         WRITEMENU(menu);
         gotoXY(31,9);write(initsample:4);
         gotoXY(39,9);write(mfdata.time[initsample].hour:2);
         gotoXY(42,9);write(mfdata.time[initsample].min:2);
         gotoXY(45,9);write(mfdata.time[initsample].sec:2);
         gotoXY(53,9);write(mfdata.data[initsample]:6:2);

         gotoXY(31,10);write(finsample:4);
         gotoXY(39,10);write(mfdata.time[finsample].hour:2);
         gotoXY(42,10);write(mfdata.time[finsample].min:2);
         gotoXY(45,10);write(mfdata.time[finsample].sec:2);
         gotoXY(53,10);write(mfdata.data[finsample]:6:2);

         gotoXY(31,14);write(max:4);
         gotoXY(39,14);write(mfdata.time[max].hour:2);
         gotoXY(42,14);write(mfdata.time[max].min:2);
         gotoXY(45,14);write(mfdata.time[max].sec:2);
         gotoXY(53,14);write(mfdata.data[max]:6:2);

         gotoXY(31,15);write(min:4);
         gotoXY(39,15);write(mfdata.time[min].hour:2);
         gotoXY(42,15);write(mfdata.time[min].min:2);
         gotoXY(45,15);write(mfdata.time[min].sec:2);
         gotoXY(53,15);write(mfdata.data[min]:6:2);

         gotoXY(33,19);write(totmass:7:3);
         gotoXY(33,20);write(avmassflow:7:3);
         gotoXY(1,25);

         opt:=readkey;
         if opt=#13 then opt:='q';

         if (opt='i') or (opt='I') then READINT(31,9,4,initsample);
         if (opt='f') or (opt='F') then READINT(31,10,4,finsample);
         if initsample>finsample then initsample:=finsample-1;
         if initsample<1 then initsample:=1;
         if finsample>mfdata.nsamples then finsample:=mfdata.nsamples;

      until (opt='q') or (opt='Q');
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : MFCALI                                        ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          : calibrate the calibration constants of speed***}
{***                        power, friction constants and massflow cali-***}
{***                        bration graph                                ***}
{S**   CALL SEQUENCE      : MFCALI(Csp)                                   ***}
{I**   INPUT PARAMETERS   : record containing massflow constants         ***}
{O**   OUTPUT PARAMETERS  : updated record of massflow constants         ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED     : detmfd;filehand,datalog                       ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           : for details about massflow constants record ***}
{***                        see TYPES                                    ***}
{***                        the massflow constants record is automati-   ***}
{***                        cally saved in the const.dat file deleting   ***}
{***                        all previous values of the record            ***}
{*************************************************************************}

PROCEDURE mfcali(VAR Csp:constant;setpt:real;pulselength,outch,mode:integer;Ct:time;
                 respdata:responsedata);

   CONST
      pass='lexi';
   VAR
      i,ntest:integer;
      Vav,dum,mf,mfact:real;
      menu:screenmenu;
      cal,opt:char;
      key:string[4];

   BEGIN
      clrscr;
      gotoXY(10,10);write('ENTER Password: ---- '); {Password = LEXI}
      gotoXY(26,10);
      for i:=1 to 4 do
         begin
            opt:=readkey;
            write('x');
            insert(opt,key,i);
         end;

      if key=pass then
         begin
            INITMENU2(menu);

            with Csp do
               repeat
                  clrscr;WRITEMENU(menu);

                  gotoXY(54,9);write(speed:10);
                  gotoXY(54,10);write(power:8:4);
                  gotoXY(54,11);write(dh:6:3);
                  gotoXY(54,14);write(Fa:8:4);
                  gotoXY(54,15);write(Fb:8:4);
                  gotoXY(54,18);write(error:8:4);
                  gotoXY(54,19);write(offset:8:4);
                  gotoXY(54,20);write(slope:8:4);
                  gotoXY(5,25);write('option?');

                  opt:=readkey;if opt=#13 then opt:='q';

                  case opt of
                     's','S':READREAL(54,9,10,speed);
                     'p','P':READREAL(54,10,8,power);
                     'l','L':READREAL(54,11,6,dh);
                     'f','F':READREAL(54,14,8,Fa);
                     'v','V':READREAL(54,15,8,Fb);
                     'e','E':READREAL(54,18,8,error);
                     'x','X':begin
                               HIGHLIGHT(0);gotoXY(15,23);
                               write(' Automatic or Manual Calibration? (A/M) ');normvideo;
                               write(' ');cal:=readkey;if cal=#13 then cal:='A';
                               write(cal);
                               if (cal='M') or (cal='m') then READREAL(54,19,8,offset)
                               else
                                  begin
                                     ntest:=3;
                                     clrscr;gotoXY(10,5);HIGHLIGHT(0);write('CALIBRATING OFFSET');
                                     normvideo;
                                     gotoXY(10,8);
                                     writeln('NOTE: - The belt must be empty ');
                                     gotoXY(10,10);write('Enter no of Tests to be done: [',ntest,']');
                                     readint(45,10,2,ntest);
                                     gotoXY(45,10);writeln(ntest:2);
                                     HIGHLIGHT(1);gotoXY(50,5);
                                     write(' Please Wait ');normvideo;
                                     offset:=0;
                                     gotoXY(10,14);
                                     writeln('Test No:    Det. Massflow:  Speed:');
```

```
                                    for i:=1 to ntest do
                                       begin
                                        PULSERESPONSE(setpt,pulselength,outch,respdata);
                                          DETMASSFLOW(Csp,Ct,respdata,mode,dum,mf,Vav);
                                          gotoXY(12,14+i);
                                          writeln( i:5,'        ',mf:10:2,'        ',Vav:5:3);
                                          offset:=offset+mf/ntest;
                                       end;
                                    gotoXY(10,25);
                                    write('Press <ENTER> to continue');readln;
                                 end;
                           end;
                     'y','Y':begin
                              HIGHLIGHT(0);gotoXY(15,23);
                              write(' Automatic or Manual Calibration? (A/M) ');normvideo;
                              write(' ');cal:=readkey;if cal=#13 then cal:='A';write(cal);
                              if (cal='M') or (cal='m') then READREAL(54,20,8,slope)
                              else
                                 begin
                                    ntest:=3;
                                    clrscr;gotoXY(10,5);HIGHLIGHT(0);
                                    write('CALIBRATING SLOPE');normvideo;
                                    gotoXY(10,8);writeln('NOTE: - The belt must be fully loaded ');
                                    gotoXY(10,10);write('Enter no of Tests to be done: [',ntest,']');
                                    readint(45,10,2,ntest);
                                    gotoXY(45,10);writeln(ntest:2);
                                    slope:=0;
                                    for i:=1 to ntest do
                                       begin
                                          clrscr;gotoXY(30,10);HIGHLIGHT(1);
                                          write(' Please Wait ');normvideo;
                                          PULSERESPONSE(setpt,pulselength,outch,respdata);
                                          DETMASSFLOW(Csp,Ct,respdata,mode,dum,mf,Vav);
                                          clrscr;gotoXY(30,5);writeln('Test No  ',i);
                                          gotoXY(5,10);writeln('Det. Massflow : ',mf:7:2,
                                                               '          Vav : ',Vav:5:3);
                                          gotoXY(5,13);
                                          write('Enter actual unitmass for previous test [kg/m] : ');
                                          readreal(55,13,7,mfact);
                                          gotoXY(10,20);
                                          write('Press <ENTER> to continue');readln;
                                          slope:=slope+(mf-offset)/(mfact*Vav*3.6)/ntest;
                                       end;
                                 end;
                           end;
                     end;
                  until (opt='q') or (opt='Q');
                  SAVECONST(Csp);
               end
            else
               begin
                  sound(1000);delay(1000);nosound;
               end;
      END;

{************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0       KCAD     07-04-90                                     ***}
{***                                                                     ***}
{***   UNITEND DETMF.PAS                                                 ***}
{************************************************************************}
END.
```

C.3.8 Detmfd.pas

```
{*************************************************************************}
{***   UNIT PROCEDURE LIBARY CONTAINING DATAFILES TO BE USED IN        ***}
{***        DETMF.PAS                                                  ***}
{***   FILE: DETMFD.PAS                                                ***}
{*************************************************************************}

UNIT detmfd;

{*************************************************************************}
                        INTERFACE
{*************************************************************************}

   USES types;

   PROCEDURE initmenu1(VAR menu:screenmenu);
   PROCEDURE initmenu2(VAR menu:screenmenu);

{*************************************************************************}
                        IMPLEMENTATION
{*************************************************************************}

{*************************************************************************}
{N**   MODULE NAME       : INITMENU1                                    ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE         : initializes the menu defined in this sub-    ***}
{***                       routine                                      ***}
{S**   CALL SEQUENCE     : INITMENU1(menu)                              ***}
{I**   INPUT PARAMETERS  : none                                         ***}
{O**   OUTPUT PARAMETERS : menu1 (type screenmenu)                      ***}
{G**   GLOBAL VARIABLES  : none                                         ***}
{M**   MODULES CALLED    : Types                                        ***}
{E**   ERROR CONDITIONS  : none                                         ***}
{C**   COMMENTS          : details about screenmenu in TYPES            ***}
{*************************************************************************}

PROCEDURE initmenu1(VAR menu:screenmenu);

   BEGIN
      menu[1]:='╔═══════════════════════════════════════════════════════════════════════╗';
      menu[2]:='║                                                                       ║';
      menu[3]:='║                    MASSFLOW STATISTICS:                               ║';
      menu[4]:='║                                                                       ║';
      menu[5]:='║                                                                       ║';
      menu[6]:='║          ┌──────────────────┬──────┬──────────┬──────────────┐         ║';
      menu[7]:='║          │                  │  no  │   Time   │ inst. Massflow │      ║';
      menu[8]:='║          ├──────────────────┼──────┼──────────┼──────────────┤         ║';
      menu[9]:='║          │ Initial Sample:  │      │   :  :   │          t/hr  │      ║';
     menu[10]:='║          │   Final Sample:  │      │   :  :   │          t/hr  │      ║';
     menu[11]:='║          └──────────────────┴──────┴──────────┴──────────────┘         ║';
     menu[12]:='║                                                                       ║';
     menu[13]:='║          ┌──────────────────┬──────┬──────────┬──────────────┐         ║';
     menu[14]:='║          │        Maximum:  │      │   :  :   │          t/hr  │      ║';
     menu[15]:='║          │        Minimum:  │      │   :  :   │          t/hr  │      ║';
     menu[16]:='║          └──────────────────┴──────┴──────────┴──────────────┘         ║';
     menu[17]:='║                                                                       ║';
     menu[18]:='║          ┌──────────────────┬──────────┐                              ║';
     menu[19]:='║          │       Total Mass:│     tons │                              ║';
     menu[20]:='║          │ Average Massflow:│      t/hr│                              ║';
     menu[21]:='║          └──────────────────┴──────────┘                              ║';
     menu[22]:='╠═══════════════════════════════════════════════════════════════════════╣';
     menu[23]:='║ Options: chg: i - initial sample, f - final sample;      q - quit     ║';
     menu[24]:='╚═══════════════════════════════════════════════════════════════════════╝';
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITMENU2                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                        routine                                       ***}
{S**   CALL SEQUENCE      : INITMENU2(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : menu1 (type screenmenu)                       ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : details about screenmenu in TYPES             ***}
{*************************************************************************}

PROCEDURE initmenu2(VAR menu:screenmenu);

  BEGIN
    menu[1]:='╔═══════════════════════════════════════════════════════════════════╗';
    menu[2]:='║                                                                   ║';
    menu[3]:='║          MASSFLOW CONSTANTS CALIBRATION ROUTINE:                  ║';
    menu[4]:='║                                                                   ║';
    menu[5]:='║                                                                   ║';
    menu[6]:='║     ┌─────┬──────────────┬──────────────────┬──────────────┐       ║';
    menu[7]:='║     │ Key │ Constants:   │      units:      │    Value:    │       ║';
    menu[8]:='║     ├─────┼──────────────┼──────────────────┼──────────────┤       ║';
    menu[9]:='║     │  s  │    Speed     │ m/sec / Pc count │              │       ║';
   menu[10]:='║     │  p  │    Power     │ W / Pc count     │              │       ║';
   menu[11]:='║     │  l  │ Belt Length  │ meters           │              │       ║';
   menu[12]:='║     └─────┴──────────────┴──────────────────┴──────────────┘       ║';
   menu[13]:='║     ┌─────┬──────────────┬──────────────────┬──────────────┐       ║';
   menu[14]:='║     │  f  │    Force1    │ Newton/m^2/sec   │              │       ║';
   menu[15]:='║     │  v  │    Force2    │ Newton/m^3/sec^2 │              │       ║';
   menu[16]:='║     └─────┴──────────────┴──────────────────┴──────────────┘       ║';
   menu[17]:='║     ┌─────┬──────────────┬──────────────────┬──────────────┐       ║';
   menu[18]:='║     │  e  │    Error     │ tons/hour        │              │       ║';
   menu[19]:='║     │  x  │    Offset    │ tons/hour        │              │       ║';
   menu[20]:='║     │  y  │    Slope     │ t/hr / t/hr      │              │       ║';
   menu[21]:='║     └─────┴──────────────┴──────────────────┴──────────────┘       ║';
   menu[22]:='║          NOTE:  Calibrate Offset before calibrating Slope         ║';
   menu[23]:='║                              q  -  Quit                           ║';
   menu[24]:='╚═══════════════════════════════════════════════════════════════════╝';
  END;

{*************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0        KCAD     28-02-90                                    ***}
{***                                                                     ***}
{***   UNITEND  PLOTGRAD.PAS                                             ***}
{*************************************************************************}
END.
```

C.3.9 Datalog.pas

```
{************************************************************************}
{*** UNIT CATALOG LIBRARY FOR DATALOGGING PROCEDURES                  ***}
{*** FILE: datalog.PAS                                               ***}
{************************************************************************}

UNIT datalog;

{************************************************************************}
                              INTERFACE
{************************************************************************}

   USES dt2801_4,types,filehand,crt,dos;

   PROCEDURE step(VAR data:responsedata;counts:real;nloop:longint;ch:integer);
   PROCEDURE stepresponse(setpt:real;ch:integer;VAR data:responsedata);
   PROCEDURE chksteptiming(setpt:real;ch:integer;data:responsedata);
   PROCEDURE pulse(VAR data:responsedata;pulse1,pulse2:real;nloop:longint;ch,pulselength:integer);
   PROCEDURE pulseresponse(VAR setpt:real;pulselength,ch:integer;VAR data:responsedata);
   PROCEDURE chkpulsetiming(setpt:real;pulselength,ch:integer;data:responsedata);
   PROCEDURE rndom(VAR data:responsedata;pulse1,pulse2:real;nloop:longint;ch,pulselength:integer);
   PROCEDURE randomresponse(setpt:real;pulselength,ch:integer;VAR data:responsedata);
   PROCEDURE chkrandomtiming(setpt:real;pulselength,ch:integer;data:responsedata);


{************************************************************************}
                            IMPLEMENTATION
{************************************************************************}


{************************************************************************}
{N**   MODULE NAME       :  STEPRESPONSE                                ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE         :  uses the DT2801 I/O card to step a          ***}
{***                        specified output channel and Loggs up to 6 ***}
{***                        input channels                             ***}
{S**   CALL SEQUENCE     :  STEPRESPONSE(setpoint,output channel,data   ***}
{***                        record)                                    ***}
{I**   INPUT PARAMETERS  :  setpoint before step [V], channel to step,  ***}
{***                        data record (see comments)                 ***}
{O**   OUTPUT PARAMETERS :  data record containing response            ***}
{G**   GLOBAL VARIABLES  :  none                                       ***}
{M**   MODULES CALLED    :  DT2081_4,Crt,Types                          ***}
{E**   ERROR CONDITIONS  :  if timing has not yet been calibrated      ***}
{C**   COMMENTS          :  details about data record: Responsedata is ***}
{***                        a data record containing information about ***}
{***                        a response in terms of: no of samples      ***}
{***                        , sampling frequency, no of channels to    ***}
{***                        log, size of step, title of response,      ***}
{***                        legends for input channels and the actual  ***}
{***                        response (for details see TYPES)           ***}
{***                        Note that this procedure actually consists ***}
{***                        of two procedure. The 'step' procedure is  ***}
{***                        the skeleton data aquasition procedure     ***}
{************************************************************************}

PROCEDURE step(VAR data:responsedata;counts:real;nloop:longint;ch:integer);

   VAR
      i,j,dum:integer;
      k:longint;

   BEGIN
      for i:=0 to data.nsamples-1 do
         begin
            if i=data.whenstep then
               DAC_WRITE(ch,round(counts)); {SEND A STEP TO THE OUTPUT CHANNEL}
            for j:=1 to data.nochannel do
               ADC_READ(j,data.data[j,i]);  {READ A SAMPLE FROM THE INPUT CHANNEL}
            for k:=1 to nloop do dum:=1;    {WAIT LOOP}
         end;
   END;

{************************************************************************}
```

```
PROCEDURE stepresponse(setpt:real;ch:integer;VAR data:responsedata);

   VAR
      i,j,dum:integer;
      nloop,k:longint;
      counts:real;


   BEGIN
      LOADLOOPDATA(nloop); {Load no of waitloops required to put sampling frequency right}
      counts:=(setpt+data.step)*2048/10.0;

      if nloop=0 then
         begin
            sound(2000);delay(100);nosound;clrscr;HIGHLIGHT(1);
            gotoXY(15,10);writeln('TIMING OF DATALOG NOT YET CALIBRATED !!!');
            gotoXY(15,13);writeln('Press any key to perform calibration');
            normvideo;readln;
            CHKSTEPTIMING(setpt,ch,data);
            LOADLOOPDATA(nloop);
         end;

      step(data,counts,nloop,ch);
   END;


{*************************************************************************}
{N**   MODULE NAME        :  CHKSTEPTIMING                                ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          :  checks the timing of the step datalogging    ***}
{***                         routine (see STEPRESPONSE)                    ***}
{S**   CALL SEQUENCE      :  CHKSTEPTIMING(setpoint,output channel,data   ***}
{***                         record)                                       ***}
{I**   INPUT PARAMETERS   :  setpoint before step, channel to step, data***}
{***                         record                                        ***}
{O**   OUTPUT PARAMETERS  :  no of wait loops to be executed, stored on   ***}
{***                         disc                                          ***}
{G**   GLOBAL VARIABLES   :  none                                          ***}
{M**   MODULES CALLED     :  DT2081_4,Types,Dos                           ***}
{E**   ERROR CONDITIONS   :  none                                          ***}
{C**   COMMENTS           :  for details about the data record see STEP-***}
{***                         RESPONSE                                      ***}
{*************************************************************************}

PROCEDURE chksteptiming(setpt:real;ch:integer;data:responsedata);

   VAR
      l,waitloops:integer;
      dt,nloop:longint;
      t1,t2:dostime;
      counts,reset,tsamp,tact:real;
      ok:char;

   BEGIN
      clrscr;HIGHLIGHT(0);
      gotoXY(20,5);writeln('TIMING CALIBRATION OF DATALOG');
      normvideo;
      gotoXY(10,8);writeln('    treq[msec]:    tact[msec]:        noloop:');

      LOADLOOPDATA(nloop);
      waitloops:=nloop;
      tsamp:=1000/data.frequency;
      counts:=(setpt+data.step)*2048/10.0;
      reset:=setpt*2048/10.0;
      l:=8;

      repeat
         l:=l+1;
         HIGHLIGHT(1);gotoXY(10,24);
         write('              Please Wait - Performing Test            ');normvideo;

         gettime(t1.hour,t1.min,t1.sec,t1.sec100);
         step(data,counts,nloop,ch);
         gettime(t2.hour,t2.min,t2.sec,t2.sec100);

         DAC_WRITE(ch,round(reset)); {RESET OUTPUT CHANNEL TO SETPOINT}

         TIMEDIFF(t1,t2,dt);
         tact:=dt/(data.nsamples-1);

         gotoXY(10,1);write('     ',tsamp:9:3,'      ',tact:9:3,'      ',nloop:9);
         gotoXY(10,24);write('            Ok? [N]                                  ');
         gotoXY(32,24);ok:=readkey;if ok=#13 then ok:='N';write(ok);

         if (ok='N') or (ok='n') then
            begin
               gotoXY(10,24);write('ENTER No. of Waitstates to be included: ');
               readint(50,24,5,waitloops);nloop:=waitloops;
            end;

      until (ok='y') or (ok='Y');

      SAVELOOPDATA(nloop);
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        :  PULSERESPONSE                                ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          :  uses the DT2801 I/O card to pulse a speci- ***}
{***                          fied output channel and loggs up to 6 input***}
{***                          channels                                    ***}
{S**   CALL SEQUENCE      :  PULSERESPONSE(setpoint,pulselength,output  ***}
{***                          channel, data record)                       ***}
{I**   INPUT PARAMETERS   :  setpoint before pulse [V], pulselength,     ***}
{***                          channel to pulse, data record               ***}
{O**   OUTPUT PARAMETERS  :  data record containing the response          ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  DT2081_4,Crt,Types                          ***}
{E**   ERROR CONDITIONS   :  if timing has not yet been calibrated       ***}
{C**   COMMENTS           :  for details about the data record see STEP-***}
{***                          RESPONSE                                    ***}
{***                          Note that this procedure actually consists ***}
{***                          of two procedure. The 'pulse' procedure is ***}
{***                          the skeleton data aquasition procedure      ***}
{*************************************************************************}

PROCEDURE pulse(VAR data:responsedata;pulse1,pulse2:real;nloop:longint;ch,pulselength:integer);

   VAR
      i,j,dum:integer;
      k:longint;

   BEGIN
      for i:=0 to data.nsamples-1 do
         begin
            if i=data.whenstep then DAC_WRITE(ch,round(pulse1)); {SEND A PULSE TO THE OUTPUT CHANNEL}
            if i=data.whenstep+pulselength then DAC_WRITE(ch,round(pulse2)); {RESET THE PULSE TO OUTPUT CH.}
            for j:=1 to data.nochannel do ADC_READ(j,data.data[j,i]);  {READ A SAMPLE FROM THE INPUT CH.}
            for k:=1 to nloop do dum:=1; {WAIT LOOP}
         end;
   END;

{*************************************************************************}

PROCEDURE pulseresponse(VAR setpt:real;pulselength,ch:integer;VAR data:responsedata);

   VAR
      nloop:longint;
      pulse1,pulse2:real;

   BEGIN
      LOADLOOPDATA(nloop);
      pulse1:=(setpt+data.step)*2048/10.0;
      pulse2:=setpt*2048/10.0;

      if nloop=0 then
         begin
            sound(2000);delay(100);nosound;clrscr;HIGHLIGHT(1);
            gotoXY(15,10);writeln('TIMING OF DATALOG NOT YET CALIBRATED !!!');
            gotoXY(15,13);writeln('Press any key to perform calibration');
            normvideo;readln;
            CHKPULSETIMING(setpt,pulselength,ch,data);
            LOADLOOPDATA(nloop);
         end;

         pulse(data,pulse1,pulse2,nloop,ch,pulselength);

         if data.nsamples<=data.whenstep+pulselength then setpt:=setpt+data.step;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        :  CHKPULSETIMING                               ***}
{***   ------------------------------------------------------------------  ***}
{A**   PROCEDURE          :  checks the timing of the pulse datalogging ***}
{***                         routine (see PULSERESPONSE)                   ***}
{S**   CALL SEQUENCE      :  CHKPULSETIMING(setpoint,pulselength,output ***}
{***                         channel, data record)                         ***}
{I**   INPUT PARAMETERS   :  setpoint before pulse, pulselength, channel***}
{***                         to pulse, data record                         ***}
{O**   OUTPUT PARAMETERS  :  no of wait loops to be executed, stored on ***}
{***                         disc                                          ***}
{G**   GLOBAL VARIABLES   :  none                                          ***}
{M**   MODULES CALLED     :  DT2081_4,Types,Dos                           ***}
{E**   ERROR CONDITIONS   :  none                                          ***}
{C**   COMMENTS           :  for details about the data record see STEP-***}
{***                         RESPONSE                                      ***}
{*************************************************************************}

PROCEDURE chkpulsetiming(setpt:real;pulselength,ch:integer;data:responsedata);

  VAR
    l,waitloops:integer;
    dt,nloop:longint;
    t1,t2:dostime;
    pulse1,pulse2,tsamp,tact:real;
    ok:char;

  BEGIN
    clrscr;HIGHLIGHT(0);
    gotoXY(20,5);writeln('TIMING CALIBRATION OF DATALOG');
    normvideo;
    gotoXY(10,8);writeln('    treq[msec]:    tact[msec]:        noloop:');

    LOADLOOPDATA(nloop);
    waitloops:=nloop;
    tsamp:=1000/data.frequency;
    pulse1:=(setpt+data.step)*2048/10.0;
    pulse2:=setpt*2048/10.0;
    l:=8;

    repeat
      l:=l+1;
      HIGHLIGHT(1);gotoXY(10,24);
      write('          -       Please Wait - Performing Test            ');normvideo;

      gettime(t1.hour,t1.min,t1.sec,t1.sec100);
      pulse(data,pulse1,pulse2,nloop,ch,pulselength);
      gettime(t2.hour,t2.min,t2.sec,t2.sec100);

      TIMEDIFF(t1,t2,dt);
      tact:=dt/(data.nsamples-1);

      gotoXY(10,1);write('     ',tsamp:9:3,'      ',tact:9:3,'     ',nloop:9);
      gotoXY(10,24);write('            OK? [N]                                          ');
      gotoXY(32,24);ok:=readkey;if ok=#13 then ok:='N';write(ok);

      if (ok='N') or (ok='n') then
        begin
          gotoXY(10,24);write('ENTER No. of Waitstates to be included: ');
          readint(50,24,5,waitloops);nloop:=waitloops;
        end;

    until (ok='y') or (ok='Y');

    SAVELOOPDATA(nloop);
  END;
```

```
{*************************************************************************}
{N**   MODULE NAME        :  PSEUDO RANDOM SEQUENCE RESPONSE             ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          :  uses the DT2801 I/O card to send pulsedata ***}
{***                          to a specified channel in a random fashion.***}
{***                          The resulting response can be used in Auto ***}
{***                          Correlation Techniques. Up to 6 input      ***}
{***                          channels can be logged                     ***}
{S**   CALL SEQUENCE      :  RANDOMRESPONSE(setpoint,pulselength,output ***}
{***                          channel, data record)                      ***}
{I**   INPUT PARAMETERS   :  setpoint before test [V], pulse length,     ***}
{***                          channel to pulse, data record              ***}
{O**   OUTPUT PARAMETERS  :  data record containing response             ***}
{G**   GLOBAL VARIABLES   :  none                                        ***}
{M**   MODULES CALLED     :  DT2081_4,Crt,Types                          ***}
{E**   ERROR CONDITIONS   :  if timing has not yet been calibrated       ***}
{C**   COMMENTS           :  1) for details about the data record see    ***}
{***                             STEPRESPONSE                            ***}
{***                          2) the no of samples may have to be in-    ***}
{***                             creased in TYPES (from 500 -> 5000)     ***}
{***                          Note that this procedure actually consists ***}
{***                          of two procedure. The 'rndom' procedure    ***}
{***                          is the skeleton data aquasition procedure  ***}
{*************************************************************************}

PROCEDURE rndom(VAR data:responsedata;pulse1,pulse2:real;nloop:longint;ch,pulselength:integer);

   VAR
      i,j,c,dum:integer;
      k:longint;
      x:real;

   BEGIN
      c:=0;
      randomize;

      for i:=0 to data.nsamples-1 do
         begin
            c:=c+1;
            if c=pulselength then
               begin
                  x:=random;
                  if x>0.5 then DAC_WRITE(ch,round(pulse1)) {SEND A  PULSE TO OUTPUT CHANNEL}
                  else DAC_WRITE(ch,round(pulse2)); {RESET OUTPUT CHANNEL TO DEFAULT VALUE}
                  c:=0;
               end;

            for j:=1 to data.nochannel do ADC_READ(j,data.data[j,i]);  {READ A SAMPLE FROM INPUT CHANNEL}
            for k:=1 to nloop do dum:=1;   {WAIT LOOP}
         end;
   END;

{*************************************************************************}

PROCEDURE randomresponse(setpt:real;pulselength,ch:integer;VAR data:responsedata);

   VAR
      nloop:longint;
      pulse1,pulse2:real;

   BEGIN
      LOADLOOPDATA(nloop);
      pulse1:=(setpt+data.step)*2048/10.0;
      pulse2:=setpt*2048/10.0;

     if nloop=0 then
         begin
            sound(2000);delay(100);nosound;clrscr;HIGHLIGHT(1);
            gotoXY(15,10);writeln('TIMING OF DATALOG NOT YET CALIBRATED !!!');
            gotoXY(15,13);writeln('Press any key to perform calibration');
            normvideo;readln;
            CHKRANDOMTIMING(setpt,pulselength,ch,data);
            LOADLOOPDATA(nloop);
         end;

      rndom(data,pulse1,pulse2,nloop,ch,pulselength);

      DAC_WRITE(ch,round(pulse2)); {RESET OUTPUT CHANNEL TO DEFAULT VALUE}
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        :  CHKRANDOMTIMING                              ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          :  checks the timing of the Random Datalogging***}
{***                         routine (see RANDOMRESPONSE)                 ***}
{S**   CALL SEQUENCE      :  CHKRANDOMTIMING(setpoint,pulselength,output***}
{***                         channel, data record)                        ***}
{I**   INPUT PARAMETERS   :  setpoint before pulse, pulselength, channel***}
{***                         to step, data record                         ***}
{O**   OUTPUT PARAMETERS  :  no of wait loops to be executed, stored on ***}
{***                         disc                                         ***}
{G**   GLOBAL VARIABLES   :  none                                         ***}
{M**   MODULES CALLED     :  DT2081_4,Types,Dos                           ***}
{E**   ERROR CONDITIONS   :  none                                         ***}
{C**   COMMENTS           :  for details about the data record see STEP-***}
{***                         RESPONSE                                     ***}
{*************************************************************************}

PROCEDURE chkrandomtiming(setpt:real;pulselength,ch:integer;data:responsedata);

  VAR
    l,waitloops:integer;
    dt,nloop:longint;
    t1,t2:dostime;
    pulse1,pulse2,tsamp,tact:real;
    ok:char;

  BEGIN
    clrscr;HIGHLIGHT(0);
    gotoXY(20,5);writeln('TIMING CALIBRATION OF DATALOG');
    normvideo;
    gotoXY(10,8);writeln('    treq[msec]:    tact[msec]:        noloop:');

    LOADLOOPDATA(nloop);
    waitloops:=nloop;
    tsamp:=1000/data.frequency;
    pulse1:=(setpt+data.step)*2048/10.0;
    pulse2:=setpt*2048/10.0;
    l:=8;

    repeat
      l:=l+1;
      HIGHLIGHT(1);gotoXY(10,24);
      write('              Please Wait - Performing Test            ');normvideo;

      gettime(t1.hour,t1.min,t1.sec,t1.sec100);
      rndom(data,pulse1,pulse2,nloop,ch,pulselength);
      gettime(t2.hour,t2.min,t2.sec,t2.sec100);

      DAC_WRITE(ch,round(pulse2)); {RESET OUTPUT CHANNEL TO DEFAULT VALUE}

      TIMEDIFF(t1,t2,dt);
      tact:=dt/(data.nsamples-1);

      gotoXY(10,1);write('     ',tsamp:9:3,'      ',tact:9:3,'      ',nloop:9);
      gotoXY(10,24);write('             Ok? [N]                                          ');
      gotoXY(32,24);ok:=readkey;if ok=#13 then ok:='N';write(ok);

      if (ok='N') or (ok='n') then
        begin
          gotoXY(10,24);write('ENTER No. of Waitstates to be included: ');
          readint(50,10,5,waitloops);nloop:=waitloops;
        end;

    until (ok='y') or (ok='Y');

    SAVELOOPDATA(nloop);
  END;

{*************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENT                        ***}
{***     1.0       KCAD     26-02-90                                     ***}
{***                                                                     ***}
{***   UNITEND DATALOG.PAS                                               ***}
{*************************************************************************}
END.
```

C.3.10 Dlog.pas

```
{*************************************************************************}
{***   PROGRAM PERFORMS A STEPTEST AND RECORDS THE RESPONSE            ***}
{***   FILE: DLOG.PAS                                                  ***}
{*************************************************************************}

{*************************************************************************}
{N**   PROGRAM NAME      : DLOG                                         ***}
{***   ------------------------------------------------------------   ***}
{A**   DESCRIPTION       : this program performs a step or pulse test  ***}
{***                       and records the response of up to six input ***}
{***                       channels and stores the data on a file      ***}
{S**   CALL SEQUENCE     : MAIN PROGRAM                                 ***}
{I**   INPUT PARAMETERS  : sampling frequency, no of samples, size of  ***}
{***                       step or pulse, step up or down, filename,   ***}
{***                       title of response, sample at which to step  ***}
{***                       the input, length of pulse, operation mode  ***}
{***                       (i.e. step or pulse or random)              ***}
{O**   OUTPUT PARAMETERS : response an a file                          ***}
{G**   GLOBAL VARIABLES  : none                                        ***}
{M**   MODULES CALLED    : Graph,Crt,Filehand,Datalog,Dt2801_4,Types,  ***}
{***                       Plotgraf,Dlogd                              ***}
{E**   ERROR CONDITIONS  : none                                        ***}
{C**   COMMENTS          : none                                        ***}
{*************************************************************************}

PROGRAM dlog;

{$M 65000,0,655360}

   USES
      graph,crt,filehand,datalog,dt2801_4,Types,plotgraf,dlogd;

   CONST
      pathdata='data\';

   VAR
      i,pos,pulselength,outputch:integer;
      data:responsedata;
      setpt,t1,t2,t3:real;
      name:str50;
      mode:string[6];
      step:string[4];
      new,save,option:char;
      menu1:screenmenu;
      limits:responsescale;

{*************************************************************************}
{***                        PROCEDURES                                 ***}
{*************************************************************************}

{-------------------------------------------------------------------------}
{***  write main menu on screen                                        ***}
{-------------------------------------------------------------------------}

PROCEDURE initmainmenu;

   BEGIN
      clrscr;
      WRITEMENU(menu1);
      if data.step>0 then step:='UP' else step:='DOWN';

      gotoXY(63,12);write(data.nsamples:4);
      gotoXY(64,13);write(data.frequency:5:1);

      gotoXY(49,15);write(mode);
      gotoXY(50,16);write(mode);gotoXY(63,16);write(data.step:5:2);
      gotoXY(42,18);write(mode);gotoXY(63,18);write(data.whenstep:4);

      if mode='STEP' then
         begin
            gotoXY(8,14);write('t - toggle Step [',step,']');
         end;

      if mode='PULSE' then
         begin
            gotoXY(42,19);write('Length of Pulse    : ',pulselength:4);
         end;

      if mode='RANDOM' then
         begin
            gotoXY(50,16);write('PULSE ');
            gotoXY(42,18);write('Length of Pulse    : ',pulselength:4);
         end;

      gotoXY(5,24);write('Option: ');
   END;
```

```
{------------------------------------------------------------------------}
{***  Save Response                                                   ***}
{------------------------------------------------------------------------}

PROCEDURE saveresponse;

   BEGIN
      data.title:=CHGTITLE('TITLE',data.title);
      FILENAME(name,'.dat');
      SAVERESPONSEDATA(name,data);
      save:='y';
   END;

{------------------------------------------------------------------------}
{***  Perform Steptest                                                ***}
{------------------------------------------------------------------------}

PROCEDURE doresponse;

   BEGIN
      clrscr;
      if save='n' then
         begin
            gotoXY(20,10);write('SAVE PREVIOUS RESPONSE ? [n]');
            if readkey<>#13 then saveresponse;
         end;

      clrscr;
      HIGHLIGHT(1);
      gotoXY(25,10);writeln('PERFORMING TEST - PLEASE WAIT');
      new:='n';

      if mode='STEP' then
         begin
            STEPRESPONSE(setpt,outputch,data);
            setpt:=setpt+data.step;
            data.step:=data.step*-1;
         end
      else if mode='PULSE' then PULSERESPONSE(setpt,pulselength,outputch,data)
      else if mode='RANDOM' then RANDOMRESPONSE(setpt,pulselength,outputch,data);

      normvideo;
      save:='n';
   END;


{------------------------------------------------------------------------}
{***  Change Fixed Variables                                          ***}
{------------------------------------------------------------------------}

PROCEDURE chgfixvar;

   VAR
      opt:char;
      y:integer;
      clear:string[10];
      menu:screenmenu;

   BEGIN
      INITMENU2(menu);
      clear:='          ';

      repeat
         clrscr;
         WRITEMENU(menu);

         gotoXY(28,6);write(mode);
         gotoXY(49,7);writeln(data.nsamples:4);
         gotoXY(49,8);writeln(data.frequency:6:1);
         gotoXY(51,9);writeln(setpt:5:2);
         gotoXY(30,10);write(mode);gotoXY(51,10);write(data.step:5:2);
         gotoXY(22,11);write(mode);gotoXY(49,11);write(data.whenstep:4);
         gotoXY(40,14);write(mode);gotoXY(51,14);write(outputch);

         if mode='PULSE' then
            begin
               gotoXY(18,12);write('p - Length of Pulse          : ',pulselength:4);
            end;

         if mode='RANDOM' then
            begin
               gotoXY(30,10);write('PULSE ');
               gotoXY(40,14);write('PULSE ');
               gotoXY(18,11);write('p - Length of Pulse          : ',pulselength:4);
            end;

         gotoXY(51,15);write(data.nochannel);
         for i:=1 to data.nochannel do
            begin
               gotoXY(37,15+i);write('Channel ',i,' :   ',data.labels[i]);
            end;

         gotoXY(5,24);write('Option [q]: ');opt:=readkey;
         if opt=#13 then opt:='q';
```

```
        case opt of
           'C','c':begin
                     READINT(51,15,1,data.nochannel);
                     SAVELOOPDATA(0);
                     if data.nochannel>6 then
                        begin
                           data.nochannel:=6;
                           sound(2000);delay(100);nosound;
                        end;
                   end;
           'F','f':begin
                     READREAL(49,8,5,data.frequency);
                     SAVELOOPDATA(0);
                   end;
           'L','l':for i:=1 to data.nochannel do
                     begin
                        pos:=POSITION(data.labels[i]);
                        data.labels[i]:=CHGSTRING(data.labels[i],clear,pos,51,14+i);
                        normvideo;
                        gotoXY(51,15+i);write(clear);
                        gotoXY(51,15+i);write(data.labels[i]);
                     end;
           'M','m':begin
                     if mode='STEP' then mode:='PULSE'
                     else if mode='PULSE' then mode:='RANDOM'
                     else mode:='STEP';
                     SAVELOOPDATA(0);
                   end;
           'N','n':READINT(49,7,4,data.nsamples);
           'O','o':begin
                     READINT(51,14,1,outputch);
                     if (outputch>1) or (outputch<0) then
                        begin
                           outputch:=1;
                           sound(2000);delay(100);nosound;
                        end;
                   end;
           'P','p':if mode='PULSE' then READINT(49,12,4,pulselength)
                   else if mode='RANDOM' then READINT(49,11,4,pulselength);
           'E','e':begin
                     READREAL(51,9,5,setpt);
                     DAC_WRITE(outputch,round(setpt*2048/10));
                   end;
           'S','s':READREAL(51,10,5,data.step);
           'W','w':if mode<>'RANDOM' then READINT(49,11,4,data.whenstep);
           'T','t':if mode='STEP' then CHKSTEPTIMING(setpt,outputch,data)
                   else if mode='PULSE' then
                        CHKPULSETIMING(setpt,pulselength,outputch,data);
                   else if mode='RANDOM' then
                        CHKRANDOMTIMING(setpt,pulselength,outputch,data);
        end;

     until (opt='q') or (opt='Q');
   END;


{-------------------------------------------------------------------------}
{***  Draw Graph                                                      ***}
{-------------------------------------------------------------------------}

PROCEDURE drawgraf;

   BEGIN
      limits.tmin:=0;
      limits.tmax:=(data.nsamples-1)/data.frequency;
      limits.ymin:=0;
      limits.ymax:=2000;
      GRAPHSETUP(data,limits,1,data.title);
      for i:=1 to data.nochannel do
         begin
            PLOTDATALIN(data,i,limits);
            LABELS(data,i,i,data.labels[i],limits);
         end;
      repeat until keypressed;readln;
      closegraph;
   END;
```

```
{*************************************************************************}
{***                    MAIN PROGRAM                                   ***}
{*************************************************************************}

BEGIN
   INITDATA(data,pulselength,setpt);
   INITMENU1(menu1);

   outputch:=0;
   mode:='PULSE';
   save:='y';
   new:='y';
   name:='';
   insert(pathdata,name,1);

   SET_UP;
   DAC_WRITE(outputch,round(setpt*2048/10));


   repeat
      initmainmenu;
      option:=readkey;write(option);

      case option of
      #13:doresponse;
      'S','s':if new='n' then saveresponse
                else
                   begin
                      sound(2000);delay(100);nosound;
                   end;
      'T','t':data.step:=data.step*-1;
      'C','c':chgfixvar;
      'V','v':if new='n' then drawgraf
              else
                 begin
                    sound(2000);delay(100);nosound;
                 end;
      end;
   until (option='q') or (option='Q');

   if save='n' then
      begin
         clrscr;gotoXY(20,10);write('Save last Response? (n)');option:=readkey;
         if option=#13 then option:='n';
         if (option='Y') or (option='y') then saveresponse;
      end;

{*************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0       KCAD     01-03-90                                     ***}
{***                                                                     ***}
{***   FILEEND  DLOG.PAS                                                ***}
{*************************************************************************}
END.
```

C.3.11 Dlogd.pas

```
{*************************************************************************}
{***   UNIT PROCEDURE LIBARY CONTAINING DATAFILES TO BE USED IN        ***}
{***        DLOG.PAS                                                   ***}
{***   FILE: DLOGD.PAS                                                ***}
{*************************************************************************}

UNIT dlogd;

{*************************************************************************}
                        INTERFACE
{*************************************************************************}

   USES types;

   PROCEDURE initdata(VAR data:responsedata;VAR pulselength:integer;VAR setpt:real);
   PROCEDURE initmenu1(VAR menu:screenmenu);
   PROCEDURE initmenu2(VAR menu:screenmenu);


{*************************************************************************}
                        IMPLEMENTATION
{*************************************************************************}

{*************************************************************************}
{N**   MODULE NAME       : INITDATA                                     ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE         : defines default values for the data capture ***}
{***                       package DLOG                                ***}
{S**   CALL SEQUENCE     : INITDATA(data,length of pulse)               ***}
{I**   INPUT PARAMETERS  : none                                         ***}
{O**   OUTPUT PARAMETERS : response data record, length of pulse       ***}
{G**   GLOBAL VARIABLES  : none                                         ***}
{M**   MODULES CALLED    : Types                                        ***}
{E**   ERROR CONDITIONS  : none                                         ***}
{C**   COMMENTS          : for details about response data record see  ***}
{***                       TYPES                                       ***}
{*************************************************************************}

PROCEDURE initdata(VAR data:responsedata;VAR pulselength:integer;VAR setpt:real);

   BEGIN
      data.nsamples:=501;
      data.nochannel:=3;
      data.whenstep:=100;
      data.step:=-10.00;
      data.frequency:=50.0;
      data.labels[1]:='SPEED';
      data.labels[2]:='POWER';
      data.labels[3]:='TONNAGE';
      data.labels[4]:='VOLTAGE 1';
      data.labels[5]:='VOLTAGE 2';
      data.labels[6]:='VOLTAGE 3';
      data.title:=' ';

      pulselength:=250;
      setpt:=0.0;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITMENU1                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                         routine                                      ***}
{S**   CALL SEQUENCE      : INITMENU1(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                         ***}
{O**   OUTPUT PARAMETERS  : menu1 (type screenmenu)                      ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED     : Types                                        ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           : for details about screenmenu see TYPES       ***}
{*************************************************************************}

PROCEDURE initmenu1(VAR menu:screenmenu);

  BEGIN
    menu[1]:='╔═══════════════════════════════════════════════════════════╗';
    menu[2]:='║                                                           ║';
    menu[3]:='║            RESPONSE MEASURING PACKAGE                     ║';
    menu[4]:='║                                                           ║';
    menu[5]:='╠═══════════════════════════╦═══════════════════════════════╣';
    menu[6]:='║                           ║                               ║';
    menu[7]:='║                           ║                               ║';
    menu[8]:='║           MENU:           ║          FORMAT:              ║';
    menu[9]:='║                           ║                               ║';
   menu[10]:='║                           ║                               ║';
   menu[11]:='║                           ║                               ║';
   menu[12]:='║  <ENTER> - Start logging Data ║  No of Samples      :         ║';
   menu[13]:='║                           ║  Sampling Frequency :      Hz ║';
   menu[14]:='║                           ║                               ║';
   menu[15]:='║                           ║  TEST =        Response       ║';
   menu[16]:='║  v - View Response        ║  Size of            :         ║';
   menu[17]:='║  c - Change Format        ║                               ║';
   menu[18]:='║  s - Save Response        ║        at Sample    :         ║';
   menu[19]:='║                           ║                               ║';
   menu[20]:='║                           ║                               ║';
   menu[21]:='║  q - Quit                 ║                               ║';
   menu[22]:='║                           ║                               ║';
   menu[23]:='╚═══════════════════════════╩═══════════════════════════════╝';
   menu[24]:='                                                             ';
  END;

{*************************************************************************}
{N**   MODULE NAME        : INITMENU2                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                         routine                                      ***}
{S**   CALL SEQUENCE      : INITMENU2(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                         ***}
{O**   OUTPUT PARAMETERS  : menu2 (type screenmenu)                      ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED     : Types                                        ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           : for details about screenmenu see TYPES       ***}
{*************************************************************************}

PROCEDURE initmenu2(VAR menu:screenmenu);

  BEGIN
    menu[1]:='╔═══════════════════════════════════════════════════════════╗';
    menu[2]:='║                                                           ║';
    menu[3]:='║            CHANGE FORMAT:                                 ║';
    menu[4]:='║                                                           ║';
    menu[5]:='║                                                           ║';
    menu[6]:='║             m - Mode:        Response                     ║';
    menu[7]:='║             n - No of Samples          :                  ║';
    menu[8]:='║             f - Sampling Frequency     :        Hertz     ║';
    menu[9]:='║             e - Setpoint               :                  ║';
   menu[10]:='║             s - Size of                :                  ║';
   menu[11]:='║             w -       at Sample No     :                  ║';
   menu[12]:='║                                                           ║';
   menu[13]:='║                                                           ║';
   menu[14]:='║             o - Output channel to      :                  ║';
   menu[15]:='║             c - No of Channels to Log  :                  ║';
   menu[16]:='║             l - Labels:        Channel 1 :                ║';
   menu[17]:='║                                                           ║';
   menu[18]:='║                                                           ║';
   menu[19]:='║                                                           ║';
   menu[20]:='║                                                           ║';
   menu[21]:='║                                                           ║';
   menu[22]:='║       t - Calibrate Timing          q - Quit from this Menu ║';
   menu[23]:='╚═══════════════════════════════════════════════════════════╝';
   menu[24]:='                                                             ';
  END;


{*************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0       KCAD     28-02-90                                     ***}
{***                                                                     ***}
{***   UNITEND  DLOGD.PAS                                               ***}
{*************************************************************************}
END.
```

C.3.12 Graf.pas

```
{*************************************************************************}
{***   PROGRAM TO DISPLAY RESPONSE DATA FILES GRAPHICALLY              ***}
{***   FILE:  GRAF.PAS                                                ***}
{*************************************************************************}

{*************************************************************************}
{N**   PROGRAM NAME      : GRAF                                        ***}
{***   ------------------------------------------------------------    ***}
{A**   DESCRIPTION       : loads a real time response and displays the ***}
{***                       response on the screen                      ***}
{S**   CALL SEQUENCE     : Main Program                                ***}
{I**   INPUT PARAMETERS  : file to display graph off                  ***}
{O**   OUTPUT PARAMETERS : none                                        ***}
{G**   GLOBAL VARIABLES  : none                                        ***}
{M**   MODULES CALLED    : Plotgraf,Crt,Filehand,Graph,Types,Grafd     ***}
{E**   ERROR CONDITIONS  : none                                        ***}
{C**   COMMENTS          : none                                        ***}
{*************************************************************************}

PROGRAM graf;

{65000,0,655360}

   USES
      grafd,plotgraf,crt,filehand,graph,types;

   CONST
      pathdata='data\';

   TYPE
      labtype=string[15];
      {str50 defined in Filehand as string[50]}

   VAR
      recorddata:array[1..5] of responsedata;
      lab:array[1..5] of labtype;
      name:array[1..5] of str50;
      channel:array[1..5] of integer;
      i,nochannel,pos:integer;
      limits:responsescale;
      disc:text;
      id:string[4];
      title:str50;
      menu1:screenmenu;
      load,new,option,key:char;


{*************************************************************************}
{***                      PROCEDURES                                   ***}
{*************************************************************************}

{-------------------------------------------------------------------------}
{***   change format of the responses to display                       ***}
{-------------------------------------------------------------------------}

PROCEDURE chgformat;

   VAR
      pos,i,j:integer;
      menu:screenmenu;
      clrlabel:string[15];
      prevname,clrname:str50;
      key:char;


   PROCEDURE dispmenu;

      BEGIN
         clrscr;
         WRITEMENU(menu);
         HIGHLIGHT(0);
         gotoXY(52,7);write(i);
         normvideo;
         gotoXY(22,11);write(lab[i]);
         gotoXY(22,13);write(name[i]);
         gotoXY(22,15);write(channel[i]);
      END;
```

```
BEGIN
   INITMENU2(menu);
   clrlabel:='                ';
   clrname:='                                                  ';

   for i:=1 to nochannel do
      begin
         dispmenu;
         gotoXY(22,19);write('Press <c> to Change or <ENTER> to Continue  ');
         key:=readkey;

         if key <> #13 then
            begin
               dispmenu;
               gotoXY(22,19);write('Press <ENTER> to Continue  ');
               HIGHLIGHT(0);
               gotoXY(22,11);writeln(clrlabel);
               lab[i]:=CHGSTRING(lab[i],clrlabel,1,22,11);
               normvideo;

               dispmenu;
               gotoXY(22,19);write('Press <ENTER> to Continue  ');
               HIGHLIGHT(0);
               pos:=POSITION(name[i])-4;
               gotoXY(22,13);writeln(clrname);
               prevname:=name[i];
               load:='y';
               name[i]:=CHGSTRING(name[i],clrname,pos,22,13);
               if prevname<>name[i] then
                  begin
                     for j:= 1 to nochannel do
                        if (name[i]=name[j]) and (i<>j) and (load='y') then
                           begin
                              COPYRESPONSEDATA(recorddata[j],recorddata[i]);
                              load:='n';
                              sound(1000);delay(500);nosound;
                           end;
                     if load='y' then LOADRESPONSEDATA(name[i],recorddata[i]);
                  end
               else if recorddata[i].frequency=0 then COPYRESPONSEDATA(recorddata[1],recorddata[i]);

               normvideo;

               dispmenu;
               gotoXY(22,19);write('Press <ENTER> to Continue  ');
               HIGHLIGHT(0);
               gotoXY(22,15);writeln(channel[i]);
               gotoXY(22,15);key:=readkey;
               if key<>#13 then VAL(key,channel[i],j);
               normvideo;

               dispmenu;

               if i=1 then
                  begin
                     if new='y' then
                        begin
                           for j:=2 to 5 do name[j]:=name[1];
                           AUTORESPSCALE(recorddata[i],limits,channel[i]);
                           new:='n';
                        end
                     else
                        begin
                           gotoXY(22,19);write('AUTOscale [N]?  ');key:=readkey;
                           if (key='Y') or (key='y') then AUTORESPSCALE(recorddata[1],limits,channel[1]);
                        end;

                     gotoXY(15,19);write('Fix File <f>, Channel <c> or Nothing <ENTER> Globally  ');
                     case readkey of
                        'F','f':for j:=2 to 5 do
                                    begin
                                       name[j]:=name[1];
                                       COPYRESPONSEDATA(recorddata[1],recorddata[j]);
                                    end;
                        'C','c':for j:=2 to 5 do channel[j]:=channel[1];
                     end;
                  end;
            end;

      end;
      title:=CHGTITLE('Title of Graph ',title);
END;
```

C.3.13 Grafd.pas

```
{*************************************************************************}
{***   UNIT PROCEDURE LIBARY CONTAINING DATAFILES TO BE USED IN        ***}
{***        GRAF.PAS                                                   ***}
{***   FILE: GRAFD.PAS                                                 ***}
{*************************************************************************}

UNIT grafd;

{*************************************************************************}
                        INTERFACE
{*************************************************************************}

   USES types;

   PROCEDURE initmenu1(VAR menu:screenmenu);
   PROCEDURE initmenu2(VAR menu:screenmenu);


{*************************************************************************}
                        IMPLEMENTATION
{*************************************************************************}

{*************************************************************************}
{N**   MODULE NAME        : INITMENU1                                   ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-   ***}
{***                        routine                                     ***}
{S**   CALL SEQUENCE      : INITMENU1(menu)                             ***}
{I**   INPUT PARAMETERS   : none                                        ***}
{O**   OUTPUT PARAMETERS  : menu1 (type screenmenu)                     ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : Types                                       ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : details about screenmenu in TYPES           ***}
{*************************************************************************}

PROCEDURE initmenu1(VAR menu:screenmenu);

   BEGIN
      menu[1]:='
      menu[2]:='
      menu[3]:='
      menu[4]:='
      menu[5]:='
      menu[6]:='
      menu[7]:='
      menu[8]:='
      menu[9]:='
     menu[10]:='
     menu[11]:='
     menu[12]:='
     menu[13]:='
     menu[14]:='
     menu[15]:='
     menu[16]:='
     menu[17]:='
     menu[18]:='
     menu[19]:='
     menu[20]:='
     menu[21]:='
     menu[22]:='
     menu[23]:='
     menu[24]:='
   END;
```

```
╔═══════════════════════════════════════════════════════════════════════════╗
║                                                                           ║
║                  GRAPHICAL RESPONSE DISPLAY PACKAGE                       ║
║                                                                           ║
║                                                                           ║
║                                MENU:                                      ║
║                                                                           ║
║                                                                           ║
║                  n - No of channels to display (1..5) =                   ║
║                  d - Define files to display                              ║
║                                                                           ║
║                  c - Change scales of Graph                               ║
║                  v - View Graph                                           ║
║                                                                           ║
║                                                                           ║
║                  q - Quit                                                 ║
╚═══════════════════════════════════════════════════════════════════════════╝
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITMENU2                                     ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                         routine                                      ***}
{S**   CALL SEQUENCE      : INITMENU2(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                         ***}
{O**   OUTPUT PARAMETERS  : menu2  (type screenmenu)                      ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED     : Types                                        ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           : details about screenmenu in TYPES            ***}
{*************************************************************************}

PROCEDURE initmenu2(VAR menu:screenmenu);

  BEGIN
      menu[1]:='╔═══════════════════════════════════════════════════════════╗';
      menu[2]:='║                                                           ║';
      menu[3]:='║              DEFINE FILES TO DISPLAY                      ║';
      menu[4]:='║                                                           ║';
      menu[5]:='║                                                           ║';
      menu[6]:='║                                                           ║';
      menu[7]:='║              FORMAT OF DISPLAY CHANNEL                    ║';
      menu[8]:='║                                                           ║';
      menu[9]:='║                                                           ║';
     menu[10]:='║                                                           ║';
     menu[11]:='║        Label  :                                           ║';
     menu[12]:='║                                                           ║';
     menu[13]:='║        File   :                                           ║';
     menu[14]:='║                                                           ║';
     menu[15]:='║        Channel:                                           ║';
     menu[16]:='║                                                           ║';
     menu[17]:='║                                                           ║';
     menu[18]:='║                                                           ║';
     menu[19]:='║                                                           ║';
     menu[20]:='╚═══════════════════════════════════════════════════════════╝';
     menu[21]:='                                                            ';
     menu[22]:='                                                            ';
     menu[23]:='                                                            ';
     menu[24]:='                                                            ';
  END;


{*************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0       KCAD     28-02-90                                     ***}
{***                                                                     ***}
{***   UNITEND  GRAFD.PAS                                                ***}
{*************************************************************************}
END.
```

C.3.14 Mfcm.pas

```
{*************************************************************************}
{***   PROGRAM TO CORRELATE MASSFLOW DERIVED FROM THE ENERGY RESPONSE  ***}
{***          TO ACTUAL MASSFLOW ON AN CONVEYOR BELT                   ***}
{***   FILE: MFCM.PAS                                                  ***}
{*************************************************************************}

{*************************************************************************}
{N**   PROGRAM NAME       : MFCM                                        ***}
{***   -------------------------------------------------------------   ***}
{A**   DESCRIPTION        : correlates the actual massflow on a horizon-***}
{***                        tal conveyor to a massflow figure determined***}
{***                        from the kinetic energy response of the sys-***}
{***                        tem, calling the responses manually from a  ***}
{***                        datafile                                    ***}
{S**   CALL SEQUENCE      : MAIN PROGRAM                                ***}
{I**   INPUT PARAMETERS   : name of datafile                           ***}
{O**   OUTPUT PARAMETERS  : actual and determined massflow figures      ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : Crt,Types,Filehand,Detmf,Math,Plotgraf,     ***}
{***                        Mfcmd                                       ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : none                                        ***}
{*************************************************************************}

PROGRAM mfcm;

{$M 65000,0,655360}

   USES Crt,types,filehand,mfcmd,detmf,math,plotgraf,graph;


   CONST
      pathdata='data\';
      pathcorr='data\';


   VAR
      corrname,name,namef:str50;
      nset,ntest,mode,actmf,samplepos,testpos:integer;
      corrdata:correlationdata;
      regdata:regressiondata;
      corrlimits:corrscale;
      Csp:constant;
      Cm:massflowconst;
      Ct:time;
      menu0:screenmenu;
      option,new:char;


{*************************************************************************}
{***                    PROCEDURES                                     ***}
{*************************************************************************}

{*************************************************************************}
{***  Manual Massflow Data Aquasition                                  ***}
{*************************************************************************}

PROCEDURE manualmassflow;

   VAR
      menu:screenmenu;
      opt:char;
```

```
{------------------------------------------------------------------------}
{***  change position of counters in filenames                      ***}
{------------------------------------------------------------------------}

   PROCEDURE chgpos;

      BEGIN
         repeat
            clrscr;
            gotoXY(15,7);writeln('Filename: ',name);
            gotoXY(15,15);writeln('enter position of main sample counter using direction keys: ');
            HIGHLIGHT(1);gotoXY(24+samplepos,8);write('^');normvideo;
            option:=readkey;

            if option<>#13 then
               if option=#0 then
                  case readkey of
                     #75:samplepos:=samplepos-1;
                     #77:samplepos:=samplepos+1;
                  end;

         until option=#13;

         repeat
            clrscr;
            gotoXY(15,7);writeln('Filename: ',name);
            gotoXY(15,15);writeln('enter position of test counter using direction keys: ');
            HIGHLIGHT(1);gotoXY(24+samplepos,8);write('C');
            gotoXY(24+testpos,8);write('^');normvideo;
            option:=readkey;

            if option<>#13 then
               if option=#0 then
                  case readkey of
                     #75:testpos:=testpos-1;
                     #77:testpos:=testpos+1;
                  end;

         until option=#13;
      END;


{------------------------------------------------------------------------}
{***  change filenames to be used in porcess                        ***}
{------------------------------------------------------------------------}

   PROCEDURE chgname;

      VAR
         menu:screenmenu;
         opt:char;

      BEGIN
         INITMENU12(menu);

         repeat
            clrscr;WRITEMENU(menu);
            gotoXY(19,9);write(name);
            gotoXY(19,12);
            if actmf=0 then write(namef)
            else
               begin
                  gotoXY(5,11);writeln('                                                    ');
                  gotoXY(5,8);writeln('                                                    ');
               end;
            gotoXY(19,15);write(corrname);
            gotoXY(19,18);write(corrdata.title);
            gotoXY(4,25);write('Option[q]: ');
            opt:=readkey;
            if opt=#13 then opt:='q';

            case opt of
               'n','N':FILENAME(name,'');
               'z','Z':if actmf=0 then FILENAME(namef,'');
               's','S':FILENAME(corrname,'');
               't','T':corrdata.title:=CHGTITLE('Title ',corrdata.title);
               'p','P':chgpos;
            end;

         until opt='q';
      END;
```

```
{------------------------------------------------------------------------}
{***  Initialize Variables                                            ***}
{------------------------------------------------------------------------}

   PROCEDURE initvars;

      VAR
         modechar,mfchar:string[20];
         option:char;
         pulselength:integer;
         menu:screenmenu;

      BEGIN
         INITMENU11(menu);
         repeat
            case mode of
               3:modechar:='STANDARD: LONG PULSE';
               4:modechar:='SPECIAL: SHORT PULSE';
            end;

            case actmf of
               0:mfchar:='BINMASS';
               1:mfchar:='NUCLEAR';
            end;

            clrscr;WRITEMENU(menu);
            gotoXY(56,10);writeln(nset);
            gotoXY(56,12);writeln(ntest);
            gotoXY(50,18);writeln(mfchar);
            gotoXY(45,19);writeln(modechar);
            gotoXY(15,14);
            if actmf=0 then writeln('z - no of Zero Massflow Samples       :  ',corrdata.nzero);
            gotoXY(5,25);write('Option [q]: ');


            option:=readkey;
            if option=#13 then option:='q';

            case option of
               'n','N':begin
                         READINT(56,10,5,nset);
                         corrdata.nsamples:=nset*ntest;
                       end;
               't','T':begin
                         READINT(56,12,5,ntest);
                         corrdata.nsamples:=nset*ntest;
                       end;
               'z','Z':if actmf=0 then READINT(56,14,5,corrdata.nzero);
               'a','A':begin
                         actmf:=actmf+1;
                         if actmf=2 then actmf:=0;
                       end;
               'm','M':begin
                         mode:=mode+1;
                         if mode>4 then mode:=3;
                       end;
               'f','F':chgname;
            end;
         until (option='Q') or (option='q');

         if mode=4 then
            begin
               pulselength:=Ct.stepup-Ct.stepdn;
               HIGHLIGHT(0);
               gotoXY(25,22);write('PULELENGTH [',pulselength:4,']: ');READINT(44,22,4,pulselength);
               Ct.ssdn:=Ct.stepdn + pulselength;
               Ct.stepup:=Ct.ssdn;
               normvideo;
            end;
      END;
```

```
{------------------------------------------------------------------------}
{***  determine massflow on belt                                    ***}
{------------------------------------------------------------------------}

   PROCEDURE mfbelt(name:string);

      VAR
         i,j:integer;
         s:string[1];

{------------------------------------------------------------------------}

      PROCEDURE getmass(point,n:integer;name:str50);

         VAR
            j,k:integer;
            massflowup,massflowdn,Vav:real;
            respdata:responsedata;

         BEGIN
            for j:=1 to n do
               begin
                  delete(name,testpos,1);
                  str(j,s);
                  insert(s,name,testpos);
                  LOADRESPONSEDATA(name,respdata);

                  DETMASSFLOW(Csp,Ct,respdata,mode,massflowdn,massflowup,Vav);
                  if actmf=0 then corrdata.data[1,point+j]:=MASSBIN(Cm,respdata)
                  else corrdata.data[1,point+j]:=MASSFLOWNUC(Cm,respdata);

                  corrdata.data[2,point+j]:=massflowdn;
                  corrdata.data[3,point+j]:=massflowup;
                  corrdata.data[4,point+j]:=Vav;
               end;
         END;

{------------------------------------------------------------------------}

      PROCEDURE chgsetcounter;

         VAR
            code,no,no1:integer;
            s2:string[2];


         BEGIN
            s:=copy(name,samplepos,1);
            val(s,no,code);

            if code <> 0 then
               repeat
                  clrscr;
                  gotoXY(20,10);writeln('Invlid Counter entered! ');
                  gotoXY(20,12);writeln('Press <ENTER> to continue');readln;
                  chgpos;
                  s:=copy(name,samplepos,1);
                  val(s,no,code);
               until code = 0;

            no:=no+1;
            if no<10 then
               begin
                  delete(name,samplepos,1);
                  str(no,s);
                  insert(s,name,samplepos);
               end
            else
               begin
                  s:=copy(name,samplepos-1,1);
                  val(s,no1,code);
                  if code=0 then no:=no+no1*10;

                  delete(name,samplepos-1,2);
                  str(no,s2);
                  insert(s2,name,samplepos-1);
               end;
         END;

{------------------------------------------------------------------------}

      BEGIN
         for i:=1 to nset do
            begin
               getmass((i-1)*ntest,ntest,name);
               chgsetcounter;
            end;

         if actmf=0 then
            begin
               getmass(corrdata.nsamples,corrdata.nzero,namef);
               MASSFLOWBELT(Cm,corrdata);
            end;

         corrlimits.ymax[1]:=corrlimits.ymin[1];
         corrlimits.ymax[2]:=corrlimits.ymin[2];
      END;
```

```
{-------------------------------------------------------------------------}
{***  Main Body of Procedure: MANUALMASSFLOW                          ***}
{-------------------------------------------------------------------------}

   BEGIN
      INITMENU1(menu);

      repeat
         clrscr;WRITEMENU(menu);
         gotoXY(5,25);write('Option? [q] ');
         opt:=readkey;if opt=#13 then opt:='q';

         case opt of
            'v','V':initvars;
            'r','R':if nset>0 then
                       begin
                          mfbelt(name);
                          REGRESS(corrdata,regdata);
                          AUTOCORRSCALE(corrdata,regdata,corrlimits);
                          new:='n';
                       end
                    else
                       begin
                          sound(1000);delay(100);nosound;
                       end;
            'f','F':chgname;
         end;

      until (opt='q') or (opt='Q');
   END;


{*************************************************************************}
{***  display results of regression test                              ***}
{*************************************************************************}

PROCEDURE results;

   VAR
      i:integer;
      xmax:real;
      menu:screenmenu;
      opt:char;

   BEGIN
      INITMENU2(menu);

      xmax:=0;
      for i:=1 to corrdata.nsamples do
          if corrdata.data[1,i]>xmax then xmax:=corrdata.data[1,i];

      repeat
         clrscr;WRITEMENU(menu);

         with regdata do
            for i:=1 to 2 do
               begin
                  gotoXY(38+(i-1)*20,10);writeln(slope[i]:8:4);
                  gotoXY(38+(i-1)*20,11);writeln(yconst[i]:8:4);

                  gotoXY(38+(i-1)*20,13);writeln(xmax:6:2,' t/hr');
                  gotoXY(38+(i-1)*20,14);writeln(error[i]:6:2,' t/hr');
                  gotoXY(38+(i-1)*20,15);writeln(error[i]/xmax*100:6:2,' %');
                  gotoXY(38+(i-1)*20,17);writeln(r[i]:6:4);
               end;
         gotoXY(5,25);write('Option: [q]');
         opt:=readkey;if opt=#13 then opt:='q';

         case opt of
            'c','C':CHGCORRSCALE(corrlimits);
            'v','V':begin
                       insert('STEP-DOWN: ',corrdata.title,1);
                       CORRGRAPH(corrdata,regdata,corrlimits,1);
                       delete(corrdata.title,1,11);

                       insert('STEP-UP: ',corrdata.title,1);
                       CORRGRAPH(corrdata,regdata,corrlimits,2);
                       delete(corrdata.title,1,9);
                    end;
         end;
      until (opt='q') or (opt='Q');
   END;
```

```
{*************************************************************************}
{***  transfer files to and from disc                                  ***}
{*************************************************************************}

PROCEDURE transfer;

   VAR
      menu:screenmenu;
      opt:char;

   BEGIN
      INITMENU3(menu);

      repeat
         clrscr;WRITEMENU(menu);
         gotoXY(5,25);write('Option? [q]');
         opt:=readkey;if opt=#13 then opt:='q';

         case opt of
            'l','L':begin
                      FILENAME(corrname,'');
                      LOADCORRDATA(corrname,corrdata,regdata);
                      REGRESS(corrdata,regdata);
                      AUTOCORRSCALE(corrdata,regdata,corrlimits);
                      opt:='q';
                      new:='n';
                    end;
            's','S':begin
                      FILENAME(corrname,'');
                      SAVECORRDATA(corrname,corrdata,regdata);
                      opt:='q';
                    end;
         end;
      until (opt='q') or (opt='Q');

   END;


{*************************************************************************}
{***  Data Processing option                                           ***}
{*************************************************************************}

PROCEDURE dataproc;

   VAR
      i:integer;
      filttime,nuczero,zerocali,slopecali,tsamp,totalact,totaldet:real;
      tempcorr:correlationdata;
      tempreg:regressiondata;
      timedata:responsedata;
      timelimits:responsescale;
      opt:char;
      menu:screenmenu;
      done:boolean;


{-------------------------------------------------------------------------}
  PROCEDURE processdata;

      VAR t0,t1,i,nsamp:integer;


      BEGIN
         clrscr;
         HIGHLIGHT(1);gotoXY(20,10);write(' Please Wait - Processing Data ');
         normvideo;

         t0:=round(timelimits.tmin*timedata.frequency);
         if t0<0 then t0:=0;
         t1:=round(timelimits.tmax*timedata.frequency);
         if t1>corrdata.nsamples then t1:=corrdata.nsamples;
         nsamp:=t1-t0+1;

         for i:=1 to timedata.nsamples do
            begin
               timedata.data[1,i-1]:=round((corrdata.data[1,i]-nuczero)*10);
               timedata.data[2,i-1]:=round((corrdata.data[3,i]-zerocali)/slopecali*10);
{               timedata.data[3,i-1]:=rotlund(corrdata.data[4,i]*100);}
            end;

         if filttime<=0 then filttime:=0.00000000000001;
         for i:=1 to 2 do FILTER(1/filttime,i,timedata);

         totalact:=AREA(timedata,1,t0,nsamp-1)/3600/10;
         totaldet:=AREA(timedata,2,t0,nsamp-1)/3600/10;

         tempcorr.nsamples:=nsamp;
         for i:=t0 to t1 do
            begin
               tempcorr.data[1,i+1-t0]:=timedata.data[1,i]/10;
               tempcorr.data[2,i+1-t0]:=0.0001;
               tempcorr.data[3,i+1-t0]:=timedata.data[2,i]/10;
            end;

         REGRESS(tempcorr,tempreg);
      END;
```

```
{-------------------------------------------------------------------------}

PROCEDURE dispgraph;

   VAR
      i:integer;
      menu:screenmenu;
      opt:char;

   BEGIN
      INITMENU41(menu);

      repeat
         clrscr;WRITEMENU(menu);

         gotoXY(5,25);write('Option? [q] ');
         opt:=readkey;if opt=#13 then opt:='q';

         case opt of
            't','T':begin
                      if done=false then processdata;done:=true;
                      GRAPHSETUP(timedata,timelimits,1,'TONNAGE');
                      for i:=1 to 2 do
                         PLOTDATALIN(timedata,i,timelimits);
                      LABELS(timedata,1,1,'Actual Tonnage',timelimits);
                      LABELS(timedata,2,2,'Determined Tonnage',timelimits);
                      writeln(Eoln);closegraph;readln;
                    end;
            'c','C':begin
                      if done=false then processdata;done:=true;
                      CORRGRAPH(tempcorr,tempreg,corrlimits,2);
                    end;
            's','S':begin
                      CHGSCALE(timelimits);
                      corrlimits.ymin[2]:=timelimits.ymin/10;
                      corrlimits.ymax[2]:=timelimits.ymax/10;
                      corrlimits.xmin[2]:=timelimits.ymin/10;
                      corrlimits.xmax[2]:=timelimits.ymax/10;
                      processdata;
                    end;
            'a','A':begin
                      AUTORESPSCALE(timedata,timelimits,1);
                      AUTOCORRSCALE(tempcorr,tempreg,corrlimits);
                      processdata;
                    end;
         end;

      until (opt='q') or (opt='Q');

   END;


{-------------------------------------------------------------------------}
{***   Main body of Data Processing Procedure                          ***}
{-------------------------------------------------------------------------}

   BEGIN
      clrscr;
      HIGHLIGHT(0);gotoXY(5,10);
      write(' Please enter sampling time of tonnage samples (in sec) : ');READREAL(63,10,5,tsamp);normvideo;

      filttime:=0;
      nuczero:=0;
      zerocali:=regdata.yconst[2];
      slopecali:=regdata.slope[2];

      timedata.frequency:=1/tsamp;
      timedata.nsamples:=corrdata.nsamples;
      timedata.nochannel:=3;
      timedata.labels[1]:='*0.1 t/hr';
      tempcorr.nsamples:=timedata.nsamples;

      timelimits.tmin:=0;
      timelimits.tmax:=(timedata.nsamples-1)/timedata.frequency;
      processdata;done:=true;
      AUTORESPSCALE(timedata,timelimits,1);
      AUTOCORRSCALE(tempcorr,tempreg,corrlimits);

      INITMENU4(menu);

      repeat
         clrscr;writemenu(menu);

         gotoXY(13,11);writeln(nuczero:5:2);
         gotoXY(13,14);writeln(zerocali:5:2);
         gotoXY(13,16);writeln(slopecali:5:2);
         gotoXY(13,19);writeln(filttime:5:2);
         gotoXY(60,10);writeln(tempreg.yconst[2]:5:4);
         gotoXY(60,11);writeln(tempreg.slope[2]:5:4);
         gotoXY(60,14);writeln(tempreg.r[2]:5:4);
         gotoXY(57,21);writeln(totalact:8:2);
         gotoXY(57,22);writeln(totaldet:8:2);
         gotoXY(60,23);writeln(((totaldet-totalact)/totalact*100):5:3);
```

```
        gotoXY(5,25);write('Option [g]: ');
        opt:=readkey;if opt=#13 then opt:='g';

        case opt of
           'o','O':begin
                     READREAL(11,11,7,nuczero);
                     done:=false;
                   end;
           'z','Z':begin
                     READREAL(11,14,9,zerocali);
                     done:=false;
                   end;
           's','S':begin
                     READREAL(11,16,7,slopecali);
                     done:=false;
                   end;
           'f','F':begin
                     READREAL(13,19,7,filttime);
                     done:=false;
                   end;
           'g','G':dispgraph;
           'c','C':begin
                     processdata;done:=true;
                   end;
        end;
     until (opt='q') or (opt='Q');
  END;


{*************************************************************************}
{***                    MAIN PROGRAM                                   ***}
{*************************************************************************}

BEGIN
   LOADCONST(Csp);
   LOADCONSTM(Cm);
   INITVAR(Ct,nset,ntest,corrdata.nzero,samplepos,testpos,mode,actmf);
   with corrdata do
      begin
         nsamples:=nzero+nset*ntest;
         title:='MASSFLOW CORRELATION: '
      end;

   new:='y';

   corrname:='';
   insert('.cor',corrname,1);
   insert(pathcorr,corrname,1);

   name:='';
   insert('.dat',name,1);
   insert('',name,1);
   insert(pathdata,name,1);
   namef:=name;
   INITMENU0(menu0);

   repeat
      clrscr;
      WRITEMENU(menu0);
      gotoXY(5,25);write('Option? ');
      option:=readkey;

      case option of
         'm','M':manualmassflow;
         'r','R':if new='n' then results
                 else
                    begin
                       sound(1000);delay(100);nosound;
                    end;
         't','T':transfer;
         'd','D':if new='n' then dataproc
                 else
                    begin
                       sound(1000);delay(100);nosound;
                    end;
      end;
   until (option='q') or (option='Q');

{*************************************************************************}
{H**   REVISION HISTORY:                                                ***}
{***   VERSION      BY        DATE       COMMENTS                       ***}
{***     1.0       KCAD     01-03-90                                    ***}
{***                                                                    ***}
{***   FILEEND  MFCM.PAS                                                ***}
{*************************************************************************}
END.
```

C.3.15 Mfcmd.pas

```
{************************************************************************}
{***   UNIT PROCEDURE LIBARY CONTAINING DATAFILES TO BE USED IN        ***}
{***        MFCM.PAS                                                   ***}
{***   FILE: MFCMD.PAS                                                 ***}
{************************************************************************}

UNIT mfcmd;

{************************************************************************}
                        INTERFACE
{************************************************************************}

   USES types;

   PROCEDURE initvar(VAR Ct:time; VAR nsamples,ntest,nzero,samplepos,testpos,mode,actmf:integer);
   PROCEDURE initmenu0(VAR menu:screenmenu);
   PROCEDURE initmenu1(VAR menu:screenmenu);
   PROCEDURE initmenu11(VAR menu:screenmenu);
   PROCEDURE initmenu12(VAR menu:screenmenu);
   PROCEDURE initmenu2(VAR menu:screenmenu);
   PROCEDURE initmenu3(VAR menu:screenmenu);
   PROCEDURE initmenu4(VAR menu:screenmenu);
   PROCEDURE initmenu41(VAR menu:screenmenu);

{************************************************************************}
                        IMPLEMENTATION
{************************************************************************}

{************************************************************************}
{N**   MODULE NAME        : INITVAR                                    ***}
{***   ------------------------------------------------------------   ***}
{A**   PROCEDURE          : Initializes the Variables used by the      ***}
{***                        Program MFCM                               ***}
{S**   CALL SEQUENCE      : INITVAR(Ct,nsamples,ntest,nzero,samplepos, ***}
{***                              testpos,mode)                        ***}
{I**   INPUT PARAMETERS   : none                                       ***}
{O**   OUTPUT PARAMETERS  : timing variables, no of samples, no of tests***}
{***                        per sample, no of initilizing samples,     ***}
{***                        position of sample and test counter in file-***}
{***                        name and the initial operation mode        ***}
{G**   GLOBAL VARIABLES   : none                                       ***}
{M**   MODULES CALLED     : Types                                      ***}
{E**   ERROR CONDITIONS   : none                                       ***}
{C**   COMMENTS           : for details about time see TYPES           ***}
{************************************************************************}

PROCEDURE initvar(VAR Ct:time; VAR nsamples,ntest,nzero,samplepos,testpos,mode,actmf:integer);

   BEGIN
      Ct.ssup:=0;
      Ct.ssdn:=250;
      Ct.stepdn:=100;
      Ct.stepup:=300;
      Ct.ssend:=500;

      nsamples:=10;
      ntest:=5;
      nzero:=5;

      samplepos:=6;
      testpos:=6;

      mode:=3;    {i.e. long pulse option}
      actmf:=0;   {0 - actual massflow is obtained from mass in the hopper bin}
                  {1 - actual massflow is obtained from a nuclear weightometer}
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITMENU0                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                        routine                                       ***}
{S**   CALL SEQUENCE      : INITMENU0(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : menu0 (type screenmenu)                       ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about screenmenu see TYPES        ***}
{*************************************************************************}

PROCEDURE initmenu0(VAR menu:screenmenu);

  BEGIN
       menu[1]:='╔═══════════════════════════════════════════════════════════╗';
       menu[2]:='║                                                           ║';
       menu[3]:='║          MASSFLOW CORRELATION PACKAGE                     ║';
       menu[4]:='║                                                           ║';
       menu[5]:='║                                                           ║';
       menu[6]:='║                        MENU:                              ║';
       menu[7]:='║                                                           ║';
       menu[8]:='║                                                           ║';
       menu[9]:='║                                                           ║';
      menu[10]:='║                                                           ║';
      menu[11]:='║                m - Massflow Data Aquasition from          ║';
      menu[12]:='║                       Real Time Responses                 ║';
      menu[13]:='║                                                           ║';
      menu[14]:='║                r - Results                                ║';
      menu[15]:='║                                                           ║';
      menu[16]:='║                t - Transfer Data to and from file         ║';
      menu[17]:='║                                                           ║';
      menu[18]:='║                d - Data Processing                        ║';
      menu[19]:='║                                                           ║';
      menu[20]:='║                                                           ║';
      menu[21]:='║                                                           ║';
      menu[22]:='║                q - Quit                                   ║';
      menu[23]:='║                                                           ║';
      menu[24]:='╚═══════════════════════════════════════════════════════════╝';

  END;

{*************************************************************************}
{N**   MODULE NAME        : INITMENU1                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                        routine                                       ***}
{S**   CALL SEQUENCE      : INITMENU1(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : menu  (type screenmenu)                       ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about screenmenu see TYPES        ***}
{*************************************************************************}

PROCEDURE initmenu1(VAR menu:screenmenu);

  BEGIN
       menu[1]:='╔═══════════════════════════════════════════════════════════╗';
       menu[2]:='║                                                           ║';
       menu[3]:='║       Manual Massflow Data Acquisition                    ║';
       menu[4]:='║                                                           ║';
       menu[5]:='║                                                           ║';
       menu[6]:='║                        MENU:                              ║';
       menu[7]:='║                                                           ║';
       menu[8]:='║                                                           ║';
       menu[9]:='║                                                           ║';
      menu[10]:='║                                                           ║';
      menu[11]:='║                v - Variables Initialization               ║';
      menu[12]:='║                                                           ║';
      menu[13]:='║                f - Filenames Definition                   ║';
      menu[14]:='║                                                           ║';
      menu[15]:='║                r - Run                                    ║';
      menu[16]:='║                                                           ║';
      menu[17]:='║                                                           ║';
      menu[18]:='║                                                           ║';
      menu[19]:='║                                                           ║';
      menu[20]:='║                                                           ║';
      menu[21]:='║                                                           ║';
      menu[22]:='║                q - Quit                                   ║';
      menu[23]:='║                                                           ║';
      menu[24]:='╚═══════════════════════════════════════════════════════════╝';

  END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITMENU11                                    ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                        routine                                       ***}
{S**   CALL SEQUENCE      : INITMENU11(menu)                              ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : menu                                          ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about screenmenu see TYPES        ***}
{*************************************************************************}

PROCEDURE initmenu11(VAR menu:screenmenu);

  BEGIN
       menu[1]:='                                                                ';
       menu[2]:='                                                                ';
       menu[3]:='          Variables Initialization                              ';
       menu[4]:='                                                                ';
       menu[5]:='                                                                ';
       menu[6]:='                                                                ';
       menu[7]:='                        MENU:                                   ';
       menu[8]:='                                                                ';
       menu[9]:='                                                                ';
      menu[10]:='          n - No of Massflow Sample Sets        :              ';
      menu[11]:='              (including Initial Zero Massflow Samples)         ';
      menu[12]:='          t - No of Tests per Set               :              ';
      menu[13]:='                                                                ';
      menu[14]:='                                                                ';
      menu[15]:='                                                                ';
      menu[16]:='                                                                ';
      menu[17]:='                                                                ';
      menu[18]:='          a - Actual Massflow obtained from                     ';
      menu[19]:='          m - Mode of Operation      :                          ';
      menu[20]:='                                                                ';
      menu[21]:='                                                                ';
      menu[22]:='                                                                ';
      menu[23]:='          q - Quit                                              ';
      menu[24]:='                                                                ';

  END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITMENU12                                    ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                        routine                                       ***}
{S**   CALL SEQUENCE      : INITMENU12(menu)                              ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : menu                                          ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about screenmenu see TYPES        ***}
{*************************************************************************}

PROCEDURE initmenu12(VAR menu:screenmenu);

  BEGIN
       menu[1]:='                                                                ';
       menu[2]:='                                                                ';
       menu[3]:='          Filenames Definition                                  ';
       menu[4]:='                                                                ';
       menu[5]:='                                                                ';
       menu[6]:='                                                                ';
       menu[7]:='          n - Filename of First Massflow Sample :               ';
       menu[8]:='              (i.e. First Zero Massflow Sample)                 ';
       menu[9]:='                                                                ';
      menu[10]:='                                                                ';
      menu[11]:='          z - Filename of Final Zero Massflow Sample :          ';
      menu[12]:='                                                                ';
      menu[13]:='                                                                ';
      menu[14]:='          s - Filename to Save Correlation Data to:             ';
      menu[15]:='                                                                ';
      menu[16]:='                                                                ';
      menu[17]:='          t - Title of Correlation Test                         ';
      menu[18]:='                                                                ';
      menu[19]:='                                                                ';
      menu[20]:='                                                                ';
      menu[21]:='          p - Indicate Position of Counters in Filenames        ';
      menu[22]:='                                                                ';
      menu[23]:='          q - Quit                                              ';
      menu[24]:='                                                                ';

  END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITMENU2                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                         routine                                      ***}
{S**   CALL SEQUENCE      : INITMENU2(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                         ***}
{O**   OUTPUT PARAMETERS  : menu2 (type screenmenu)                      ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED     : Types                                        ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           : for details about screenmenu see TYPES       ***}
{*************************************************************************}

PROCEDURE initmenu2(VAR menu:screenmenu);

  BEGIN
       menu[1]:='
       menu[2]:='
       menu[3]:='
       menu[4]:='
       menu[5]:='
       menu[6]:='
       menu[7]:='
       menu[8]:='
       menu[9]:='
      menu[10]:='
      menu[11]:='
      menu[12]:='
      menu[13]:='
      menu[14]:='
      menu[15]:='
      menu[16]:='
      menu[17]:='
      menu[18]:='
      menu[19]:='
      menu[20]:='
      menu[21]:='
      menu[22]:='
      menu[23]:='
      menu[24]:='
```

```
╔═══════════════════════════════════════════════════════════════════════╗';
║                                                                       ║';
║                  Results:                                             ║';
║                                                                       ║';
║                                                                       ║';
║                                                As from :              ║';
║                                        ┌──────────────┬──────────────┐ ║';
║                                        │ Step Dn Response │ Step Up Response │ ║';
║    Y constant:                         │              │              │ ║';
║    Slope:                              │              │              │ ║';
║                                        │              │              │ ║';
║    Full Scale Deflection               │              │              │ ║';
║    Error at Mean                       │              │              │ ║';
║    % error of                          │              │              │ ║';
║                                        │              │              │ ║';
║    Correlation coefficient             │              │              │ ║';
╠════════════════════════════════════════╧══════════════╧══════════════╣ ║';
║                                                                       ║';
║                v - View Correlation Graph                             ║';
║                c - Change Scales of Correlation Graph                 ║';
║                                                                       ║';
║                q - Quit                                               ║';
╚═══════════════════════════════════════════════════════════════════════╝';
```

```
  END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITMENU3                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                         routine                                      ***}
{S**   CALL SEQUENCE      : INITMENU3(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                         ***}
{O**   OUTPUT PARAMETERS  : menu3 (type screenmenu)                      ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED     : Types                                        ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           : for details about screenmenu see TYPES       ***}
{*************************************************************************}

PROCEDURE initmenu3(VAR menu:screenmenu);

  BEGIN
       menu[1]:='
       menu[2]:='
       menu[3]:='
       menu[4]:='
       menu[5]:='
       menu[6]:='
       menu[7]:='
       menu[8]:='
       menu[9]:='
      menu[10]:='
      menu[11]:='
      menu[12]:='
      menu[13]:='
      menu[14]:='
      menu[15]:='
      menu[16]:='
      menu[17]:='
      menu[18]:='
      menu[19]:='
      menu[20]:='
      menu[21]:='
      menu[22]:='
      menu[23]:='
      menu[24]:='
```

```
╔═══════════════════════════════════════════════════════════════════════╗';
║                                                                       ║';
║                  Transfer Files                                       ║';
║                                                                       ║';
║                                                                       ║';
║                                                                       ║';
║                     MENU:                                             ║';
║                                                                       ║';
║                                                                       ║';
║                l - Load Correlation Data File                         ║';
║                                                                       ║';
║                s - Save Correlation Data File                         ║';
║                                                                       ║';
║                                                                       ║';
║                                                                       ║';
║                                                                       ║';
║                                                                       ║';
║                q - Quit                                               ║';
╚═══════════════════════════════════════════════════════════════════════╝';
```

```
  END;
```

```
{************************************************************************}
{N**   MODULE NAME        : INITMENU4                                    ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-    ***}
{***                        routine                                      ***}
{S**   CALL SEQUENCE      : INITMENU4(menu)                              ***}
{I**   INPUT PARAMETERS   : none                                        ***}
{O**   OUTPUT PARAMETERS  : menu4                                       ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : Types                                       ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : for details about screenmenu see TYPES      ***}
{************************************************************************}

PROCEDURE initmenu4(VAR menu:screenmenu);

  BEGIN
       menu[1]:='╔═══════════════════════════════════════════════════════════════════╗';
       menu[2]:='║                                                                   ║';
       menu[3]:='║              DATA PROCESSING SUBROUTINES                          ║';
       menu[4]:='║                                                                   ║';
       menu[5]:='╠═══════════════════════════════════╦═══════════════════════════════╣';
       menu[6]:='║                                   ║                               ║';
       menu[7]:='║          Options:                 ║   Correlation Results:        ║';
       menu[8]:='║                                   ║                               ║';
       menu[9]:='║                                   ║                               ║';
      menu[10]:='║  o - Offset of weightometer       ║      Y-Intercept:             ║';
      menu[11]:='║    =                              ║      Slope       :            ║';
      menu[12]:='║                                   ║                               ║';
      menu[13]:='║  z - Zero calibration             ║      Correlation              ║';
      menu[14]:='║    =                              ║      Coefficient:             ║';
      menu[15]:='║  s - Slope calibration            ║                               ║';
      menu[16]:='║    =                              ╠═══════════════════════════════╣';
      menu[17]:='║                                   ║                               ║';
      menu[18]:='║  f - Filter time constant         ║      TOTALS from              ║';
      menu[19]:='║    =                              ║                               ║';
      menu[20]:='╠═══════════════════════════════════╝                               ║';
      menu[21]:='║  c - Do Calculation               ║  Weightometer  :         tons ║';
      menu[22]:='║  g - Graphical Results            ║  New Technique :         tons ║';
      menu[23]:='║  q - Quit                         ║  % Error       :              ║';
      menu[24]:='╚═══════════════════════════════════╩═══════════════════════════════╝';

  END;

{************************************************************************}
{N**   MODULE NAME        : INITMENU41                                   ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-    ***}
{***                        routine                                      ***}
{S**   CALL SEQUENCE      : INITMENU41(menu)                             ***}
{I**   INPUT PARAMETERS   : none                                        ***}
{O**   OUTPUT PARAMETERS  : menu  (type screenmenu)                      ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : Types                                       ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : for details about screenmenu see TYPES      ***}
{************************************************************************}

PROCEDURE initmenu41(VAR menu:screenmenu);

  BEGIN
       menu[1]:='╔═══════════════════════════════════════════════════════════════════╗';
       menu[2]:='║                                                                   ║';
       menu[3]:='║              Graphical Results                                    ║';
       menu[4]:='║                                                                   ║';
       menu[5]:='║                                                                   ║';
       menu[6]:='║                                                                   ║';
       menu[7]:='║                        MENU:                                      ║';
       menu[8]:='║                                                                   ║';
       menu[9]:='║                                                                   ║';
      menu[10]:='║                                                                   ║';
      menu[11]:='║                t - Graph of Real Time Response                    ║';
      menu[12]:='║                                                                   ║';
      menu[13]:='║                c - Graph of Correlation Graph                     ║';
      menu[14]:='║                                                                   ║';
      menu[15]:='║                                                                   ║';
      menu[16]:='║                s - Change Scales of Real Time Response            ║';
      menu[17]:='║                                                                   ║';
      menu[18]:='║                a - Auto-scale Real Time and Correlation Graph     ║';
      menu[19]:='║                                                                   ║';
      menu[20]:='║                                                                   ║';
      menu[21]:='║                                                                   ║';
      menu[22]:='║                q - Quit                                           ║';
      menu[23]:='║                                                                   ║';
      menu[24]:='╚═══════════════════════════════════════════════════════════════════╝';

  END;

{************************************************************************}
{H**   REVISION HISTORY:                                              ***}
{***   VERSION      BY        DATE       COMMENTS                     ***}
{***     1.0        KCAD     28-02-90                                 ***}
{***                                                                  ***}
{***   UNITEND  MPCMD.PAS                                             ***}
{************************************************************************}
END.
```

C.3.16 Mfca.pas

```
{*************************************************************************}
{***   PROGRAM TO DETERMINE THE MASSFLOW ON A HORIZONTAL CONVEYOR BELT ***}
{***           FROM THE POWER AND SPEED RESPONSE IN AN ON-LINE FASHION ***}
{***           AND TO CORRELATE THIS THIS MASSFLOW FIGURE TO A KNOWN   ***}
{***           MASSFLOW FIGURE                                         ***}
{***   FILE: MFCA.PAS                                                  ***}
{*************************************************************************}

{*************************************************************************}
{N**   PROGRAM NAME      : MFCA                                        ***}
{***   ------------------------------------------------------------    ***}
{A**   DESCRIPTION       : Determines the massflow on a horizontal     ***}
{***                       conveyor by determing the kinetic energy    ***}
{***                       consumed by the system during a pulse per-  ***}
{***                       turbation and correlates this reading to a  ***}
{***                       known massflow figure exists                ***}
{S**   CALL SEQUENCE     : MAIN PROGRAM                                ***}
{I**   INPUT PARAMETERS  : n.a.                                       ***}
{O**   OUTPUT PARAMETERS : massflow figures                           ***}
{G**   GLOBAL VARIABLES  : none                                       ***}
{M**   MODULES CALLED    : Crt,Dos,Graph,Plotgraf,Types,Filehand,Detmf,***}
{***                       Datalog,Math,DT2801_4,Mfcad                 ***}
{E**   ERROR CONDITIONS  : none                                       ***}
{C**   COMMENTS          : none                                       ***}
{*************************************************************************}

PROGRAM mfca;

{$M 65000,0,655360}

   USES Crt,dos,graph,plotgraf,types,filehand,mfcad,detmf,datalog,math,dt2801_4;


   CONST
      pathdata='data\.cor';


   VAR
      mode,pulselength,outch,actmf,wait:integer;
      setpt:real;
      regdata:regressiondata;
      corrdata:correlationdata;
      respdata:responsedata;
      Csp:constant;
      Ct:time;
      Cm:massflowconst;
      menu1,menu2:screenmenu;
      option:char;
      dataname:str50;


{*************************************************************************}
{***                      PROCEDURES                                   ***}
{*************************************************************************}

{***=====================================================================***}
{***  change variables                                                 ***}
{***=====================================================================***}

PROCEDURE chgvars;

   VAR
      modechar,mfchar:string[10];
      option:char;

   BEGIN
      INITMENU2(menu2);
      repeat
         case mode of
            1:modechar:='VSD: Vdn=0';
            2:modechar:='VSD: Vdn>0';
            3:modechar:='SSR: Vdn=0';
            4:modechar:='SSR: Vdn>0';
         end;

         case actmf of
            0:mfchar:='BINMASS';
            1:mfchar:='NUCLEAR';
         end;
```

```
      clrscr;
      WRITEMENU(menu2);
      gotoXY(56,10);writeln(corrdata.nsamples);
      gotoXY(10,11);if actmf=1 then writeln('                                                    ');
      gotoXY(56,11);if actmf=0 then writeln(corrdata.nzero);
      gotoXY(56,12);writeln(wait);
      gotoXY(56,13);writeln(setpt:5:2);
      gotoXY(46,16);writeln(modechar);
      gotoXY(30,20);writeln(dataname);
      gotoXY(30,19);writeln(corrdata.title);
      gotoXY(50,15);writeln(mfchar);
      gotoXY(10,11);if actmf=1 then writeln('                                                    ');
      gotoXY(5,25);write('Option [q]: ');

      option:=readkey;
      if option=#13 then option:='q';

      case option of
         'n','N':READINT(56,10,5,corrdata.nsamples);
         'z','Z':READINT(56,11,5,corrdata.nzero);
         's','S':READINT(56,12,5,wait);
         'p','P':begin
                   READREAL(56,13,5,setpt);
                   DAC_WRITE(outch,round(setpt*2048/10));
                 end;
         'm','M':begin
                   mode:=mode+1;
                   if mode>4 then mode:=1;
                 end;
         'a','A':begin
                   actmf:=actmf+1;
                   if actmf>1 then actmf:=0;
                 end;
         'f','F':FILENAME(dataname,'');
         't','T':corrdata.title:=CHGTITLE('TITLE: ',corrdata.title);
         'c','C':CHKPULSETIMING(setpt,pulselength,outch,respdata)
      end;
   until (option='Q') or (option='q');

   case mode of
      1:begin
          outch:=0;   {Outputchannel 0 is dedicated to the VSD}
          respdata.step:=-10;
          pulselength:=Ct.stepup-Ct.stepdn;
        end;
      2:begin
          outch:=0;   {Outputchannel 0 is dedicated to the VSD}
          pulselength:=Ct.stepup-Ct.stepdn;
          HIGHLIGHT(0);
          gotoXY(25,25);write(' Size of Step [',respdata.step:4:1,']: ');
          READREAL(47,25,4,respdata.step);
          normvideo;
        end;
      3:begin
          outch:=1;   {Outputchannel 1 is dedicated to the SSR}
          respdata.step:=-10;
          pulselength:=Ct.stepup-Ct.stepdn;
        end;
      4:begin
          outch:=1;   {Outputchannel 1 is dedicated to the SSR}
          respdata.step:=-10;
          HIGHLIGHT(0);
          gotoXY(25,25);write(' Pulselength [',pulselength:3,']: ');
          READINT(45,25,3,pulselength);
          Ct.stepup:=Ct.stepdn+pulselength;
          Ct.ssdn:=Ct.stepup;
          normvideo;
        end;
   end;

END;
```

```
{***=============================================================================***}
{***  determine massflow on belt                                                 ***}
{***=============================================================================***}

PROCEDURE mftest;

   VAR
      Vav:real;
      i,j,k,display,tempdum,waitloops,code,pos,count:integer;
      mf:array[1..2] of real;
      key,ss:boolean;
      opt:char;
      countstr:string[2];


{-----------------------------------------------------------------------------}
{   checks if and which key has been pressed                                  }
{-----------------------------------------------------------------------------}

   PROCEDURE checkkey;

      BEGIN
         if keypressed then
            begin
               opt:=readkey;
               if (opt='q') or (opt='Q') then key:=true;
               if actmf=0 then
                  if (opt='z') or (opt='Z') then key:=true;
            end;
      END;


{-----------------------------------------------------------------------------}
{  checks if steady state has been reached                                    }
{-----------------------------------------------------------------------------}

   PROCEDURE steadystate;

      VAR
        j:integer;
        check:responsedata;
        dev:array[1..2] of real;
        mean:array[1..2] of integer;

      BEGIN
         INITVAR2(check);
         repeat
            PULSERESPONSE(setpt,0,outch,check);
            for j:=1 to 2 do MEANCALC(check,j,0,check.nsamples-1,mean[j],dev[j]);
            if (mean[1]>100) and (mean[2]>0) then
               if (dev[1]/mean[1]<0.1) and (dev[2]/mean[2]<0.1) then
                    ss:=true;
            checkkey;
         until (ss=true) or (key=true);
      END;


{-----------------------------------------------------------------------------}
{  copies the massflow samples into the data correlation array for a later    }
{         regression test                                                     }
{-----------------------------------------------------------------------------}

   PROCEDURE mfcorrelation;

      BEGIN
         with corrdata do
            begin
               if actmf=1 then data[1,count]:=MASSFLOWNUC(Cm,respdata)
               else data[1,count]:=MASSBIN(Cm,respdata);
               data[2,count]:=mf[1];
               data[3,count]:=mf[2];
               data[4,count]:=Vav;
            end;
      END;


{-----------------------------------------------------------------------------}
{  performs a pulse response of the system and determines the massflow        }
{           from the kinetic energy response                                  }
{-----------------------------------------------------------------------------}

   PROCEDURE doresponse;

      BEGIN
         sound(2000);delay(100);nosound;
         PULSERESPONSE(setpt,pulselength,outch,respdata);
         sound(4000);delay(100);nosound;
         DETMASSFLOW(Csp,Ct,respdata,mode,mf[1],mf[2],Vav);

         mfcorrelation;
         mf[2]:=(mf[2]-Csp.offset)/Csp.slope;
      END;
```

```
{-------------------------------------------------------------------------}
{  perform an initializing test to have a zero reference with the belt    }
{          being empty                                                    }
{-------------------------------------------------------------------------}

   PROCEDURE zeromftest(VAR start:integer);


      VAR
         i:integer;

      BEGIN
         clrscr;HIGHLIGHT(0);gotoXY(30,10);writeln(' ZERO-MASSFLOW TEST: ');normvideo;
         gotoXY(20,12);writeln(' Press <ENTER> continue if belt is empty ');
         readln;
         clrscr;HIGHLIGHT(0);gotoXY(25,10);writeln(' Performing Zero-Massflow Test ');normvideo;
         for i:=start+1 to start+corrdata.nzero do
            begin
               count:=count+1;
               doresponse;
            end;
      END;


{-------------------------------------------------------------------------}

   BEGIN
      count:=0;
      waitloops:=round(wait-((respdata.nsamples-1)/respdata.frequency)+1);
      if waitloops<0 then waitloops:=0;

      if actmf=0 then zeromftest(count);

      repeat
         DISPLAYINIT('MASSFLOW (t/hr)','ERROR :  +/-');

         repeat
            key:=false;
            ss:=false;

            steadystate;

            if (key=false) and (ss=true) then
               begin
                  count:=count+1;
                  doresponse;

                  DISPLAYNO(mf[2],Csp.error);

                  checkkey;
                  i:=0;
                  repeat
                     i:=i+1;
                     delay(1000);
                     checkkey;
                  until (i=1) or (key=true); {i=waitloops}
               end;

         until (count=corrdata.nsamples) or (key=true);

         closegraph;
         tempdum:=0;

         with corrdata do
            begin
               if count<nsamples then
                  begin
                     tempdum:=nsamples;
                     nsamples:=count;
                  end;

               if actmf=0 then
                  begin
                     zeromftest(count);
                     MASSFLOWBELT(Cm,corrdata);
                  end;


               if corrdata.nsamples>5 then REGRESS(corrdata,regdata);

               if tempdum<>0 then FILENAME(dataname,'');
               SAVECORRDATA(dataname,corrdata,regdata);

               if tempdum=0 then
                  begin
                     pos:=POSITION(dataname)-6;
                     countstr:=copy(dataname,pos,2);
                     val(countstr,count,code);
                     if count<9 then delete(dataname,pos+1,1)
                        else delete(dataname,pos,2);
                     count:=count+1;
                     str(count,countstr);
                     if count<10 then insert(countstr,dataname,pos+1)
                        else insert(countstr,dataname,pos);
                  end;
```

```
            count:=0;
            if actmf=0 then
               begin
                  for j:=1 to nzero do
                     for k:=1 to 3 do data[k,j]:=data[k,nsamples+j];
                  count:=nzero;
               end;
            if tempdum<>0 then nsamples:=tempdum;

         end;
      until (opt='q') or (opt='Q');
   END;


{*************************************************************************}
{***                    MAIN PROGRAM                                   ***}
{*************************************************************************}

BEGIN
   dataname:='';
   insert(pathdata,dataname,1);

   LOADCONST(Csp);
   LOADCONSTM(Cm);

   INITVAR1(Ct,corrdata,respdata,wait,actmf,mode,outch,setpt);
   pulselength:=Ct.stepup-Ct.stepdn;

   SET_UP;
   DAC_WRITE(outch,round(setpt*2048/10));

   INITMENU1(menu1);

   repeat
      clrscr;WRITEMENU(menu1);
      gotoXY(5,25);write('Option? ');
      option:=readkey;

      case option of
         'i','I':chgvars;
         'r','R':mftest;
         'c','C':mfcali(Csp,setpt,pulselength,outch,mode,Ct,respdata);
      end;
   until (option='q') or (option='Q');

{*************************************************************************}
{H**   REVISION HISTORY:                                                ***}
{***   VERSION      BY        DATE       COMMENTS                       ***}
{***     1.0       KCAD     01-03-90                                    ***}
{***                                                                    ***}
{***   FILEEND  MFCA.PAS                                                ***}
{*************************************************************************}
END.
```

C.3.17 Mfcad.pas

```
{************************************************************************}
{***   UNIT PROCEDURE LIBRARY CONTAINING DATAFILES TO BE USED IN       ***}
{***        MFCA                                                       ***}
{***   FILE: MFCAD.PAS                                                 ***}
{************************************************************************}

UNIT mfcad;

{************************************************************************}
                        INTERFACE
{************************************************************************}

   USES types;

   PROCEDURE initvar1(VAR Ct:time;VAR corrdata:correlationdata;VAR respdata:responsedata;VAR wait,actmf,
                      mode,outch:integer;VAR setpoint:real);
   PROCEDURE initvar2(VAR check:responsedata);
   PROCEDURE initmenu1(VAR menu:screenmenu);
   PROCEDURE initmenu2(VAR menu:screenmenu);


{************************************************************************}
                        IMPLEMENTATION
{************************************************************************}

{************************************************************************}
{N**   MODULE NAME       : INITVAR1                                     ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE         : initializes the variables used by the        ***}
{***                       Program MFCA                                 ***}
{S**   CALL SEQUENCE     : INITVAR1(Ct,corrdata,respdata,massact,mode   ***}
{***                               setpoint, corropt)                   ***}
{I**   INPUT PARAMETERS  : none                                         ***}
{O**   OUTPUT PARAMETERS : Ct - timing consatnts, values for correlat-  ***}
{***                       ion and response data record, initial set-   ***}
{***                       point of to be applied to plant, initiali-   ***}
{***                       various options                              ***}
{G**   GLOBAL VARIABLES  : none                                         ***}
{M**   MODULES CALLED    : Types                                        ***}
{E**   ERROR CONDITIONS  : none                                         ***}
{C**   COMMENTS          : for details about data records see TYPES     ***}
{************************************************************************}

PROCEDURE initvar1(VAR Ct:time;VAR corrdata:correlationdata;VAR respdata:responsedata;VAR wait,actmf,mode,
                   outch:integer;VAR setpoint:real);

   BEGIN
      Ct.asup:=0;
      Ct.ssdn:=200;
      Ct.stepdn:=100;
      Ct.stepup:=250;
      Ct.asend:=500;

      respdata.nsamples:=501;
      respdata.frequency:=50;
      respdata.nochannel:=3;
      respdata.step:=-10;
      respdata.whenstep:=Ct.stepdn;

      setpoint:=0.0;

      corrdata.nsamples:=500;
      corrdata.nzero:=3;
      corrdata.title:='Massflow Test: ';

      outch:=0;
      wait:=15;
      mode:=1;          {i.e VSD,Vdn=0}
      actmf:=1;         {0 - actual massflow is obtained from Hopper bin}
                        {1 - actual massflow obtained from nuclear weightometer}
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITVAR2                                      ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the variables used by the         ***}
{***                        Program MFCA                                  ***}
{S**   CALL SEQUENCE      : INITVAR2(check)                               ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : variables for check response data record      ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about data record see TYPES       ***}
{*************************************************************************}

PROCEDURE initvar2(VAR check:responsedata);

   BEGIN
      check.nsamples:=501;
      check.frequency:=50;
      check.nochannel:=3;
      check.step:=0;
      check.whenstep:=0;
   END;

{*************************************************************************}
{N**   MODULE NAME        : INITMENU1                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                        routine                                       ***}
{S**   CALL SEQUENCE      : INITMENU1(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : menu1 (type screenmenu)                       ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about screenmenu see TYPES        ***}
{*************************************************************************}

PROCEDURE initmenu1(VAR menu:screenmenu);

  BEGIN
       menu[1]:='╔═══════════════════════════════════════════════════════════════════╗';
       menu[2]:='║                                                                   ║';
       menu[3]:='║            MASSFLOW CORRELATION PACKAGE                           ║';
       menu[4]:='║                                                                   ║';
       menu[5]:='║                                                                   ║';
       menu[6]:='║                                                                   ║';
       menu[7]:='║                        MENU:                                      ║';
       menu[8]:='║                                                                   ║';
       menu[9]:='║                                                                   ║';
      menu[10]:='║                                                                   ║';
      menu[11]:='║                    i - Initialize Variables                       ║';
      menu[12]:='║                                                                   ║';
      menu[13]:='║                    c - Calibrate Massflow Constants               ║';
      menu[14]:='║                                                                   ║';
      menu[15]:='║                    r - Run Massflow Test                          ║';
      menu[16]:='║                                                                   ║';
      menu[17]:='║                                                                   ║';
      menu[18]:='║                                                                   ║';
      menu[19]:='║                                                                   ║';
      menu[20]:='║                                                                   ║';
      menu[21]:='║                                                                   ║';
      menu[22]:='║                    q - Quit                                       ║';
      menu[23]:='║                                                                   ║';
      menu[24]:='╚═══════════════════════════════════════════════════════════════════╝';

  END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITMENU2                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : Initializes the Menu defined in this Sub-     ***}
{***                        routine                                       ***}
{S**   CALL SEQUENCE      : INITMENU2(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : menu1 (type screenmenu)                       ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about screenmenu see TYPES        ***}
{*************************************************************************}

PROCEDURE initmenu2(VAR menu:screenmenu);

  BEGIN
       menu[1]:='╔═══════════════════════════════════════════════════════════════════════╗';
       menu[2]:='║                                                                       ║';
       menu[3]:='║              INITIALIZE VARIABLES                                     ║';
       menu[4]:='║                                                                       ║';
       menu[5]:='║                                                                       ║';
       menu[6]:='║                                MENU:                                  ║';
       menu[7]:='║                                                                       ║';
       menu[8]:='║                                                                       ║';
       menu[9]:='║                                                                       ║';
      menu[10]:='║          n - No of Massflow Samples per File   :                      ║';
      menu[11]:='║          z - No of Initial Zero-Massf. Samples :                      ║';
      menu[12]:='║          s - Sampling Time [sec]               :                      ║';
      menu[13]:='║          p - Steady State Setpoint             :                      ║';
      menu[14]:='║                                                                       ║';
      menu[15]:='║          a - Actual Massflow obtained from                            ║';
      menu[16]:='║          m - Mode of Operation      :                                 ║';
      menu[17]:='║                                                                       ║';
      menu[18]:='║                                                                       ║';
      menu[19]:='║          t - Title    :                                               ║';
      menu[20]:='║          f - Filename :                                               ║';
      menu[21]:='║                                                                       ║';
      menu[22]:='║                                                                       ║';
      menu[23]:='║     c - Calibrate Timing of datalogging Procedure      q - Quit       ║';
      menu[24]:='╚═══════════════════════════════════════════════════════════════════════╝';
  END;

{*************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0       KCAD     28-02-90                                     ***}
{***                                                                     ***}
{***   UNITEND  MFCAD.PAS                                                ***}
{*************************************************************************}
END.
```

C.3.18 Mf.pas

```
{*************************************************************************}
{***   PROGRAM TO DETERMINE THE MASSFLOW ON A HORIZONTAL CONVEYOR BELT ***}
{***          FROM THE ENERGY AND SPEED RESPONSE IN AN ON-LINE FASHION***}
{***   FILE: MF.PAS                                                    ***}
{*************************************************************************}

{*************************************************************************}
{N**   PROGRAM NAME      : MF                                           ***}
{***   ------------------------------------------------------------   ***}
{A**   DESCRIPTION       : Determines the massflow on a horizontal      ***}
{***                       conveyor by determing the kinetic energy     ***}
{***                       consumed by the system during a pulse per-   ***}
{***                       turbation.                                   ***}
{S**   CALL SEQUENCE     : MAIN PROGRAM                                 ***}
{I**   INPUT PARAMETERS  : n.a.                                         ***}
{O**   OUTPUT PARAMETERS : massflow figures                            ***}
{G**   GLOBAL VARIABLES  : none                                         ***}
{M**   MODULES CALLED    : Crt,Dos,Graph,Plotgraf,Types,Filehand,Detmf,***}
{***                       Datalog,Math,DT2801_4,Mfd                    ***}
{E**   ERROR CONDITIONS  : none                                         ***}
{C**   COMMENTS          : none                                         ***}
{*************************************************************************}

PROGRAM mf;

{$M 65000,0,655360}

   USES Crt,dos,graph,plotgraf,types,filehand,mfd,detmf,datalog,math,dt2801_4;


   CONST
      pathdata='data\.dam';


   VAR
      mode,pulselength,outch,wait:integer;
      step,setpt:real;
      respdata:responsedata;
      mfdata:massflowdata;
      Csp:constant;
      Ct:time;
      menu1,menu2:screenmenu;
      option:char;
      dataname:str50;


{*************************************************************************}
{***                     PROCEDURES                                    ***}
{*************************************************************************}

{***=====================================================================***}
{***  change variables                                                 ***}
{***=====================================================================***}

PROCEDURE chgvars;

   VAR
      modechar,masschar:string[10];
      option:char;

   BEGIN
      INITMENU2(menu2);
      repeat
         case mode of
            1:modechar:='VSD: Vdn=0';
            2:modechar:='VSD: Vdn>0';
            3:modechar:='SSR: Vdn=0';
            4:modechar:='SSR: Vdn>0';
         end;

         clrscr;
         WRITEMENU(menu2);
         gotoXY(56,10);writeln(mfdata.nsamples);
         gotoXY(56,12);writeln(wait);
         gotoXY(56,13);writeln(setpt:5:2);
         gotoXY(46,16);writeln(modechar);
         gotoXY(30,20);writeln(dataname);
         gotoXY(30,19);writeln(mfdata.title);
         gotoXY(5,25);write('Option [q]: ');
```

```
        option:=readkey;
        if option=#13 then option:='q';

        case option of
           'n','N':READINT(56,10,5,mfdata.nsamples);
           's','S':READINT(56,12,5,wait);
           'p','P':begin
                     READREAL(56,13,5,setpt);
                     DAC_WRITE(outch,round(setpt*2048/10));
                   end;
           'm','M':begin
                     mode:=mode+1;
                     if mode>4 then mode:=1;
                     SAVELOOPDATA(0);
                   end;
           'f','F':FILENAME(dataname,'');
           't','T':mfdata.title:=CHGTITLE('TITLE: ',mfdata.title);
           'c','C':CHKPULSETIMING(setpt,pulselength,outch,respdata)
        end;
     until (option='Q') or (option='q');

     case mode of
        1:begin
             outch:=0;   {Outputchannel 0 is dedicated to the VSD}
             step:=-10;
             pulselength:=Ct.stepup-Ct.stepdn;
          end;
        2:begin
             outch:=0;   {Outputchannel 0 is dedicated to the VSD}
             pulselength:=Ct.stepup-Ct.stepdn;
             HIGHLIGHT(0);
             gotoXY(25,19);write(' Size of Step [',step:4:1,']: ');
             READREAL(47,19,4,step);
             normvideo;
          end;
        3:begin
             outch:=1;   {Outputchannel 1 is dedicated to the SSR}
             step:=-10;
             pulselength:=Ct.stepup-Ct.stepdn;
          end;
        4:begin
             outch:=1;   {Outputchannel 1 is dedicated to the SSR}
             step:=-10;
             HIGHLIGHT(0);
             gotoXY(25,19);write(' Pulselength [',pulselength:3,']: ');
             READINT(45,19,3,pulselength);
             Ct.stepup:=Ct.stepdn+pulselength;
             Ct.ssdn:=Ct.stepup;
             normvideo;
          end;
     end;

  END;


{***=============================================================***}
{***  determine massflow on belt                                 ***}
{***=============================================================***}

PROCEDURE masstest;

  VAR
     Vav,zeromf:real;
     i,j,k,tempdum,waitloops,code,count,pos:integer;
     mf,dum:real;
     key,ss:boolean;
     opt:char;
     countstr:string[2];


{-------------------------------------------------------------------}
{   checks if and which key has been pressed                        }
{-------------------------------------------------------------------}

  PROCEDURE checkkey;

     BEGIN
        if keypressed then
           begin
              opt:=readkey;
              if (opt='q') or (opt='Q') then key:=true;
           end;
     END;
```

```
{-------------------------------------------------------------------------}
{  checks if steady state has been reached                                }
{-------------------------------------------------------------------------}

   PROCEDURE steadystate;

      VAR
         j:integer;
         check:responsedata;
         dev:array[1..2] of real;
         mean:array[1..2] of integer;

      BEGIN
         INITVAR2(check);
         repeat
            PULSERESPONSE(setpt,0,outch,check);
            for j:=1 to 2 do MEANCALC(check,j,0,check.nsamples-1,mean[j],dev[j]);
            if (mean[1]>100) and (mean[2]>0) then
               if (dev[1]/mean[1]<0.1) and (dev[2]/mean[2]<0.1) then
                  ss:=true;
            checkkey;
         until (ss=true) or (key=true);
      END;


{-------------------------------------------------------------------------}
{  performs a pulse response of the system and determines the massflow    }
{           from the kinetic energy response                              }
{-------------------------------------------------------------------------}

   PROCEDURE doresponse;


      BEGIN
         sound(2000);delay(100);nosound;
         PULSERESPONSE(setpt,pulselength,outch,respdata);
         sound(4000);delay(100);nosound;
         with mfdata do
            gettime(time[count].hour,time[count].min,time[count].sec,time[count].sec100);

         DETMASSFLOW(Csp,Ct,respdata,mode,dum,mf,Vav);

         mfdata.data[count]:=(mf-Csp.offset)/Csp.slope;
         if mfdata.data[count]<0 then mfdata.data[count]:=0;
      END;


{-------------------------------------------------------------------------}

   BEGIN
      count:=0;
      waitloops:=round(wait-((respdata.nsamples-1)/respdata.frequency)+1);
      if waitloops<0 then waitloops:=0;

      repeat
         DISPLAYINIT('MASSFLOW [t/hr]','ERROR :  +/-');

         repeat
            key:=false;
            ss:=false;

           steadystate;

            if (key=false) and (ss=true) then
               begin
                  count:=count+1;
                  doresponse;

                  DISPLAYNO(mfdata.data[count],mfdata.error);

                  checkkey;
                  i:=0;
                  repeat
                     i:=i+1;
                     delay(1000);
                     checkkey;
                  until (key=true) or (i=1); {i=waitloops};
               end;

         until (count=mfdata.nsamples) or (key=true);

         closegraph;

         if count<mfdata.nsamples then
            begin
               tempdum:=mfdata.nsamples;
               mfdata.nsamples:=count;
               FILENAME(dataname,'');
               SAVEMPDATA(dataname,mfdata);
               mfdata.nsamples:=tempdum;
            end
         else
            begin
               SAVEMPDATA(dataname,mfdata);
               pos:=POSITION(dataname)-6;
               countstr:=copy(dataname,pos,2);
               val(countstr,count,code);
```

```
            if count<9 then delete(dataname,pos+1,1)
               else delete(dataname,pos,2);
            count:=count+1;
            str(count,countstr);
            if count<10 then insert(countstr,dataname,pos+1)
               else insert(countstr,dataname,pos);
         end;

      count:=0;
   until (key=true);
END;


{*************************************************************************}
{***                    MAIN PROGRAM                                   ***}
{*************************************************************************}

BEGIN
   dataname:='';
   insert(pathdata,dataname,1);

   LOADCONST(Csp);

   INITVAR1(Ct,respdata,mfdata,wait,mode,outch,setpt);
   pulselength:=Ct.stepup-Ct.stepdn;

   SET_UP;
   DAC_WRITE(outch,round(setpt*2048/10));

   INITMENU1(menu1);

   repeat
      clrscr;
      WRITEMENU(menu1);
      gotoXY(5,25);write('Option? ');

      option:=readkey;

      case option of
         'i','I':chgvars;
         'r','R':masstest;
         'm','M':begin
                    FILENAME(dataname,'');
                    LOADMFDATA(dataname,mfdata);
                    MFSTATS(mfdata);
                 end;
         'c','C':MFCALI(Csp,setpt,pulselength,outch,mode,Ct,respdata);
      end;
   until (option='q') or (option='Q');

{*************************************************************************}
{H**   REVISION HISTORY:                                                ***}
{***   VERSION      BY        DATE       COMMENTS                       ***}
{***     1.0       KCAD     06-04-90                                    ***}
{***                                                                     ***}
{***   FILEEND  MF.PAS                                                  ***}
{*************************************************************************}
END.
```

C.3.19 Mfd.pas

```
{*************************************************************************}
{***   UNIT PROCEDURE LIBRARY CONTAINING DATAFILES TO BE USED IN       ***}
{***        MF                                                         ***}
{***   FILE: MFD.PAS                                                   ***}
{*************************************************************************}

UNIT mfd;

{*************************************************************************}
                        INTERFACE
{*************************************************************************}

   USES types;

   PROCEDURE initvar1(VAR Ct:time;VAR respdata:responsedata;VAR mfdata:massflowdata;
                      VAR wait,mode,outch:integer;VAR setpoint:real);
   PROCEDURE initvar2(VAR check:responsedata);
   PROCEDURE initmenu1(VAR menu:screenmenu);
   PROCEDURE initmenu2(VAR menu:screenmenu);


{*************************************************************************}
                        IMPLEMENTATION
{*************************************************************************}

{*************************************************************************}
{N**   MODULE NAME       : INITVAR1                                     ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE         : initializes the variables used by the        ***}
{***                       Program MF                                   ***}
{S**   CALL SEQUENCE     : INITVAR1(Ct,corrdata,respdata,massact,mode   ***}
{***                              setpoint, corropt)                    ***}
{I**   INPUT PARAMETERS  : none                                         ***}
{O**   OUTPUT PARAMETERS : Ct - timing consatnts, values for correlat-  ***}
{***                       ion and response data record, initial set-   ***}
{***                       point of to be applied to plant, initiali-   ***}
{***                       various options                              ***}
{G**   GLOBAL VARIABLES  : none                                         ***}
{M**   MODULES CALLED    : Types                                        ***}
{E**   ERROR CONDITIONS  : none                                         ***}
{C**   COMMENTS          : for details about data records see TYPES     ***}
{*************************************************************************}

PROCEDURE initvar1(VAR Ct:time;VAR respdata:responsedata;VAR mfdata:massflowdata;
                   VAR wait,mode,outch:integer;VAR setpoint:real);

   BEGIN
      Ct.ssup:=0;
      Ct.ssdn:=200;
      Ct.stepdn:=100;
      Ct.stepup:=250;
      Ct.ssend:=500;

      respdata.nsamples:=501;
      respdata.frequency:=50;
      respdata.nochannel:=3;
      respdata.step:=-10;
      respdata.whenstep:=Ct.stepdn;

      mfdata.nsamples:=500;
      mfdata.title:='Massflow Meter: ';
      setpoint:=0;

      outch:=0;
      wait:=15;
      mode:=1;          {i.e VSD,Vdn=0}
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITVAR2                                      ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the variables used by the         ***}
{***                         Program MF                                   ***}
{S**   CALL SEQUENCE      : INITVAR2(check)                               ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : variables for check response data record      ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about data record see TYPES       ***}
{*************************************************************************}

PROCEDURE initvar2(VAR check:responsedata);

   BEGIN
      check.nsamples:=501;
      check.frequency:=50;
      check.nochannel:=3;
      check.step:=0;
      check.whenstep:=0;
   END;


{*************************************************************************}
{N**   MODULE NAME        : INITMENU1                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                         routine                                      ***}
{S**   CALL SEQUENCE      : INITMENU1(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : menu1 (type screenmenu)                       ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about screenmenu see TYPES        ***}
{*************************************************************************}

PROCEDURE initmenu1(VAR menu:screenmenu);

  BEGIN
      menu[1]:='╔═══════════════════════════════════════════════════════════╗';
      menu[2]:='║                                                           ║';
      menu[3]:='║          MASSFLOW ESTIMATION PACKAGE                      ║';
      menu[4]:='║                                                           ║';
      menu[5]:='║                                                           ║';
      menu[6]:='║                                                           ║';
      menu[7]:='║                    MENU:                                  ║';
      menu[8]:='║                                                           ║';
      menu[9]:='║                                                           ║';
     menu[10]:='║                                                           ║';
     menu[11]:='║                    i - Initialize Variables               ║';
     menu[12]:='║                                                           ║';
     menu[13]:='║                    m - Massflow Statistics                ║';
     menu[14]:='║                                                           ║';
     menu[15]:='║                    c - Calibrate massflow constants       ║';
     menu[16]:='║                                                           ║';
     menu[17]:='║                                                           ║';
     menu[18]:='║                    r - Run Massflow meter                 ║';
     menu[19]:='║                                                           ║';
     menu[20]:='║                                                           ║';
     menu[21]:='║                                                           ║';
     menu[22]:='║                    q - Quit                               ║';
     menu[23]:='║                                                           ║';
     menu[24]:='╚═══════════════════════════════════════════════════════════╝';

  END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITMENU2                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : Initializes the Menu defined in this Sub-     ***}
{***                        routine                                       ***}
{S**   CALL SEQUENCE      : INITMENU2(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : menu1 (type screenmenu)                       ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about screenmenu see TYPES        ***}
{*************************************************************************}

PROCEDURE initmenu2(VAR menu:screenmenu);

  BEGIN
      menu[1]:='╔═══════════════════════════════════════════════════════════════════╗';
      menu[2]:='║                                                                   ║';
      menu[3]:='║            INITIALIZE VARIABLES                                   ║';
      menu[4]:='║                                                                   ║';
      menu[5]:='║                                                                   ║';
      menu[6]:='║                                                                   ║';
      menu[7]:='║                          MENU:                                    ║';
      menu[8]:='║                                                                   ║';
      menu[9]:='║                                                                   ║';
     menu[10]:='║          n - No of Massflow Samples per File   :                  ║';
     menu[11]:='║                                                                   ║';
     menu[12]:='║          s - Sampling Time [sec]               :                  ║';
     menu[13]:='║          p - Steady State Setpoint             :                  ║';
     menu[14]:='║                                                                   ║';
     menu[15]:='║                                                                   ║';
     menu[16]:='║          m - Mode of Operation      :                             ║';
     menu[17]:='║                                                                   ║';
     menu[18]:='║                                                                   ║';
     menu[19]:='║          t - Title    :                                           ║';
     menu[20]:='║          f - Filename :                                           ║';
     menu[21]:='║                                                                   ║';
     menu[22]:='║                                                                   ║';
     menu[23]:='║     c - Calibrate Timing of datalogging Procedure      q - Quit   ║';
     menu[24]:='╚═══════════════════════════════════════════════════════════════════╝';

  END;

{*************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0        KCAD     28-02-90                                    ***}
{***                                                                     ***}
{***   UNITEND  MFD.PAS                                                  ***}
{*************************************************************************}
END.
```

C.3.20 Estimate.pas

```
{*************************************************************************}
{***   UNIT PROCEDURE LIBRARY TO DETERMINE THE TRANSFER FUNCTION OF A  ***}
{***        MEASURED RESPONSE                                          ***}
{***   FILE: ESTIMATE.PAS                                             ***}
{*************************************************************************}

UNIT estimate;

{*************************************************************************}
                        INTERFACE
{*************************************************************************}

   USES
      crt,math,model,types;

   PROCEDURE set_up_u1(data:responsedata);
   PROCEDURE set_up_y(c:integer;data:responsedata);
   PROCEDURE set_up_u2r(p:parameterdata;data:responsedata);
   PROCEDURE set_up_u2c(p:parameterdata;data:responsedata);
   PROCEDURE lse(VAR B:vectorsml);
   PROCEDURE lse2(VAR p:parameterdata;VAR data:responsedata);
   PROCEDURE nelm(eps:real;VAR para:parameterdata;VAR data:responsedata);


{*************************************************************************}
                        IMPLEMENTATION
{*************************************************************************}

{*************************************************************************}
{N**   MODULE NAME       : SET_UP_U1                                    ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE         : sets up the U matrix used for the Least      ***}
{***                       Squares parameter estimation technique to    ***}
{***                       find the poles of the transfer function      ***}
{S**   CALL SEQUENCE     : SET_UP_U1(data)                              ***}
{I**   INPUT PARAMETERS  : measured response data record, limits of    ***}
{***                       response                                     ***}
{O**   OUTPUT PARAMETERS : U matrix (type matrixbig)                    ***}
{G**   GLOBAL VARIABLES  : limits of response, U matrix                 ***}
{M**   MODULES CALLED    : Types                                        ***}
{E**   ERROR CONDITIONS  : none                                         ***}
{C**   COMMENTS          : for details about response data, limits and ***}
{***                       matrixbig see TYPES                          ***}
{*************************************************************************}

PROCEDURE set_up_u1(data:responsedata);

   VAR
      i,j,m,max,min:integer;

   BEGIN
      min:=round(limits.tmin * data.frequency);
      max:=round(limits.tmax * data.frequency);
      m:=model1.order;
      if min<m+1 then min:=m+1;

      for i:=min to max-1 do
         begin
            U.a[i-m,1]:=1;
            for j:=1 to m do U.a[i-m,j+1]:=-data.data[1,i-j];
         end;
      U.rows:=max-min;
      U.cols:=m+1;
   END;


{*************************************************************************}
{N**   MODULE NAME       : SET_UP_Y                                     ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE         : sets up the Y vector used for the Least      ***}
{***                       Squares parameter estimation technique to    ***}
{***                       find the poles (c:=1) or the zeroes (c:=2)   ***}
{***                       of the transfer function                     ***}
{S**   CALL SEQUENCE     : SET_UP_Y(c,data)                             ***}
{I**   INPUT PARAMETERS  : c indicates case (1 = poles; 2 = zeroes);   ***}
{***                       measured response data record, limits of    ***}
{***                       response                                     ***}
{O**   OUTPUT PARAMETERS : Y vector (type vectorbig)                    ***}
{G**   GLOBAL VARIABLES  : limits of response, Y vector                 ***}
{M**   MODULES CALLED    : Types                                        ***}
{E**   ERROR CONDITIONS  : none                                         ***}
{C**   COMMENTS          : for details about response data, limits and ***}
{***                       vectorbig see TYPES                          ***}
{*************************************************************************}

PROCEDURE set_up_y(c:integer;data:responsedata);
```

```
   VAR
      i,j,m,min,max:integer;

   BEGIN
      min:=round(limits.tmin * data.frequency);
      max:=round(limits.tmax * data.frequency);
      m:=modell.order;

      if (c=1) and (min<m+1) then c:=m+1 else c:=min;
      for i:=c to max-1 do Y.v[i+1-c]:=data.data[1,i];
      Y.elements:=max-c;
   END;


{*************************************************************************}
{N**   MODULE NAME        : SET_UP_U2R                                    ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : sets up the U matrix used for the Least       ***}
{***                        Squares parameter estimation technique to     ***}
{***                        find the zeroes of the transfer function      ***}
{***                        with real poles                               ***}
{S**   CALL SEQUENCE      : SET_UP_U2R(parameter,data)                    ***}
{I**   INPUT PARAMETERS   : measured data record, parameter data record  ***}
{***                        containing the poles of the transfer          ***}
{***                        function, limits of response                  ***}
{O**   OUTPUT PARAMETERS  : U matrix (type matrixbig)                     ***}
{G**   GLOBAL VARIABLES   : limits of response, U matrix                  ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about response data, parameter    ***}
{***                        data, limits and matrixbig see TYPES          ***}
{*************************************************************************}

PROCEDURE set_up_u2r(p:parameterdata;data:responsedata);

   VAR
      i,j,min,max:integer;
      t:real;

   BEGIN
      min:=round(limits.tmin * data.frequency);
      max:=round(limits.tmax * data.frequency);
      for i:=min to max-1 do
         begin
            t:=i/data.frequency;
            U.a[i+1,1]:=1;
            for j:=1 to modell.order do U.a[i+1,j+1]:=exp(-p.para[3+j]*t);
         end;
      U.rows:=max-min;
      U.cols:=modell.order+1;
   END;


{*************************************************************************}
{N**   MODULE NAME        : SET_UP_U2C                                    ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : sets up the U matrix used for the Least       ***}
{***                        Squares parameter estimation technique to     ***}
{***                        find the zeroes of the transfer function      ***}
{***                        with complex poles                            ***}
{S**   CALL SEQUENCE      : SET_UP_U2C(parameter,data)                    ***}
{I**   INPUT PARAMETERS   : measured response data record, parameter     ***}
{***                        data record containing the poles of the       ***}
{***                        transfer function, limits of response         ***}
{O**   OUTPUT PARAMETERS  : U matrix (type matrixbig)                     ***}
{G**   GLOBAL VARIABLES   : limits of response, U matrix                  ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   :                                              ***}
{C**   COMMENTS           : for details about response data, parameter    ***}
{***                        data, limits and matrixbig see TYPES          ***}
{*************************************************************************}

PROCEDURE set_up_u2c(p:parameterdata;data:responsedata);

   VAR
      i,j,min,max:integer;
      t,factor:real;

   BEGIN
      min:=round(limits.tmin * data.frequency);
      max:=round(limits.tmax * data.frequency);
      for i:=min to max-1 do
         begin
            t:=i/data.frequency;

            factor:=exp(-p.para[4]*t);
            U.a[i+1,1]:=1;
            U.a[i+1,2]:=factor*cos(p.para[5]*t);
            U.a[i+1,3]:=factor*sin(p.para[5]*t);
            if modell.order=3 then U.a[4,i+1]:=exp(-p.para[6]*t);
         end;
      U.rows:=max-min;
      U.cols:=modell.order+1;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : LSE                                          ***}
{***   ------------------------------------------------------------     ***}
{A**   PROCEDURE          : performs the Least Squares estimation algo- ***}
{***                        rithm by solving the equation Ut*U|Ut*Y      ***}
{S**   CALL SEQUENCE      : LSE(B)                                       ***}
{I**   INPUT PARAMETERS   : vector B, matrix U (matrixbig) and vector Y ***}
{***                        (vectorbig)                                  ***}
{O**   OUTPUT PARAMETERS  : solution Matrix                              ***}
{G**   GLOBAL VARIABLES   : matrix U and vector Y                        ***}
{M**   MODULES CALLED     : Math, Types                                  ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           : for details about matrixbig and vectorbig    ***}
{***                        TYPES                                        ***}
{*************************************************************************}

PROCEDURE lse(VAR B:vectorsml);

   VAR
      A:matrixsml;
      i,j:integer;

   BEGIN
      UtU(A);
      UtY(B);
      GAUSS_1(A,B);
   END;


{*************************************************************************}
{N**   MODULE NAME        : LSE2                                         ***}
{***   ------------------------------------------------------------     ***}
{A**   PROCEDURE          : fits a transfer function to a measured       ***}
{***                        response in the s-domain using the 2 Stage   ***}
{***                        least squares estimation algorithm           ***}
{S**   CALL SEQUENCE      : LSE2(parameter,data)                         ***}
{I**   INPUT PARAMETERS   : measured response data record, modell data   ***}
{***                        record                                       ***}
{O**   OUTPUT PARAMETERS  : coefficients of transfer function           ***}
{G**   GLOBAL VARIABLES   : modell (type modeldata)                      ***}
{M**   MODULES CALLED     : Crt,Math, Types                              ***}
{E**   ERROR CONDITIONS   : if poles in the z-plane are too close to the***}
{***                        |z|=1 contour                                ***}
{C**   COMMENTS           : for details about response data, parameter   ***}
{***                        data and modeldata see TYPES                 ***}
{*************************************************************************}

PROCEDURE lse2(VAR p:parameterdata;VAR data:responsedata);

   VAR
      B:vectorsml;
      discriminant:real;
      i,j:integer;

   BEGIN
      clrscr;gotoXY(16,6);writeln('REGRESSING USING THE LEAST SQUARES ESTIMATOR');
      SET_UP_U1(data);
      SET_UP_Y(1,data);
      LSE(B);

      gotoXY(10,12);writeln('Pole Position - Distance from Origin in the z-plane:  ',sqrt(B.v[3]):4:2);
      if abs(abs(sqrt(B.v[3]))-1)<0.05 then
         begin
            HIGHLIGHT(1);gotoXY(25,15);writeln('!!! MODEL NOT ACCURATE !!!');
            HIGHLIGHT(0);gotoXY(11,17);
            writeln('Poles in the Z-Plane are close to the |Z| = 1 contour');
         end;

      case modell.order of
         1:begin    {1st order Response}
              p.para[4]:=-ln(-B.v[2])*data.frequency;   {pole}
              SET_UP_U2r(p,data);
              SET_UP_Y(2,data);
              LSE(B);
              p.para[3]:=-B.v[2];    {Gain}
           end;

         2:begin    {2nd order Response}
              discriminant:=sqr(B.v[2]/2) - B.v[3];

              if discriminant > 0 then     {i.e. real poles}
                 begin
                    p.para[4]:=-ln(-B.v[2]/2+sqrt(discriminant)) * data.frequency;{pole 1}
                    p.para[5]:=-ln(-B.v[2]/2-sqrt(discriminant)) * data.frequency;{pole 2}
                    SET_UP_U2r(p,data);
                    SET_UP_Y(2,data);
                    LSE(B);
                    p.para[3]:=-(B.v[2]+B.v[3]);    {gain}
                    p.para[6]:=(B.v[2]*p.para[5]+B.v[3]*p.para[4]) / p.para[3];     {zero}
                    modell.typ:=1;
                 end
```

```
            else      {i.e. complex poles}
               begin
                  p.para[4]:=-ln(sqrt(B.v[3])) * data.frequency;{real part of pole pair}
                  p.para[5]:=arctan(-sqrt(-discriminant)*2 / B.v[2]) * data.frequency;
                                                          {imaginary part of pole pair}
                  SET_UP_U2c(p,data);
                  SET_UP_Y(2,data);
                  LSE(B);
                  p.para[3]:=B.v[1];
                  p.para[6]:=(sqr(p.para[4])+sqr(p.para[5]))*B.v[2]/(p.para[5]*B.v[3]-B.v[2]*p.para[4]);
                  model1.typ:=2;
               end;
         end;
      end;
      DETRESP(p,data);
      normvideo;
   END;

{*************************************************************************}
{N**   MODULE NAME       : NELM                                           ***}
{***   ------------------------------------------------------------     ***}
{A**   PROCEDURE         : Nelm is a non-linear regression package,       ***}
{***                       which determines the parameters of a trans-   ***}
{***                       fer function                                  ***}
{S**   CALL SEQUENCE     : NELM(epsilon,para,data)                        ***}
{I**   INPUT PARAMETERS  : parameter data record containing initial      ***}
{***                       estimates for the parameters of the transfer***}
{***                       function, epsilon = required accuracy limit,***}
{***                       measured data record, limits of response      ***}
{O**   OUTPUT PARAMETERS : updated parameter data record, updated        ***}
{***                       modelled response                             ***}
{G**   GLOBAL VARIABLES  : limits of response                            ***}
{M**   MODULES CALLED    : Crt, Types, Model                              ***}
{E**   ERROR CONDITIONS  : none                                          ***}
{C**   COMMENTS          : - for more information about response data,  ***}
{***                         limits and parameter data see TYPES         ***}
{***                       - to quit from NELM press <ENTER>             ***}
{*************************************************************************}

PROCEDURE nelm(eps:real;VAR para:parameterdata;VAR data:responsedata);

   VAR
      f,fs,f0,a0,s:real;

      k,ix,i,j,k1,nn,n1,n,np,xn,ih,is,k0,il,kev:integer;
      h:array[1..10] of real;
      fp:array[1..22] of real;
      p:array[1..120] of real;
      xs:array[1..10] of real;
      x,xd:parameterdata;
      loopend,endd:char;


{-------------------------------------------------------------------------}
{***   SUBPROCEDURE INNERLOOP                                          ***}
{-------------------------------------------------------------------------}

   PROCEDURE innerloop;

      VAR
         i:integer;

      BEGIN

         if ih=1 then is:=2
                 else is:=1;

         for i:=1 to n1 do
             if i<>ih then
                if fp[i]>fp[is] then is:=i;

         k:=(ih-1)*n+1;k0:=nn+1;
         for i:=1 to n do
            begin
               x.para[i]:=2*p[k0]-p[k];
               k:=k+1;
               k0:=k0+1;
            end;

         k:=k-n;
         COPYPARA(para,x);
         DETRESP(para,data);
         f:=SUMERRSQR(para,data);

         if f<fp[il] then
            begin
               k0:=nn+1;
               for i:=1 to n do
                  begin
                     xs[i]:=2*x.para[i]-p[k0];
                     k0:=k0+1;
                  end;
```

```
            for i:=1 to x.no do xd.para[i]:=xs[i];
            COPYPARA(para,xd);
            DETRESP(para,data);
            fs:=SUMERRSQR(para,data);

            if fs<fp[il] then
               begin
                  for i:=1 to n do
                     begin
                        p[k]:=xs[i];
                        k:=k+1;
                     end;
                  fp[ih]:=fs;
                  il:=ih;ih:=is;
               end
            else
               begin
                  il:=ih;ih:=is;fp[il]:=f;
                  for i:=1 to n do
                     begin
                        p[k]:=x.para[i];
                        k:=k+1;
                     end;
               end;
         end

      else if f<fp[is] then
         begin
            fp[ih]:=f;
            ih:=is;
            for i:=1 to n do
               begin
                  p[k]:=x.para[i];
                  k:=k+1;
               end;
         end

      else
         begin
            if f<fp[ih] then
               begin
                  for i:=1 to n do
                     begin
                        p[k]:=x.para[i];
                        k:=k+1;
                     end;
                  fp[ih]:=f;
                  k:=k-n;
               end;

            k0:=nn+1;
            for i:=1 to n do
               begin
                  xs[i]:=0.5*(p[k]+p[k0]);
                  k:=k+1;
                  k0:=k0+1;
               end;
            k:=k-n;

            for i:=1 to 10 do xd.para[i]:=xs[i];
            COPYPARA(para,xd);
            DETRESP(para,data);
            fs:=SUMERRSQR(para,data);

            if fs<fp[ih] then
               begin
                  for i:=1 to n do
                     begin
                        p[k]:=xs[i];
                        k:=k+1;
                     end;
                  fp[ih]:=fs;
                  if fp[1]>fp[2] then ih:=1
                                 else ih:=2;
                  for i:=3 to n1 do
                     if fp[i]>fp[ih] then ih:=i;
               end
            else loopend:='y';
         end;
      END;
```

```
{-------------------------------------------------------------------------}
{***   SUBPROCEDURE OUTERLOOP                                          ***}
{-------------------------------------------------------------------------}

   PROCEDURE outerloop;

      VAR
         i,j:integer;

     BEGIN
         fp[1]:=fp[il];
         if il<>1 then
            begin
               k:=(il-1)*n;
               for i:=1 to n do
                  begin
                     k:=k+1;
                     x.para[i]:=p[k];
                     p[k]:=p[i];
                     p[i]:=x.para[i];
                  end;
               il:=1;
            end;
         k:=n;
         for i:=2 to nl do
            for j:=1 to n do
               begin
                  k:=k+1;
                  p[k]:=0.5*(p[k]+p[j]);
               end;
         loopend:='n';
   END;


{-------------------------------------------------------------------------}
{***   MAIN PROCEDURE                                                  ***}
{-------------------------------------------------------------------------}

   BEGIN
      SELECTPARA(para,x);
      SELECTPARA(para,xd);

      n:=x.no;ix:=0;
      for i:=1 to 22 do fp[i]:=0;
      clrscr;gotoXY(25,12);writeln('REGRESSING USING NELM');

      for i:=1 to n do
         begin
            h[i]:=0.02*abs(x.para[i]);
            if h[i]=0 then h[i]:=0.01;
         end;
      nl:=n+1;nn:=n*nl;np:=n*(n+2);endd:='n';

      COPYPARA(para,x);
      DETRESP(para,data);
      f:=SUMERRSQR(para,data);fp[1]:=f;

      if ix=0 then
         begin
            for i:=1 to n do
               begin
                  k:=i;
                  for j:=1 to nl do
                     begin
                        p[k]:=x.para[i];
                        if ((i-j+1)=0) then p[k]:=x.para[i]+h[i];
                        k:=k+n;
                     end;    {for j:=1 to nl}
               end;    {for i:=1 to n}
         end;    {if ix=0}

      repeat  {LOOP I}
         k:=n+1;
         for i:=2 to nl do
            begin
               for j:=1 to n do
                  begin
                     x.para[j]:=p[k];
                     k:=k+1;
                  end;     {for j:=1 to n}
               COPYPARA(para,x);
               DETRESP(para,data);
               f:=SUMERRSQR(para,data);fp[i]:=f;
            end;     {for i:=2 to nl}

         if fp[1]>fp[2] then
            begin
               ih:=1;
               il:=2
            end
         else
            begin
               ih:=2;
               il:=1;
            end;     {if fp[1]>fp[2]}
```

```
        i:=3;
        repeat     (LOOP II)
           if fp[i]>fp[ih] then ih:=i
           else if fp[i]<fp[il] then il:=i;
           i:=i+1;
        until i>n1;     (LOOP II)


        xn:=n;

        repeat     (LOOP III)
           k1:=nn;
           for i:=1 to n do
              begin
                 k:=i;s:=0;
                 for j:=1 to n1 do
                    begin
                       if j<>ih then s:=s+p[k];
                       k:=k+n;
                    end;     (for j:=1 to n1)
                 k1:=k1+1;
                 p[k1]:=s/xn;
              end;     (for i:= 1 to n)

           k:=nn+1;
           for i:=1 to n do
               begin
                  x.para[i]:=p[k];
                  k:=k+1;
               end;

           COPYPARA(para,x);
           DETRESP(para,data);
           f0:=SUMERRSQR(para,data);s:=0;
           for i:=1 to n1 do s:=s+sqr(fp[i]-f0);
           s:=s/xn;a0:=sqrt(s);

           gotoXY(3,3);
           writeln('Eps:',eps:10,'         Std dev.:',a0:12,'         S[Ei]^2:',f0:12);


           if (keypressed) or (a0<=eps) then
              begin
                 endd:='y';
                 loopend:='y';
              end;

           if endd<>'y' then innerloop;

        until loopend='y';     (LOOP III)

        if endd<>'y' then outerloop;

     until endd='y'; (LOOP I)

     k:=(il-1)*n+1;
     COPYPARA(para,x);
     DETRESP(para,data);
  END;

{*************************************************************************}
{H**   REVISION HISTORY:                                                  ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0       KCAD     28-02-90                                     ***}
{***                                                                     ***}
{***   UNITEND  ESTIMATE.PAS                                             ***}
{*************************************************************************}
END.
```

C.3.21 Model.pas

```
{*************************************************************************}
{***   UNIT PROCEDURE LIBRARY TO DEFINE A THEORETICAL MODEL            ***}
{***   FILE: MODEL.PAS                                                 ***}
{*************************************************************************}

UNIT model;

{*************************************************************************}
                        INTERFACE
{*************************************************************************}

   USES crt,types,plotgraf,modeld;

   VAR
      model1:modeldata;
      limits:responsescale;

   PROCEDURE fittime00(p:parameterdata;VAR data:responsedata);
   PROCEDURE fittime11(p:parameterdata;VAR data:responsedata);
   PROCEDURE fittime21(p:parameterdata;VAR data:responsedata);
   PROCEDURE fittime22(p:parameterdata;VAR data:responsedata);
   PROCEDURE fitz1(p:parameterdata;VAR datamod:responsedata);
   PROCEDURE fitz2(p:parameterdata;VAR datamod:responsedata);
   PROCEDURE initmodel(VAR parameter:parameterdata;VAR data:responsedata);
   PROCEDURE detresp(parameter:parameterdata;VAR data:responsedata);
   FUNCTION sumerrsqr(p:parameterdata;data:responsedata):real;
   PROCEDURE chgpara(VAR para:parameterdata;VAR data:responsedata);
   PROCEDURE initpara(VAR para:parameterdata);
   PROCEDURE selectpara(para:parameterdata;VAR parasel:parameterdata);
   PROCEDURE copypara(VAR para:parameterdata;parasel:parameterdata);


{*************************************************************************}
                        IMPLEMENTATION
{*************************************************************************}

{*************************************************************************}
{N**   MODULE NAME       : FITTIME00                                    ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE         : calculates a series of theoretical function ***}
{***                       values for a linear response (a/s^2) for a  ***}
{***                       given set of parameter values               ***}
{S**   CALL SEQUENCE     : FITTIME00(parameter,responsedata)           ***}
{I**   INPUT PARAMETERS  : Parameter data record, response data record ***}
{O**   OUTPUT PARAMETERS : modelled response in response data record   ***}
{G**   GLOBAL VARIABLES  : none                                        ***}
{M**   MODULES CALLED    : Types                                       ***}
{E**   ERROR CONDITIONS  : none                                        ***}
{C**   COMMENTS          : for details about parameter data record and ***}
{***                       response data record see TYPES and for the  ***}
{***                       meaning of the parameters see MODELD        ***}
{*************************************************************************}

PROCEDURE fittime00(p:parameterdata;VAR data:responsedata);

   VAR
      i:integer;
      value,t:real;


  BEGIN
      for i:=0 to data.nsamples-1 do
         begin
            t:=i/data.frequency;

            if t>p.para[2] then value:=p.para[3]*(t-p.para[2])    {if t>then dead time}
            else value:=0;          {if t< than dead time then value=0}

            data.data[2,i]:=round(p.para[1]+value);
         end;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : FITTIME11                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : calculates a series of theoretical function ***}
{***                         values for a first order response, i.e.     ***}
{***                         a(sz+1)/(sp+1) for a given set of parameter ***}
{***                         values (in the real time domain)            ***}
{S**   CALL SEQUENCE      : FITTIME11(parameter,responsedata)           ***}
{I**   INPUT PARAMETERS   : Parameter data record, response data record ***}
{O**   OUTPUT PARAMETERS  : modelled response in response data record   ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : Types                                       ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : for details about parameter data record and ***}
{***                         response data record see TYPES and for the  ***}
{***                         meaning of the parameters see MODELD        ***}
{*************************************************************************}

PROCEDURE fittime11(p:parameterdata;VAR data:responsedata);

   VAR
      value,t:real;
      i:integer;

   BEGIN
      for i:=0 to data.nsamples-1 do
         begin
            t:=i/data.frequency;

            if t>p.para[2] then     {if t>then dead time)
                value:=1-exp(-(t-p.para[2])*p.para[4])
            else value:=0;          {if t< than dead time then value=0}

            data.data[2,i]:=round(p.para[1]+p.para[3]*value);
         end;
   END;


{*************************************************************************}
{N**   MODULE NAME        : FITTIME21                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : calculates a series of theoretical function ***}
{***                         values for a second order response with real***}
{***                         poles , i.e. a(sz+1)/((sp+1)(sq+1)) for a   ***}
{***                         given set of parameter values (in the  real ***}
{***                         time domain)                                ***}
{S**   CALL SEQUENCE      : FITTIME21(parameter,responsedata)           ***}
{I**   INPUT PARAMETERS   : Parameter data record, response data record ***}
{O**   OUTPUT PARAMETERS  : modelled response in response data record   ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : Types                                       ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : for details about parameter data record and ***}
{***                         response data record see TYPES and for the  ***}
{***                         meaning of the parameters see MODELD        ***}
{*************************************************************************}

PROCEDURE fittime21(p:parameterdata;VAR data:responsedata);

   VAR
      i:integer;
      K1,K2,value,t:real;


   BEGIN
      K1:=p.para[5]*(p.para[4]-p.para[6])/(p.para[6]*(p.para[5]-p.para[4]));
      K2:=p.para[6]*(p.para[5]-p.para[6])/(p.para[6]*(p.para[4]-p.para[5]));

      for i:=0 to data.nsamples-1 do
         begin
            t:=i/data.frequency;

            if t>p.para[2] then     {if t>then dead time)
                value:= 1 + K1*exp(-p.para[4]*(t-p.para[2])) + K2*exp(-p.para[5]*(t-p.para[2]))
            else value:=0;          {if t< than dead time then value=0)

            data.data[2,i]:=round(p.para[1]+p.para[3]*value);
         end;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : FITTIME22                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : calculates a series of theoretical function ***}
{***                        values for a second order response with     ***}
{***                        complex poles , i.e.                        ***}
{***                        a(sz+1)/((s+p+jq)(s+p-jq)) for a given set  ***}
{***                        of parameter values(in the real time domain)***}
{S**   CALL SEQUENCE      : FITTIME22(parameter,responsedata)           ***}
{I**   INPUT PARAMETERS   : Parameter data record, response data record ***}
{O**   OUTPUT PARAMETERS  : modelled response in response data record   ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : Types                                       ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : for details about parameter data record and ***}
{***                        response data record see TYPES and for the  ***}
{***                        meaning of the parameters see MODELD        ***}
{*************************************************************************}

PROCEDURE fittime22(p:parameterdata;VAR data:responsedata);

   VAR
      i:integer;
      K1,value,t:real;


   BEGIN
      K1:=(p.para[4]/p.para[5])-((sqr(p.para[4])+sqr(p.para[5]))/(p.para[6]*p.para[5]));
      for i:=0 to data.nsamples-1 do
         begin
            t:=i/data.frequency;

            if t>p.para[2] then     {if t>then dead time}
                 value:=1-exp(-(t-p.para[2])*p.para[4])*(cos((t-p.para[2])*p.para[5])+
                              K1*sin((t-p.para[2])*p.para[5]))
            else value:=0;          {if t< than dead time then value=0}

            data.data[2,i]:=round(p.para[1]+p.para[3]*value);
         end;
   END;



{*************************************************************************}
{N**   MODULE NAME        : FITZ1                                         ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : calculate a series of theoretical function  ***}
{***                        values for a first order response (using a  ***}
{***                        z-domain model) for a given set of parameter***}
{***                        values                                      ***}
{S**   CALL SEQUENCE      : FITZ1(parameter,responsedata)               ***}
{I**   INPUT PARAMETERS   : Parameter data record, response data record ***}
{O**   OUTPUT PARAMETERS  : modelled response in response data record   ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : Types                                       ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : for details about parameter data record and ***}
{***                        response data record see TYPES and for the  ***}
{***                        meaning of the parameters see MODELD        ***}
{*************************************************************************}

PROCEDURE fitz1(p:parameterdata;VAR datamod:responsedata);

   VAR
      i,u_1:integer;
      B,D,T,C1:real;
      data:array[0..1000] of real;

   BEGIN
      T:=1/datamod.frequency;
      C1:=p.para[2]*T;

      B:=p.para[1]*(1-exp(-C1));
      D:=-exp(-C1);

      u_1:=1;
      data[0]:=0;
      data[1]:=0;
      data[2]:=round(B);

      for i:=3 to datamod.nsamples-1 do
          data[i]:=B*u_1 - D*data[i-1];

      for i:=0 to datamod.nsamples-1 do
         datamod.data[1,i]:=round(data[i]);
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : FITZ2                                         ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : calculate a series of theoretical function  ***}
{***                        values for a second order response (using a ***}
{***                        z-domain model) for a given set of parameter***}
{***                        values                                        ***}
{S**   CALL SEQUENCE      : FITZ2(parameter,responsedata)               ***}
{I**   INPUT PARAMETERS   : Parameter data record, response data record ***}
{O**   OUTPUT PARAMETERS  : modelled response in response data record   ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : Types                                       ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : for details about parameter data record and ***}
{***                        response data record see TYPES and for the  ***}
{***                        meaning of the parameters see MODELD        ***}
{*************************************************************************}

PROCEDURE fitz2(p:parameterdata;VAR datamod:responsedata);

   VAR
      i,u_1,u_2:integer;
      B,C,D,E,T,C1,C2:real;
      data:array[0..1000] of real;

   BEGIN
      T:=1/datamod.frequency;
      C1:=p.para[2]*T;C2:=p.para[3]*T;

      B:=p.para[1]*(1+exp(-C1)*(-cos(C2)+C1/C2*sin(C2)));
      C:=p.para[1]*exp(-C1)*(exp(-C1)-cos(C2)-C1/C2*sin(C2));
      D:=-2*exp(-C1)*cos(C2);
      E:=exp(-2*C1);

      u_2:=1;u_1:=1;
      data[0]:=0;
      data[1]:=0;
      data[2]:=round(B);

      for i:=3 to datamod.nsamples-1 do
          data[i]:=B*u_1 + C*u_2 - D*data[i-1] - E*data[i-2];

      for i:=0 to datamod.nsamples-1 do
         datamod.data[1,i]:=round(data[i]);
   END;

{*************************************************************************}
{N**   MODULE NAME        : INITMODEL                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : to choose a particular model, e.g. a first  ***}
{***                        or second order model in the t- or z-domain ***}
{S**   CALL SEQUENCE      : INITMODEL(parameter,responsedata)           ***}
{I**   INPUT PARAMETERS   : parameter data record, response data record ***}
{O**   OUTPUT PARAMETERS  : record describing the model, initial para-  ***}
{***                        meter estimates in a parameter data record  ***}
{G**   GLOBAL VARIABLES   : modell                                       ***}
{M**   MODULES CALLED     : Types, Modeld                               ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : for details about response and parameter    ***}
{***                        data record see TYPES                       ***}
{*************************************************************************}

PROCEDURE initmodel(VAR parameter:parameterdata; VAR data:responsedata);

   VAR
      label1:string[6];
      label2:string[7];
      opt:char;
      menu:screenmenu;

   BEGIN
      INITMENU1(menu);
      clrscr;WRITEMENU(menu);

      repeat
         opt:='c';
         case modell.order of
            0:label1:='LINEAR';
            1:label1:=' FIRST';
            2:label1:='SECOND';
         end;
         gotoXY(37,10);write('      ');
         gotoXY(37,10);write(label1);

         case modell.typ of
            0:label2:='   NONE';
            1:label2:='   REAL';
            2:label2:='COMPLEX';
         end;
         gotoXY(34,12);write(label2);
         gotoXY(5,20);write('Option [',opt,']: ');opt:=readkey;
         if opt=#13 then opt:='c';
```

```
        case opt of
           'o': begin
                  modell.order:= modell.order+1;
                if modell.order=3 then
                      begin        {change here if higher order models are included}
                         sound(1000);delay(250);nosound;
                         modell.order:=0;
                         modell.typ:=0;
                      end;
                   if modell.order=1 then modell.typ:=1;
                end;
           'p':if modell.order<2 then
                  begin
                     sound(500);delay(250);nosound;
                  end
               else if modell.typ=1 then modell.typ:=2 else modell.typ:=1;
        end;

     until opt='c';
     clrscr;HIGHLIGHT(1);gotoXY(35,10);writeln('PLEASE WAIT');
     INITPARA(parameter);
     DETRESP(parameter,data);
     normvideo;
  END;

{*************************************************************************}
{N**   MODULE NAME        : DETRESP                                       ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : calculate a set of theoretical function       ***}
{***                        values for a given set of parameter values    ***}
{***                        depending on the specified model              ***}
{S**   CALL SEQUENCE      : DETRESP(parameter,responsedata)               ***}
{I**   INPUT PARAMETERS   : parameter data record, response data record,***}
{***                        specified model data record                   ***}
{O**   OUTPUT PARAMETERS  : theoretical response in response data record***}
{G**   GLOBAL VARIABLES   : modell                                        ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about response, parameter or      ***}
{***                        model data record see TYPES                   ***}
{*************************************************************************}

PROCEDURE detresp(parameter:parameterdata;VAR data:responsedata);

  BEGIN
     case modell.order  of
        0:fittime00(parameter,data);
        1:fittime11(parameter,data);
        2:case modell.typ of
             1:fittime21(parameter,data);
             2:fittime22(parameter,data);
          end;
     end;
  END;


{*************************************************************************}
{N**   MODULE NAME        : SUMERRSQR                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   FUNCTION           : determines the sum of error squared between ***}
{***                        a measured and a modelled response            ***}
{S**   CALL SEQUENCE      : SUMERRSQR(parameter,responsedata)             ***}
{I**   INPUT PARAMETERS   : parameter data record, response data record,***}
{***                        limits of response                            ***}
{O**   OUTPUT PARAMETERS  : sum of error squared                         ***}
{G**   GLOBAL VARIABLES   : limits of response                           ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : for details about response and parameter      ***}
{***                        data records see TYPES                        ***}
{*************************************************************************}

FUNCTION sumerrsqr(p:parameterdata;data:responsedata):real;

  VAR
     min,max,i:integer;
     diff:longint;
     error:real;

  BEGIN
     min:=round(limits.tmin * data.frequency);
     max:=round(limits.tmax * data.frequency);

     error:=0;
     for i:=min to max-1 do
        begin
           diff:=data.data[1,i]-data.data[2,i];
           error:=error + sqr(diff);
        end;
     writeln;
     sumerrsqr:=error;
  END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : CHGPARA                                       ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : change model parameters manually              ***}
{S**   CALL SEQUENCE      : CHGPARA(parameter,responsedata)               ***}
{I**   INPUT PARAMETERS   : old parameter data record,limits of response***}
{O**   OUTPUT PARAMETERS  : updated parameter data record                ***}
{G**   GLOBAL VARIABLES   : limits of response                           ***}
{M**   MODULES CALLED     : Crt,Plotgraf,Types,Modeld                     ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           :                                             ***}
{*************************************************************************}

PROCEDURE chgpara(VAR para:parameterdata;VAR data:responsedata);

   VAR
       i,opt,fix:integer;
       error:real;
       option:char;
       par:string[8];
       menu:screenmenu;

   BEGIN
      INITMENU2(menu);

      repeat
         clrscr;
         WRITEMENU(menu);
         HIGHLIGHT(1);gotoXY(30,13);writeln('PLEASE WAIT');
         if (option<>'g') and (option<>'v') then
            begin
               DETRESP(para,data);
               error:=SUMERRSQR(para,data);
            end;
         option:='c';normvideo;
         gotoXY(39,7);writeln(error:12);

         for i:= 1 to para.no do
            begin
               if para.vari[i]=0 then par:='FIXED' else par:='VARIABLE';
               gotoXY(19,10+i);
               write(i,' - Parameter ',i,' : ',para.para[i]:10:5,'        ',par);
            end;
         gotoXY(5,20);write('option [',option,']: ');option:=readkey;writeln(option);
         if option=#13 then option:='c';

         if ord(option)<58 then
            begin
               opt:=Ord(option)-48;
               readreal(37,10+opt,10,para.para[opt]);
            end
         else if option='v' then
            begin
               gotoXY(5,22);write('Which Parameter to change from fix <-> var ? ');
               fix:=ORD(readkey)-48;
               if para.vari[fix]=0 than para.vari[fix]:=1 else para.vari[fix]:=0;
            end
         else if option='g' then GRAFCOMP(data,limits);

      until option='c';
   END;


{*************************************************************************}
{N**   MODULE NAME        : INITPARA                                      ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initialize the parameters of the specified    ***}
{***                        model                                         ***}
{S**   CALL SEQUENCE      : INITPARA(parameter)                           ***}
{I**   INPUT PARAMETERS   : modell                                       ***}
{O**   OUTPUT PARAMETERS  : initial estimates of model parameters        ***}
{G**   GLOBAL VARIABLES   : modell                                       ***}
{M**   MODULES CALLED     : Types,Modeld                                  ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           : for details about modell or parameter data    ***}
{***                        record see TYPES                              ***}
{*************************************************************************}

PROCEDURE initpara(VAR para:parameterdata);

   BEGIN
      case modell.order of
         0:INITPARA00(para);
         1:INITPARA11(para);
         2:if modell.typ = 1 then INITPARA21(para)
           else if modell.typ = 2 then INITPARA22(para);
      end;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : SELECTPARA                                    ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : copies model parameters to be regressed into***}
{***                         a seperate parameter data record which is   ***}
{***                         then used by NELM                           ***}
{S**   CALL SEQUENCE      : SELECTPARA(parameter,paraselect)              ***}
{I**   INPUT PARAMETERS   : parameter data record                         ***}
{O**   OUTPUT PARAMETERS  : new parameter data record with selected      ***}
{***                         parameters to be regressed on               ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           : for details about parameter data record see ***}
{***                         TYPES                                       ***}
{*************************************************************************}

PROCEDURE selectpara(para:parameterdata;VAR parasel:parameterdata);

   VAR
      i,k:integer;

   BEGIN
      k:=0;
      for i:=1 to para.no do
         if para.vari[i]=1 then
            begin
               k:=k+1;
               parasel.para[k]:=para.para[i];
               parasel.vari[k]:=i;
            end;

      parasel.no:=k;

      for i:= k+1 to para.no do
         begin
            parasel.para[i]:=1;
            parasel.vari[i]:=0;
         end;
   END;


{*************************************************************************}
{N**   MODULE NAME        : COPYPARA                                      ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : copies the regressed model parameters back    ***}
{***                         into the old parameter data record to deter-***}
{***                         mine the sum of error squared and to display***}
{***                         the result graphically                      ***}
{S**   CALL SEQUENCE      : COPYPARA(para,parasel)                        ***}
{I**   INPUT PARAMETERS   : regressed parameter data record               ***}
{O**   OUTPUT PARAMETERS  : updated original parameter data record        ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           : for details about parameter data record see ***}
{***                         TYPES                                       ***}
{*************************************************************************}

PROCEDURE copypara(VAR para:parameterdata;parasel:parameterdata);

   VAR
      i,n:integer;

   BEGIN
      for i:=1 to parasel.no do
         begin
            n:=parasel.vari[i];
            para.para[n]:=parasel.para[i];
         end;
   END;


{*************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0       KCAD     28-02-90                                     ***}
{***                                                                     ***}
{***   UNITEND MODEL.PAS                                                 ***}
{*************************************************************************}
END.
```

C.3.22 Modeld.pas

```
{************************************************************************}
{***   UNIT PROCEDURE LIBARY CONTAINING DATAFILES TO BE USED IN        ***}
{***        MODEL.PAS                                                  ***}
{***   FILE: MODELD.PAS                                                ***}
{************************************************************************}

UNIT modeld;

{************************************************************************}
                        INTERFACE
{************************************************************************}

   USES Types;


   PROCEDURE initmenu1(VAR menu:screenmenu);
   PROCEDURE initmenu2(VAR menu:screenmenu);
   PROCEDURE initpara00(VAR para:parameterdata);
   PROCEDURE initpara11(VAR para:parameterdata);
   PROCEDURE initpara21(VAR para:parameterdata);
   PROCEDURE initpara22(VAR para:parameterdata);

{************************************************************************}
                        IMPLEMENTATION
{************************************************************************}

{************************************************************************}
{N**   MODULE NAME       : INITMENU1                                    ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE         : initializes the menu defined in this sub-    ***}
{***                       routine                                      ***}
{S**   CALL SEQUENCE     : INITMENU1(menu)                              ***}
{I**   INPUT PARAMETERS  : none                                         ***}
{O**   OUTPUT PARAMETERS : menu1  (type screenmenu)                     ***}
{G**   GLOBAL VARIABLES  : none                                         ***}
{M**   MODULES CALLED     : Types                                        ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS          : details about screenmenu in TYPES            ***}
{************************************************************************}

PROCEDURE initmenu1(VAR menu:screenmenu);

   BEGIN
      menu[1]:='╔═══════════════════════════════════════════════════════════╗';
      menu[2]:='║                                                           ║';
      menu[3]:='║          RE-SPECIFICATION OF MODEL                        ║';
      menu[4]:='║                                                           ║';
      menu[5]:='║                                                           ║';
      menu[6]:='║                                                           ║';
      menu[7]:='║                        OPTIONS:                           ║';
      menu[8]:='║                        ========                           ║';
      menu[9]:='║                                                           ║';
     menu[10]:='║              o  -  Response =        Order                ║';
     menu[11]:='║                                                           ║';
     menu[12]:='║              p  -  Poles =                                ║';
     menu[13]:='║                                                           ║';
     menu[14]:='║                                                           ║';
     menu[15]:='║                                                           ║';
     menu[16]:='║                                                           ║';
     menu[17]:='║              c  -  Continue                               ║';
     menu[18]:='║                                                           ║';
     menu[19]:='╚═══════════════════════════════════════════════════════════╝';
     menu[20]:='';
     menu[21]:='';
     menu[22]:='';
     menu[23]:='';
     menu[24]:='';
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITMENU2                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                        routine                                       ***}
{S**   CALL SEQUENCE      : INITMENU2(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : menu2  (type screenmenu)                      ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : details about screenmenu in TYPES             ***}
{*************************************************************************}

PROCEDURE initmenu2(VAR menu:screenmenu);

  BEGIN
     menu[1]:='╔═══════════════════════════════════════════════════════════════════╗';
     menu[2]:='║                                                                   ║';
     menu[3]:='║          RE-SPECIFICATION OF MODEL PARAMETERS                     ║';
     menu[4]:='║                                                                   ║';
     menu[5]:='║                                                                   ║';
     menu[6]:='║                                                                   ║';
     menu[7]:='║          SUM OF ERROR^2 :                                         ║';
     menu[8]:='║                                                                   ║';
     menu[9]:='║                                                                   ║';
    menu[10]:='║            Parameter to change:                                   ║';
    menu[11]:='║                                                                   ║';
    menu[12]:='║                                                                   ║';
    menu[13]:='║                                                                   ║';
    menu[14]:='║                                                                   ║';
    menu[15]:='║                                                                   ║';
    menu[16]:='║                                                                   ║';
    menu[17]:='║                                                                   ║';
    menu[18]:='║   v - fixed <-> variable    g - Plot Graph    c - Continue        ║';
    menu[19]:='╚═══════════════════════════════════════════════════════════════════╝';
    menu[20]:='';
    menu[21]:='';
    menu[22]:='';
    menu[23]:='';
    menu[24]:='';
  END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITPARA00                                    ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the parameter for a zero order    ***}
{***                        model (i.e 1/s^2) as defined in this sub-     ***}
{***                        routine                                       ***}
{S**   CALL SEQUENCE      : INITPARA00(parameter)                         ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : parameter data record                        ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : details about parameter data record in TYPES***}
{*************************************************************************}

PROCEDURE initpara00(VAR para:parameterdata);

  BEGIN
     para.no:=3;                              {no of parameter}
     para.para[1]:=0;   para.vari[1]:=0;
     para.para[2]:=100;   para.vari[2]:=0;
     para.para[3]:=100;  para.vari[3]:=1;

     { Parameter of Model: Linear Response                }
     { ====================================               }
     {                                                    }
     { Parameter[1] = Baselevel          : value     flag }
     { Parameter[2] = Dead Time          : value     flag }
     { Parameter[3] = Slope of Linear    : value     flag }
     {                                                    }
     { NOTE: flag = 1 => parameter can be regressed on    }
     { ===== flag = 0 => parameter cannot be regressed on}
  END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITPARA11                                    ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the parameter for a first order ***}
{***                        model as defined in this subroutine          ***}
{S**   CALL SEQUENCE      : INITPARA11(parameter                          ***}
{I**   INPUT PARAMETERS   : none                                         ***}
{O**   OUTPUT PARAMETERS  : parameter data record                        ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           : details about parameter data record in TYPES***}
{*************************************************************************}

PROCEDURE initpara11(VAR para:parameterdata);

   BEGIN
      para.no:=4;                          {no of parameter}
      para.para[1]:=0;   para.vari[1]:=0;
      para.para[2]:=100;   para.vari[2]:=0;
      para.para[3]:=100; para.vari[3]:=1;
      para.para[4]:=0.01;   para.vari[4]:=1;


      { Parameter of Model: First Order Response - No Zeroes}
      { ====================================================}

      { no of Parameter
      { Parameter[1] = Baselevel          : value     flag   }
      { Parameter[2] = Dead Time          : value     flag   }
      { Parameter[3] = Staedy State Gain  : value     flag   }
      { Parameter[4] = Real Pole          : value     flag   }


      { NOTE: flag = 1 => parameter can be regressed on      }
      { ===== flag = 0 => parameter cannot be regressed on   }
   END;


{*************************************************************************}
{N**   MODULE NAME        : INITPARA21                                    ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the parameter for a second order***}
{***                        model with real poles as defined in this     ***}
{***                        subroutine                                   ***}
{S**   CALL SEQUENCE      : INITPARA21(parameter)                         ***}
{I**   INPUT PARAMETERS   : none                                         ***}
{O**   OUTPUT PARAMETERS  : parameter data record                        ***}
{G**   GLOBAL VARIABLES   : none                                         ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS           : details about parameter data record in TYPES***}
{*************************************************************************}

PROCEDURE initpara21(VAR para:parameterdata);

   BEGIN
      para.no:=6;                          {no of parameter}
      para.para[1]:=0;     para.vari[1]:=0;
      para.para[2]:=100;   para.vari[2]:=0;
      para.para[3]:=100;   para.vari[3]:=1;
      para.para[4]:=0.01;  para.vari[4]:=1;
      para.para[5]:=0.01;  para.vari[5]:=1;
      para.para[6]:=100;   para.vari[6]:=0;

      { Parameter of Model: Two Real Poles with Real Zero}
      { ==================================================}

      { no of Parameter
      { Parameter[1] = Baselevel          : value    flag }
      { Parameter[2] = Dead Time          : value    flag }
      { Parameter[3] = Steady State Gain  : value    flag }
      { Parameter[4] = First Real Pole    : value    flag }
      { Parameter[5] = Second Real Pole   : value    flag }
      { Parameter[6] = Real Zero          : value    flag }


      { NOTE: flag = 1 => parameter can be regressed on  }
      { ===== flag = 0 => parameter cannot be regressed on }
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME       : INITPARA22                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE         : initializes the parameter for a second order***}
{***                        model with imaginary poles as defined in     ***}
{***                        this subroutine                              ***}
{S**   CALL SEQUENCE     : INITPARA22(parameter)                          ***}
{I**   INPUT PARAMETERS  : none                                           ***}
{O**   OUTPUT PARAMETERS : parameter data record                         ***}
{G**   GLOBAL VARIABLES  : none                                           ***}
{M**   MODULES CALLED    : Types                                          ***}
{E**   ERROR CONDITIONS  : none                                           ***}
{C**   COMMENTS          : details about parameter data record in TYPES***}
{*************************************************************************}

PROCEDURE initpara22(VAR para:parameterdata);

   BEGIN
      para.no:=6;                          {no of parameter}
      para.para[1]:=0;     para.vari[1]:=0;
      para.para[2]:=100;   para.vari[2]:=0;
      para.para[3]:=100;   para.vari[3]:=0;
      para.para[4]:=0.01;  para.vari[4]:=1;
      para.para[5]:=0.01;  para.vari[5]:=1;
      para.para[6]:=100;   para.vari[6]:=1;

      { Parameter of Model: Complex Conjugate Poles with Real Zero }
      { ========================================================== }

      { no of Parameter                                            }
      { Parameter[1] = Baselevel                 : value     flag  }
      { Parameter[2] = Dead Time                 : value     flag  }
      { Parameter[3] = Staedy State Gain         : value     flag  }
      { Parameter[4] = Real Part of complex Pole : value     flag  }
      { Parameter[5] = Imag.Part of Complex Pole : value     flag  }
      { Parameter[6] = Real Zero                 : value     flag  }

      { NOTE: flag = 1 => parameter can be regressed on            }
      { ===== flag = 0 => parameter cannot be regressed on         }

   END;


{*************************************************************************}
{H**   REVISION HISTORY:                                                ***}
{***   VERSION      BY        DATE       COMMENTS                       ***}
{***     1.0       KCAD     28-02-90                                    ***}
{***                                                                    ***}
{***   UNITEND  MODELD.PAS                                              ***}
{*************************************************************************}
END.
```

C.3.23 Detfric.pas

```
{*************************************************************************}
{***   PROGRAM TO DETERMINE THE DEPENDANCE OF FRICTION UPON BELTSPEED  ***}
{***           AND MASSFLOW ON THE BELT                                ***}
{***   FILE: DETFRIC.PAS                                               ***}
{*************************************************************************}

{*************************************************************************}
{N**   PROGRAM NAME      : DETFRIC                                       ***}
{***   ------------------------------------------------------------    ***}
{A**   DESCRIPTION       : determines the dependance of friction upon    ***}
{***                       beltspeed and massflow on the belt, i.e. it   ***}
{***                       creates a file which can then be used to      ***}
{***                       determine the coefficients of the function    ***}
{***                       F = f(v,m).                                   ***}
{S**   CALL SEQUENCE     : MAIN PROGRAM                                  ***}
{I**   INPUT PARAMETERS  : datafile                                      ***}
{O**   OUTPUT PARAMETERS : friction, beltspeed, actual massflow         ***}
{G**   GLOBAL VARIABLES  : none                                          ***}
{M**   MODULES CALLED    : Crt,Types,Filehand,Detmf,Math,Plotgraf,       ***}
{***                       Detfricd                                      ***}
{E**   ERROR CONDITIONS  : none                                          ***}
{C**   COMMENTS          : none                                          ***}
{*************************************************************************}

PROGRAM detfric;

{$M 65000,0,655360}

   USES Crt,types,filehand,detfricd,detmf,math,plotgraf;


   CONST
      pathdata='data\';
      datpath='data\';


   VAR
      name,namef,datname:str50;
      i,nset,ntest,samplepos,testpos:integer;
      corrdata:correlationdata;
      regdata:regressiondata;
      Csp:constant;
      Cm:massflowconst;
      menu1,menu2:screenmenu;
      option:char;


{*************************************************************************}
{***                    PROCEDURES                                     ***}
{*************************************************************************}

{-------------------------------------------------------------------------}
{***  change position of counters in filename                          ***}
{-------------------------------------------------------------------------}

PROCEDURE chgpos;

   BEGIN
      repeat
         clrscr;
         gotoXY(15,7);writeln('Filename: ',name);
         gotoXY(15,15);writeln('enter position of main sample counter using direction keys: ');
         HIGHLIGHT(1);gotoXY(24+samplepos,8);write('^');normvideo;
         option:=readkey;

         if option<>#13 then
            if option=#0 then
               case readkey of
                  #75:samplepos:=samplepos-1;
                  #77:samplepos:=samplepos+1;
               end;

      until option=#13;
```

```
      repeat
         clrscr;
         gotoXY(15,7);writeln('Filename: ',name);
         gotoXY(15,15);writeln('enter position of test counter using direction keys: ');
         HIGHLIGHT(1);gotoXY(24+samplepos,8);write('C');
         gotoXY(24+testpos,8);write('^');normvideo;
         option:=readkey;

         if option<>#13 then
            if option=#0 then
               case readkey of
                  #75:testpos:=testpos-1;
                  #77:testpos:=testpos+1;
               end;

      until option=#13;
   END;


{-------------------------------------------------------------------------}
{***  Change Filenames                                                ***}
{-------------------------------------------------------------------------}

PROCEDURE chgname;

   VAR
      menu3:screenmenu;
      opt:char;

   BEGIN
      INITMENU3(menu3);

      repeat
         clrscr;
         WRITEMENU(menu3);
         gotoXY(19,12);write(name);
         gotoXy(19,15);write(namef);
         gotoXY(19,18);write(datname);
         gotoXY(19,21);write(corrdata.title);
         gotoXY(4,25);write('Option[q]: ');
         opt:=readkey;
         if opt=#13 then opt:='q';

         case opt of
            'n','N':FILENAME(name,'');
            'z','Z':FILENAME(namef,'');
            's','S':FILENAME(datname,'');
            't','T':corrdata.title:=CHGTITLE('Title ',corrdata.title);
         end;

      until opt='q';
   END;


{-------------------------------------------------------------------------}
{***  Change Variables                                                ***}
{-------------------------------------------------------------------------}

PROCEDURE chgvars;

   VAR
      modechar:string[10];
      option:char;
      pulselength:integer;

   BEGIN
      INITMENU2(menu2);
      repeat
         clrscr;
         WRITEMENU(menu2);
         gotoXY(56,10);writeln(nset);
         gotoXY(56,12);writeln(ntest);
         gotoXY(56,14);writeln(corrdata.nzero);
         gotoXY(5,25);write('Option [q]: ');

         option:=readkey;
         if option=#13 then option:='q';

         case option of
            's','S':READINT(56,10,5,nset);
            't','T':READINT(56,12,5,ntest);
            'z','Z':READINT(56,14,5,corrdata.nzero);
            'p','P':chgpos;
            'n','N':chgname;
         end;
         corrdata.nsamples:=corrdata.nzero+nset*ntest;
      until (option='Q') or (option='q');

   END;
```

```
{------------------------------------------------------------------------}
{***  Determine Friction, beltspeed and Mass on Belt                  ***}
{------------------------------------------------------------------------}

PROCEDURE detfriction;

   VAR
      i,j,start,range:integer;
      energy,dist:real;
      respdata:responsedata;
      s:string[1];
      s2:string[2];

   PROCEDURE getmass(c,n:integer;name:str50);

      VAR
         j,k:integer;
         Vav:real;

      BEGIN
         for j:=1 to n do
            begin
               delete(name,testpos,1);
               str(j,s);
               insert(s,name,testpos);
               LOADRESPONSEDATA(name,respdata);

               dist:=AREA(respdata,1,start,range)*Csp.speed;
               Energy:=AREA(respdata,2,start,range)*Csp.power;

               corrdata.data[1,c+j]:=MASSBIN(Cm,respdata);
               corrdata.data[2,c+j]:=dist*respdata.frequency/range;
               if dist<>0 then corrdata.data[3,c+j]:=Energy/dist else corrdata.data[3,c+j]:=0;
               corrdata.data[4,c+j]:=AREA(respdata,1,0,100)*Csp.speed*respdata.frequency/100;
            end;
      END;


   BEGIN
      LOADRESPONSEDATA(name,respdata);
      start:=0;
      range:=respdata.nsamples;
      clrscr;gotoXY(10,10);write('Starting Sample: [',start:3,'] ');READINT(33,10,4,start);
      gotoXY(10,12);write('No of Samples  : [',range:3,'] ');READINT(33,12,4,range);

      getmass(0,corrdata.nzero,name);

      for i:=1 to nset do
         begin
            if i<10 then
               begin
                  delete(name,samplepos,1);
                  str(i,s);
                  insert(s,name,samplepos);
               end
            else
               begin
                  delete(name,samplepos-1,2);
                  str(i,s2);
                  insert(s2,name,samplepos-1);
               end;
            getmass(corrdata.nzero+(i-1)*ntest,ntest,name);
         end;

      getmass(corrdata.nsamples,corrdata.nzero,namef);

      MASSFLOWBELT(Cm,corrdata);
   END;
```

```
{*************************************************************************}
{***                      MAIN PROGRAM                                 ***}
{*************************************************************************}

BEGIN
   LOADCONST(Csp);
   LOADCONSTM(Cm);
   INITVAR(nset,ntest,corrdata.nzero,samplepos,testpos);
   with regdata do
      for i:=1 to 2 do
         begin
            slope[i]:=0;
            yconst[i]:=0;
            xmean[i]:=0;
            ymean[i]:=0;
            C1[i]:=0;
            C2[i]:=0;
            r[i]:=0;
            yerror[i]:=0;
         end;


   with corrdata do
      begin
         nsamples:=nzero+nset*ntest;
         title:='FRICTION DEPENDANCE: '
      end;

   datname:='';
   insert('.dat',datname,1);
   insert(datpath,datname,1);

   name:='';
   insert('.dat',name,1);
   insert('',name,1);
   insert(pathdata,name,1);
   namef:=name;
   initmenu1(menu1);

   repeat
      clrscr;
      writemenu(menu1);
      gotoXY(5,25);write('Option? ');
      option:=readkey;

      case option of
         'i','I':chgvars;
         'd','D':detfriction;
         's','S':begin
                    FILENAME(datname,'');
                    SAVECORRDATA(datname,corrdata,regdata);
                 end;
      end;
   until (option='q') or (option='Q');

{*************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENTS                       ***}
{***     1.0       KCAD     01-03-90                                     ***}
{***                                                                     ***}
{***   FILEEND DETFRIC.PAS                                               ***}
{*************************************************************************}
END.
```

C.3.24 Detfricd.pas

```
{*************************************************************************}
{***   UNIT PROCEDURE LIBARY CONTAINING DATAFILES TO BE USED IN        ***}
{***        DETFRIC.PAS                                                ***}
{***   FILE: DETFRICD.PAS                                              ***}
{*************************************************************************}

UNIT detfricd;

{*************************************************************************}
                        INTERFACE
{*************************************************************************}

   USES types;

   PROCEDURE initvar(VAR nsamples,ntest,nzero,samplepos,testpos:integer);
   PROCEDURE initmenu1(VAR menu:screenmenu);
   PROCEDURE initmenu2(VAR menu:screenmenu);
   PROCEDURE initmenu3(VAR menu:screenmenu);

{*************************************************************************}
                        IMPLEMENTATION
{*************************************************************************}

{*************************************************************************}
{N**   MODULE NAME       : INITVAR                                      ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE         : initializes the variabels used by the        ***}
{***                       program DETFRIC                              ***}
{S**   CALL SEQUENCE     : INITVAR(nsamples,ntest,nzero,samplepos,      ***}
{***                             testpos)                               ***}
{I**   INPUT PARAMETERS  : none                                         ***}
{O**   OUTPUT PARAMETERS : no of samples, no of tests per sample, no of***}
{***                       initializing samples, position of sample and***}
{***                       of test counter in filename                  ***}
{G**   GLOBAL VARIABLES  : none                                         ***}
{M**   MODULES CALLED     : Types                                        ***}
{E**   ERROR CONDITIONS   : none                                         ***}
{C**   COMMENTS          : none                                         ***}
{*************************************************************************}

PROCEDURE initvar(VAR nsamples,ntest,nzero,samplepos,testpos:integer);

   BEGIN
      nsamples:=10;
      ntest:=5;
      nzero:=5;
      samplepos:=6;
      testpos:=6;
   END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITMENU1                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                        program                                       ***}
{S**   CALL SEQUENCE      : INITMENU1(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : menu1 (type screenmenu)                       ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : details about screenmenu in TYPES             ***}
{*************************************************************************}

PROCEDURE initmenu1(VAR menu:screenmenu);

  BEGIN
       menu[1]:='╔═══════════════════════════════════════════════════════════════╗';
       menu[2]:='║                                                               ║';
       menu[3]:='║            FRICTION DETERMINATION PACKAGE                     ║';
       menu[4]:='║                                                               ║';
       menu[5]:='║                                                               ║';
       menu[6]:='║                                                               ║';
       menu[7]:='║                        MENU:                                  ║';
       menu[8]:='║                                                               ║';
       menu[9]:='║                                                               ║';
      menu[10]:='║                                                               ║';
      menu[11]:='║                i - Initialize Variables                       ║';
      menu[12]:='║                                                               ║';
      menu[13]:='║                d - Load Responses and Determine Friction      ║';
      menu[14]:='║                                                               ║';
      menu[15]:='║                                                               ║';
      menu[16]:='║                                                               ║';
      menu[17]:='║                                                               ║';
      menu[18]:='║                                                               ║';
      menu[19]:='║                s - Save Data                                  ║';
      menu[20]:='║                                                               ║';
      menu[21]:='║                                                               ║';
      menu[22]:='║                q - Quit                                       ║';
      menu[23]:='║                                                               ║';
      menu[24]:='╚═══════════════════════════════════════════════════════════════╝';

  END;

{*************************************************************************}
{N**   MODULE NAME        : INITMENU2                                     ***}
{***   ------------------------------------------------------------      ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                        program                                       ***}
{S**   CALL SEQUENCE      : INITMENU2(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : menu2 (type screenmenu)                       ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : details about screenmenu in TYPES             ***}
{*************************************************************************}

PROCEDURE initmenu2(VAR menu:screenmenu);

  BEGIN
       menu[1]:='╔═══════════════════════════════════════════════════════════════╗';
       menu[2]:='║                                                               ║';
       menu[3]:='║            INITIALIZE VARIABLES                               ║';
       menu[4]:='║                                                               ║';
       menu[5]:='║                                                               ║';
       menu[6]:='║                                                               ║';
       menu[7]:='║                        MENU:                                  ║';
       menu[8]:='║                                                               ║';
       menu[9]:='║                                                               ║';
      menu[10]:='║          s - No of Massflow Sample Sets        :              ║';
      menu[11]:='║              (excluding Zero Massflow Samples)                ║';
      menu[12]:='║          t - No of Tests per Set               :              ║';
      menu[13]:='║                                                               ║';
      menu[14]:='║          z - No of Zero-Massflow Samples       :              ║';
      menu[15]:='║                                                               ║';
      menu[16]:='║                                                               ║';
      menu[17]:='║                                                               ║';
      menu[18]:='║                                                               ║';
      menu[19]:='║          n - Change Filenames and Titles of Correlation Data  ║';
      menu[20]:='║          p - Indicate Position of Counters in Filename        ║';
      menu[21]:='║                                                               ║';
      menu[22]:='║                                                               ║';
      menu[23]:='║          q - Quit                                             ║';
      menu[24]:='╚═══════════════════════════════════════════════════════════════╝';

  END;
```

```
{*************************************************************************}
{N**   MODULE NAME        : INITMENU3                                     ***}
{***   ------------------------------------------------------------------ ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-     ***}
{***                        program                                       ***}
{S**   CALL SEQUENCE      : INITMENU3(menu)                               ***}
{I**   INPUT PARAMETERS   : none                                          ***}
{O**   OUTPUT PARAMETERS  : menu3 (type screenmenu)                       ***}
{G**   GLOBAL VARIABLES   : none                                          ***}
{M**   MODULES CALLED     : Types                                         ***}
{E**   ERROR CONDITIONS   : none                                          ***}
{C**   COMMENTS           : details about screenmenu in TYPES             ***}
{*************************************************************************}

PROCEDURE initmenu3(VAR menu:screenmenu);

  BEGIN
       menu[1]:='╔═══════════════════════════════════════════════════════════════════╗';
       menu[2]:='║                                                                   ║';
       menu[3]:='║            CHANGE FILE NAMES AND TITLES                           ║';
       menu[4]:='║                                                                   ║';
       menu[5]:='║                                                                   ║';
       menu[6]:='║                                                                   ║';
       menu[7]:='║                          CHANGE:                                  ║';
       menu[8]:='║                                                                   ║';
       menu[9]:='║                                                                   ║';
      menu[10]:='║          n - Filename of Normal Massflow Samples:                 ║';
      menu[11]:='║              (Including Initial Massflow Samples)                 ║';
      menu[12]:='║                                                                   ║';
      menu[13]:='║                                                                   ║';
      menu[14]:='║          z - Filename of Final Zero Massflow Samples:             ║';
      menu[15]:='║                                                                   ║';
      menu[16]:='║                                                                   ║';
      menu[17]:='║          s - Filename to Save Friction Data to    :               ║';
      menu[18]:='║                                                                   ║';
      menu[19]:='║                                                                   ║';
      menu[20]:='║          t - Title of Test                                        ║';
      menu[21]:='║                                                                   ║';
      menu[22]:='║                                                                   ║';
      menu[23]:='║          q - Quit                                                 ║';
      menu[24]:='╚═══════════════════════════════════════════════════════════════════╝';

  END;

{*************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0       KCAD     28-02-90                                     ***}
{***                                                                     ***}
{***   UNITEND  DETFRICD.PAS                                             ***}
{*************************************************************************}
END.
```

C.3.25 Dconv.pas

```
{*************************************************************************}
{***   PROGRAM TO CONVERT A RECORDED TIME RESPONSE                     ***}
{***   FILE: DCONV.PAS                                                 ***}
{*************************************************************************}

{*************************************************************************}
{N**   PROGRAM NAME       : DCONV                                       ***}
{***   ------------------------------------------------------------    ***}
{A**   DESCRIPTION        : The programs loads up a recorded response.  ***}
{***                        It then can filter, subtract the offset,    ***}
{***                        disregard all samples before the step, norm-***}
{***                        alize a step_UP response and change the     ***}
{***                        sampling frequency. Lastly it saves the pro-***}
{***                        cessed response                             ***}
{S**   CALL SEQUENCE      : MAIN PROGRAM                                ***}
{I**   INPUT PARAMETERS   : file to convert, which conversion to perform***}
{***                        breakfrequency of filter, new sampling freq-***}
{***                        uency                                       ***}
{O**   OUTPUT PARAMETERS  : converted data records on file              ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : Crt,Graph,Plotgraf,Math,Filehand,Types,     ***}
{***                        Dconvd                                      ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : all channels in the data record are con-    ***}
{***                        verted                                      ***}
{*************************************************************************}

PROGRAM dconv;

   USES Crt,graph,plotgraf,math,filehand,types,dconvd;


   CONST
       pathdata='data\';


   VAR
       i,pos,ch:integer;
       break,newfreq:real;
       endid,step,save,graf:string[4];
       what:string[20];
       name:str50;
       data:responsedata;
       limits:responsescale;
       menu:screenmenu;
       module:array[1..5] of string[3];
       opt:char;

{*************************************************************************}
{***                      PROCEDURES                                   ***}
{*************************************************************************}

{-------------------------------------------------------------------------}
{***  DRAW GRAPH                                                       ***}
{-------------------------------------------------------------------------}

PROCEDURE drawgraph(what:string);

   BEGIN
      AUTORESPSCALE(data,limits,ch);
      GRAPHSETUP(data,limits,ch,data.title);
      PLOTDATALIN(data,ch,limits);
      LABELS(data,ch,1,what,limits);
      repeat until keypressed;readln;
      closegraph;
   END;
```

```
{-------------------------------------------------------------------------}
{***  Do Conversion                                                  ***}
{-------------------------------------------------------------------------}

PROCEDURE doconversion;

   VAR
      i,j,pos:integer;

   BEGIN
      HIGHLIGHT(1);gotoXY(5,25);write('PLEASE WAIT');

      pos:=POSITION(name);
      for i:=1 to 5 do
         begin
            if module[i]='ON' then
               begin
                  delete(name,pos-1,1);
                  case i of
                     1:begin
                          for j:=1 to data.nochannel do filter(break,j,data);
                          insert('f',name,pos);
                          what:='FILTERED RESPONSE';
                       end;
                     2:begin
                          offset(data);
                          insert('o',name,pos);
                          what:='Offset Subtracted';
                       end;
                     3:begin
                          disregard_up_to_step(data);
                          insert('s',name,pos);
                          what:='Step disregarded ';
                       end;
                     4:begin
                          norm(data);
                          insert('n',name,pos);
                          what:='Response Normalized';
                       end;
                     5:begin
                          chgfreq(newfreq,data);
                          insert('c',name,pos);
                          what:='Sampl. Freq. changed';
                       end;
                  end;
                  if save='YES' then SAVERESPONSEDATA(name,data);
                  if graf='YES' then drawgraph(what);
               end;
         end;

      normvideo;

      delete(name,pos-1,1);
      insert('x',name,pos);
      SAVERESPONSEDATA(name,data);

      delete(name,pos-1,1);
      insert('t',name,pos);
      LOADRESPONSEDATA(name,data);
      AUTORESPSCALE(data,limits,1);
   END;
```

```
{*************************************************************************}
{***                    MAIN PROGRAM                                   ***}
{*************************************************************************}

BEGIN
   INITMENU1(menu);

   name:='';
   insert('.dat',name,1);
   insert(pathdata,name,1);
   FILENAME(name,'.dat');
   pos:=POSITION(name);
   endid:=copy(name,pos-4,4);
   LOADRESPONSEDATA(name,data);

   for i:=1 to 5 do module[i]:='ON';
   newfreq:=data.frequency;

   save:='NO';
   graf:='NO';
   break:=50;

   repeat
      if data.step>0 then step:='UP' else step:='DOWN';
      clrscr;
      WRITEMENU(menu);
      for i:=1 to 5 do
         begin
            gotoXY(-5+i*14,9);write(module[i]);
         end;
      gotoXY(37,17);write(break:6:2);
      gotoXY(37,18);write(newfreq:6:2);
      gotoXY(41,19);write(graf);
      gotoXY(41,20);write(save);
      gotoXY(63,16);write(data.data[1,0]:5);
      gotoXY(63,17);write(data.data[1,data.nsamples-1]:5);
      gotoXY(65,18);write(data.frequency:6:2);
      gotoXY(67,19);write(data.whenstep);
      gotoXY(66,20);write(abs(data.step):5:2);
      gotoXY(67,21);write(step);
      gotoXY(5,25);write('Option ? ');opt:=readkey;

      case opt of
         'f','F':if module[1]='ON' then module[1]:='OFF' else module[1]:='ON';
         'o','O':if module[2]='ON' then module[2]:='OFF' else module[2]:='ON';
         's','S':if module[3]='ON' then module[3]:='OFF' else module[3]:='ON';
         'n','N':if module[4]='ON' then module[4]:='OFF' else module[4]:='ON';
         'c','C':if module[5]='ON' then module[5]:='OFF' else module[5]:='ON';
         'b','B':readreal(36,17,7,break);
         'd','D':begin
                    READREAL(36,18,7,newfreq);
                    if newfreq>data.frequency then
                       begin
                          sound(400);delay(500);nosound;
                          newfreq:=data.frequency;
                       end;
                 end;
         'r','R':if save='YES' then save:='NO' else save:='YES';
         'g','G':if graf='YES' then graf:='NO'
                 else
                    begin
                       graf:='YES';
                       clrscr;
                       gotoXY(20,10);write('Which channel do you want to view? ');readln(ch);
                    end;
         'x','X':begin
                    FILENAME(name,endid);
                    pos:=position(name);
                    endid:=copy(name,pos-4,4);
                    LOADRESPONSEDATA(name,data);
                 end;
         #13:doconversion;
      end;

   until (opt='q') or (opt='Q');


{*************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0       KCAD     01-03-89                                     ***}
{***                \                                                    ***}
{***   FILEEND  DCONV.DAT                                                ***}
{*************************************************************************}
END.
```

C.3.26 Dconvd.pas

```
{*************************************************************************}
{***   UNIT PROCEDURE LIBARY CONTAINING DATAFILES TO BE USED IN        ***}
{***        DCONV.PAS                                                  ***}
{***   FILE: DCONVD.PAS                                                ***}
{*************************************************************************}

UNIT dconvd;

{*************************************************************************}
                        INTERFACE
{*************************************************************************}

   USES types;

   PROCEDURE initmenu1(VAR menu:screenmenu);

{*************************************************************************}
                      IMPLEMENTATION
{*************************************************************************}

{*************************************************************************}
{N**   MODULE NAME        : INITMENU1                                   ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-   ***}
{***                        program                                     ***}
{S**   CALL SEQUENCE      : INITMENU1(menu)                             ***}
{I**   INPUT PARAMETERS   : none                                        ***}
{O**   OUTPUT PARAMETERS  : menu1 (type screenmenu)                     ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : Types                                       ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : for more details about screenmenu see TYPES ***}
{*************************************************************************}

PROCEDURE initmenu1(VAR menu:screenmenu);

   BEGIN
      menu[1]:='╔═══════════════════════════════════════════════════════════════════════╗';
      menu[2]:='║                                                                       ║';
      menu[3]:='║                            FLOW DIAGRAM                               ║';
      menu[4]:='║                                                                       ║';
      menu[5]:='║  ┌──────┐   ┌────────┐   ┌──────────┐   ┌──────────┐   ┌──────────┐  ║';
      menu[6]:='║  │      │   │Subtract│   │ Disregard │   │Normalize│   │  Change  │  ║';
      menu[7]:='║─>│Filter│──>│ Offset │──>│  Samples  │──>│Response │──>│Sampling  │  ║';
      menu[8]:='║  │      │   │        │   │before Step│   │         │   │Frequency │  ║';
      menu[9]:='║  └──────┘   └────────┘   └──────────┘   └──────────┘   └──────────┘  ║';
     menu[10]:='║    f            o             s             n             c         ║';
     menu[11]:='╠═══════════════════════════════════╦═══════════════════════════════╣';
     menu[12]:='║                                   ║                               ║';
     menu[13]:='║            Options:               ║     Fixed Variables:          ║';
     menu[14]:='║                                   ║                               ║';
     menu[15]:='║                                   ║                               ║';
     menu[16]:='║  <RETURN> - Do Conversion         ║  Initial Offset:              ║';
     menu[17]:='║  b - Breakfrequency of Filter:  Hz║  Final Offset  :              ║';
     menu[18]:='║  d - New Sampling Frequency   :  Hz║  Sampling Freq.:           Hz ║';
     menu[19]:='║  g - Plot Intermediate Responses :║  Time of Step  :              ║';
     menu[20]:='║  r - Save intermediate Responses :║  Size of Step  :              ║';
     menu[21]:='║  * - Switch Selected Module ON/OFF║        Step    :              ║';
     menu[22]:='║  x - New File                     ║                               ║';
     menu[23]:='║  q - Quit                         ║                               ║';
     menu[24]:='╚═══════════════════════════════════╩═══════════════════════════════╝';
   END;


{*************************************************************************}
{H**   REVISION HISTORY:                                               ***}
{***   VERSION      BY        DATE        COMMENTS                     ***}
{***     1.0       KCAD     28-02-90                                   ***}
{***                                                                   ***}
{***   UNITEND  DCONVD.PAS                                             ***}
{*************************************************************************}
END.
```

C.3.27 Fit.pas

```
{*************************************************************************}
{***   PROGRAM TO FIT A FUNCTION TO REAL TIME RESPONSE                  ***}
{***   FILE:  FIT.PAS                                                  ***}
{*************************************************************************}

{*************************************************************************}
{N**   PROGRAM NAME      : FIT                                          ***}
{***   ------------------------------------------------------------   ***}
{A**   DESCRIPTION       : to fit a function to a measured response     ***}
{***                       using Nelm (a non-linear regression tech-    ***}
{***                       nique) or the Least Squares estimation tech-***}
{***                       nique                                        ***}
{S**   CALL SEQUENCE     : MAIN PROGRAM                                 ***}
{I**   INPUT PARAMETERS  : file of measured data, initial estimates of ***}
{***                       parameters, accuracy requirements            ***}
{O**   OUTPUT PARAMETERS : parameters of model after running regression***}
{***                       technique                                    ***}
{G**   GLOBAL VARIABLES  : none                                         ***}
{M**   MODULES CALLED    : Crt,Filehand,Types,Plotgraf,Estimate,Model,  ***}
{***                       Fitd                                         ***}
{E**   ERROR CONDITIONS  : none                                         ***}
{C**   COMMENTS          : none                                         ***}
{*************************************************************************}

PROGRAM fit;

{$M 65000,0,655360}

   USES
      crt,filehand,types,plotgraf,estimate,model,fitd;

   CONST
      pathdat='data\';

   VAR
      channel:integer;
      eps:real;
      paradata:parameterdata;
      menu:screenmenu;
      prevname,name:str50;
      option:char;
      origdata,data:responsedata;

{-------------------------------------------------------------------------}
{***   copy response data to be regressed on into new data record      ***}
{-------------------------------------------------------------------------}

PROCEDURE copydata(ch:integer);

VAR
   i:integer;

BEGIN
   data.nsamples:=origdata.nsamples;
   data.nochannel:=2;
   data.whenstep:=origdata.whenstep;
   data.step:=origdata.step;
   data.frequency:=origdata.frequency;
   data.labels[1]:='MEASURED';
   data.labels[2]:='FITTED';
   data.title:='RESPONSE REGRESSION:';
   for i:=0 to data.nsamples-1 do
      data.data[1,i]:=origdata.data[ch,i];
END;
```

```
{*************************************************************************}
{***                    MAIN PROGRAM                                   ***}
{*************************************************************************}

BEGIN
   INITMENU1(menu);

   name:='';
   insert('.dax',name,1);
   insert(pathdat,name,1);
   FILENAME(name,'');
   LOADRESPONSEDATA(name,origdata);
   gotoXY(20,20);write('Which Channel to Investigate? ');readint(50,20,1,channel);
   copydata(channel);
   AUTORESPSCALE(origdata,limits,channel);

   model1.order:=1;
   model1.typ:=1;
   INITMODEL(paradata,data);

   repeat
      clrscr;
      WRITEMENU(menu);
      gotoXY(5,25);write('Option? ');option:=readkey;writeln(option);
      case option of
         'm','M': INITMODEL(paradata,data);
         'p','P': CHGPARA(paradata,data);
         'g','G': GRAFCOMP(data,limits);
         's','S': CHGSCALE(limits);
         'n','N': begin
                     write('Enter Epsilon: ');readln(eps);
                     NELM(eps,paradata,data);
                  end;
         'l','L':LSE2(paradata,data);
         'f','F': begin
                     prevname:=name;
                     FILENAME(name,'');
                     if prevname<>name then LOADRESPONSEDATA(name,origdata);
                     gotoXY(20,20);write('Which Channel to Investigate? ');readln(channel);
                     copydata(channel);
                     AUTORESPSCALE(origdata,limits,channel);
                  end;
      end;

   until (option='q') or (option='Q');


{*************************************************************************}
{H**   REVISION HISTORY:                                                 ***}
{***   VERSION      BY        DATE       COMMENTS                        ***}
{***     1.0       KCAD     01-03-90                                     ***}
{***                                                                     ***}
{***   FILEEND  FIT.PAS                                                  ***}
{*************************************************************************}
END.
```

C.3.28 Fitd.pas

```
{*************************************************************************}
{***   UNIT PROCEDURE LIBARY CONTAINING DATAFILES TO BE USED IN        ***}
{***        FIT.PAS                                                    ***}
{***   FILE: FITD.PAS                                                  ***}
{*************************************************************************}

UNIT fitd;

{*************************************************************************}
                        INTERFACE
{*************************************************************************}

   USES types;

   PROCEDURE initmenu1(VAR menu:screenmenu);


{*************************************************************************}
                        IMPLEMENTATION
{*************************************************************************}

{*************************************************************************}
{N**   MODULE NAME        : INITMENU1                                   ***}
{***   ------------------------------------------------------------    ***}
{A**   PROCEDURE          : initializes the menu defined in this sub-   ***}
{***                        program                                     ***}
{S**   CALL SEQUENCE      : INITMENU1(menu)                             ***}
{I**   INPUT PARAMETERS   : none                                        ***}
{O**   OUTPUT PARAMETERS  : menu1 (type screenmenu)                     ***}
{G**   GLOBAL VARIABLES   : none                                        ***}
{M**   MODULES CALLED     : Types                                       ***}
{E**   ERROR CONDITIONS   : none                                        ***}
{C**   COMMENTS           : details about screenmenu in TYPES           ***}
{*************************************************************************}

PROCEDURE initmenu1(VAR menu:screenmenu);

   BEGIN
      menu[1]:='╔═══════════════════════════════════════════════════════════╗';
      menu[2]:='║                                                           ║';
      menu[3]:='║          SYSTEM PARAMETER IDENTIFICATION PROGRAMME        ║';
      menu[4]:='║                                                           ║';
      menu[5]:='║                                                           ║';
      menu[6]:='║                                                           ║';
      menu[7]:='║                        MENU:                              ║';
      menu[8]:='║                                                           ║';
      menu[9]:='║                                                           ║';
     menu[10]:='║                                                           ║';
     menu[11]:='║                    p - Re-specify Model Parameters        ║';
     menu[12]:='║                    m - Change Model                       ║';
     menu[13]:='║                                                           ║';
     menu[14]:='║                    g - Draw Graph                         ║';
     menu[15]:='║                    s - Change Plotting Scales             ║';
     menu[16]:='║                                                           ║';
     menu[17]:='║                    n - Run Nelm                           ║';
     menu[18]:='║                    l - Run Least Squares Estimator        ║';
     menu[19]:='║                                                           ║';
     menu[20]:='║                    f - select New Measured Response       ║';
     menu[21]:='║                                                           ║';
     menu[22]:='║                    q - Quit                               ║';
     menu[23]:='║                                                           ║';
     menu[24]:='╚═══════════════════════════════════════════════════════════╝';

   END;


{*************************************************************************}
{H**   REVISION HISTORY:                                               ***}
{***   VERSION      BY        DATE       COMMENTS                      ***}
{***     1.0        KCAD     28-02-90                                  ***}
{***                                                                   ***}
{***   UNITEND  FITD.PAS                                               ***}
{*************************************************************************}
END.
```

APPENDIX D: EXPLANATION OF MENU STRUCTURES AND FUNCTIONS OF INDIVIDUAL MENUS

In this Appendix the menu structure and the functions of the available options are explained briefly. Actions required by the user in form of key strokes are declared in triangular brackets, e.g. <r> requires the user to press the key 'r'.

D.1 DLOG.EXE

D.1.1 Basic Menu Structure:

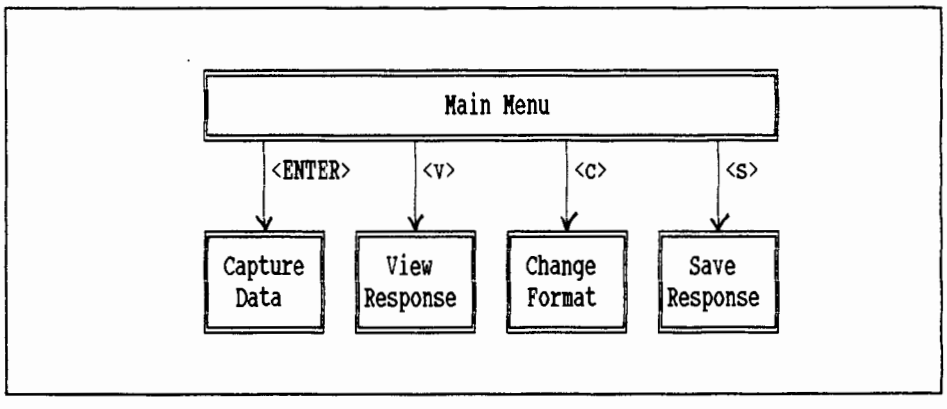


Figure D.1: Menu structure of DLOG.EXE

D.1.2 Description of Available Options:

Capture Data: Capture a real time response with parameters as specified in the Change Format Menu.

View Response: Plot the Response of all the channels logged on screen.

Save Response: Define title and save the captured response to a specified file on disc.

Change Format: Change one or more of the following variables:

<m> - Mode of Operation: Toggle between Step, Pulse or Random Sequence Test
<n> - No of samples to log
<f> - Sampling frequency
<e> - Initial Setpoint
<s> - Size of step
<w> - When to step the output channel
<o> - Which output channel to step
<c> - No of channels to log
<l> - Label of each channel logged
<p> - Length of pulse in case of a pulse response
<t> - Calibrate timing of capture routine

Step Up/Down?: When in Step Mode the option exists in the main Menu to toggle <t> the step up or down.

D.1.3 Copies of the Menus used:

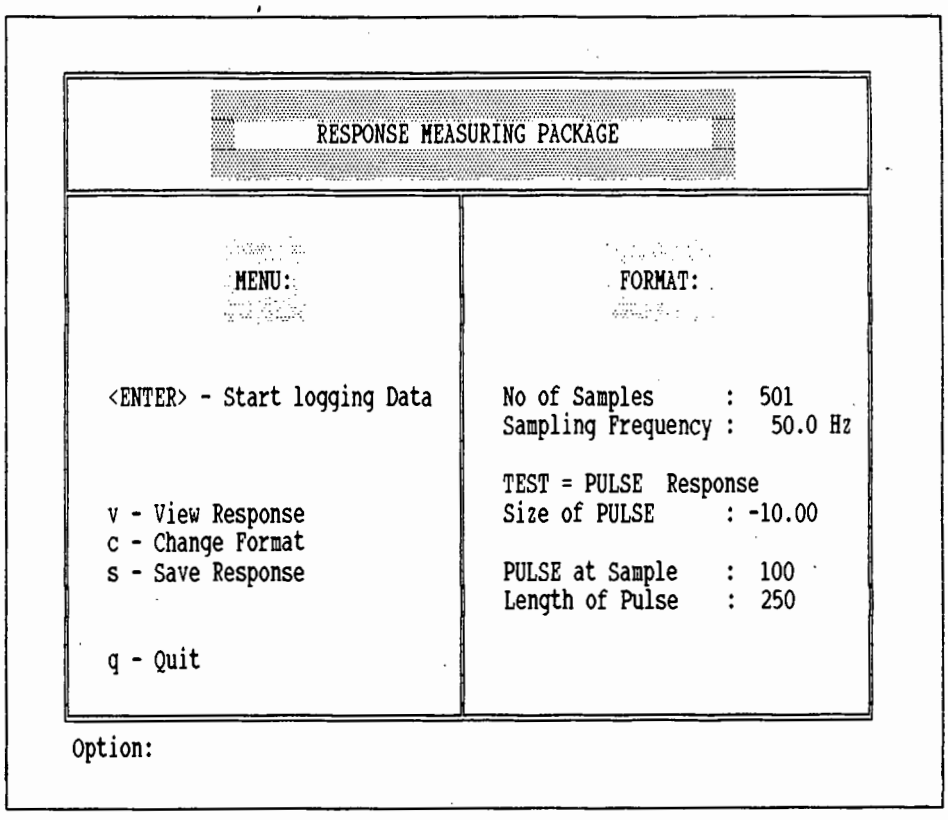


Figure D.2: Main menu of DLOG.EXE

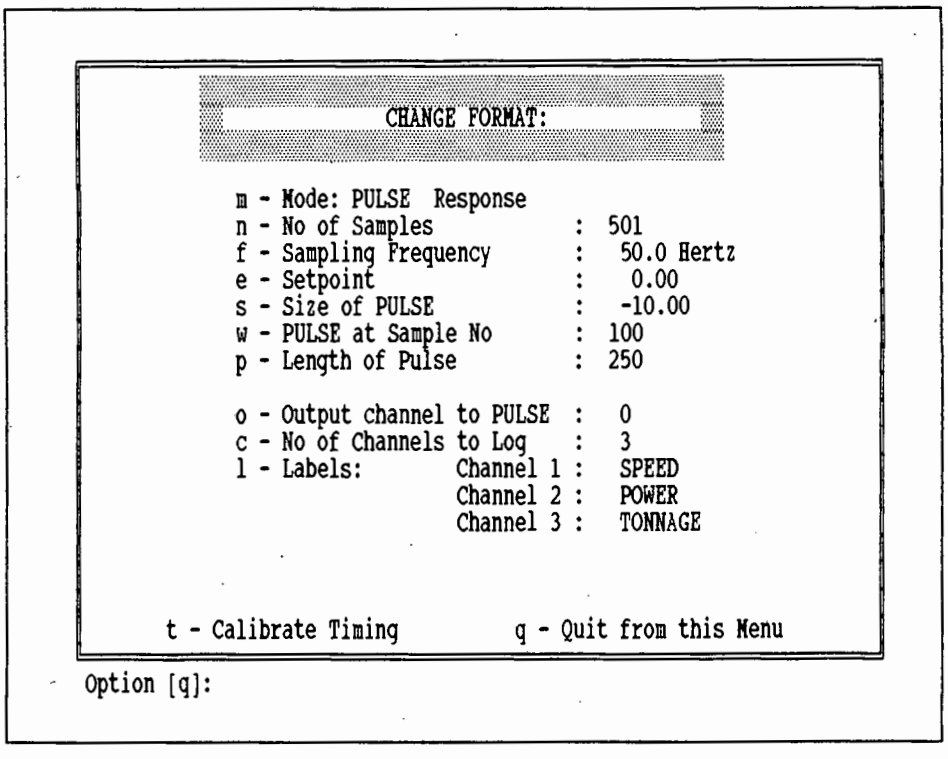


Figure D.3: Change Format menu of DLOG.EXE

D.2: MFCM.EXE

D.2.1 Basic Menu Structure:

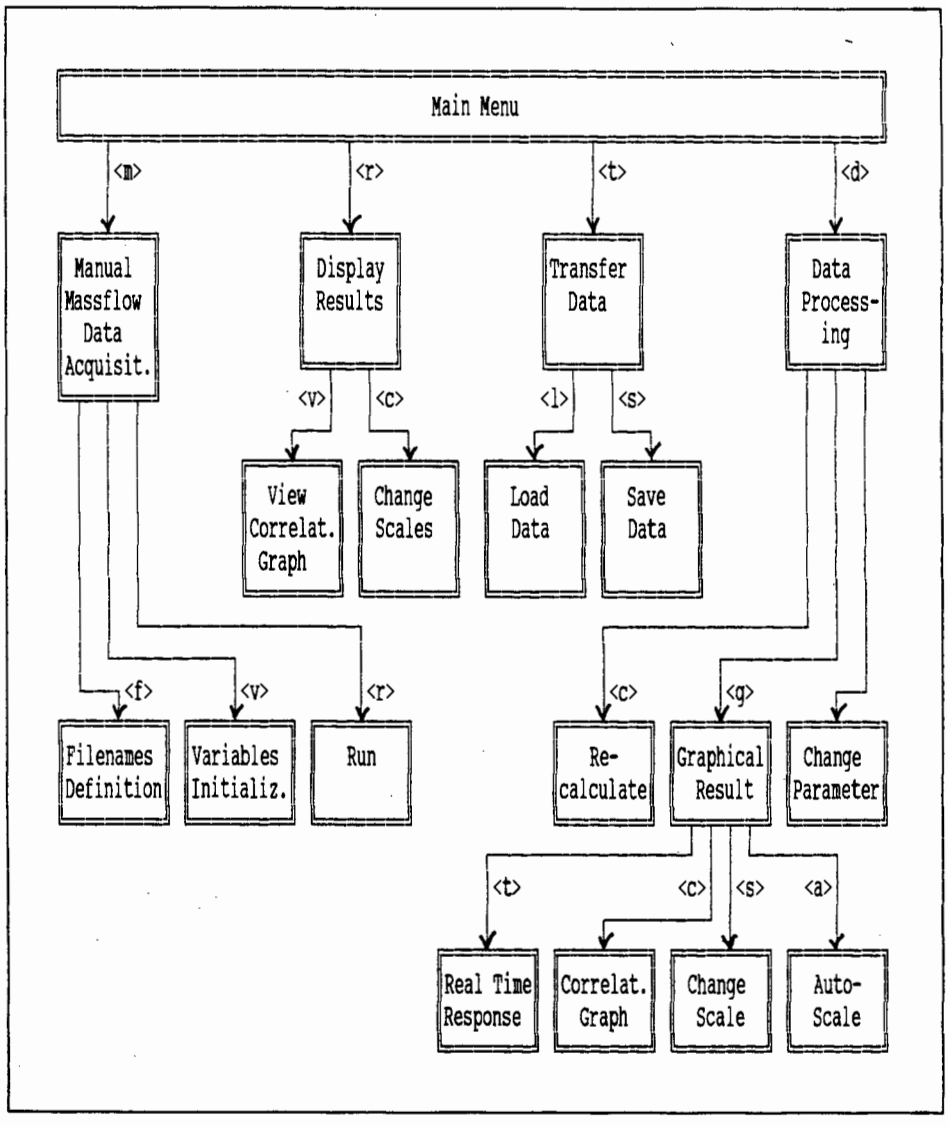


Figure D.4: Menu structure of MFCM.EXE

D.2.2 Description of Available Options:

Manual Massflow Data Acquisition: In this option the massflow on the CB is determined from a series of real time pulse responses.

To do this data acquisition automatically

certain filenames and variables need to be defined which is done in the *Filenames Definition* and *Variables Initialization* Menus.

In the *VARIABLES INITIALIZATION* menu the following needs to be defined:

<n> - no of massflow sample sets
<t> - no of massflow tests per set
<z> - no of zero massflow samples in the case where the reference massflow is obtained from the mass in the hopper bin (see description in chapter 5)
<a> - whether the reference massflow is obtained from the hopper in the bin or from a standard massflow meter
<m> - what test mode was used to in the real time response tests (either long pulse where the system is allowed to reach steady state in the perturbed mode or short pulse where the system is already stepped up again while still slowing down)

In the *FILENAMES DEFINITION* menu the following needs to be defined:

<n> - filename of real time response of first massflow sample
<z> - final zero massflow sample
<s> - the filename of where the massflow correlation data should be saved to
<t> - the title of the set of data
<p> - indicate the position of the test and set counter in the filename

The operation of the *RUN* procedure is explained in the flowchart down below:

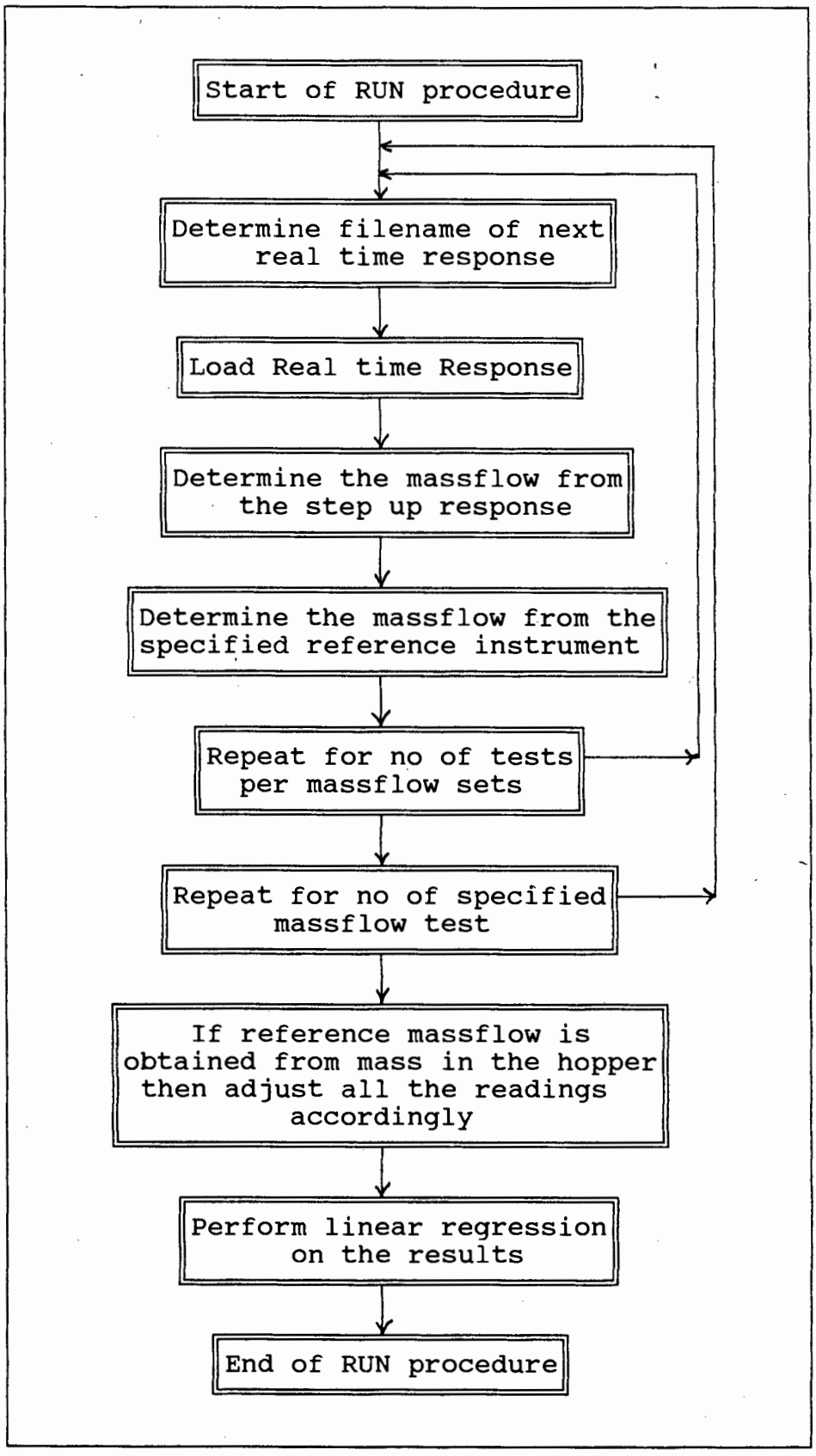


Figure D.5: Flowchart of to illustrate the working of the Manual Data Acquisition procedure

Display Results: In this option the results from the regression between the massflow estimates from the step-up and step-down tests and the reference instrument are shown numerically. The *VIEW CORRELATION GRAPH* option shows the correlation graphs of the step-down and step-up responses versus the reference massflow. The scales of these graphs can be changed in the *CHANGE SCALES* menu.

Transfer Data: In this option the results obtained from the manual data acquisition can be saved (*SAVE DATA*) and massflow correlation files obtained at a earlier stage from either the manual data acquisition or from a automatic data acquisition can be loaded (*LOAD DATA*).

Data Processing: This option can be used to determine the calibration coefficients of the correlation between the estimates from the step-up response and the reference massflow meter. The following needs to be defined:

<o> - the offset of the reference instrument
<z> - the zero offset or y intercept of the calibration curve
<s> - slope of calibration curve
<f> - filter time constant

The option of the first order filter is included to account for the dynamics of the instruments and the process involved (see explanation in section 3.3). The *RECALCULATE* option thus performs a linear regression on the calibrated and filtered data set and thus shows the resulting regression results.

The instrument is correctly calibrated if the y-intercept equals zero and the slope equals unity in these results. It has to be noted that the filter time constant improves the correlation coefficient as the noise is filtered out, but it also reduces the span over which the samples are varying. To make a judgement on the filter time constant one should have a look at the *GRAPHICAL RESULTS*. Here options exist to display the correlation graph and to display the massflow samples from the step-up response and the reference instrument versus time on a graph called 'Time Plot'. The scales of both types of graphs can be adjusted automatically or changed manually.

D.2.3 Copies of the Menus used:

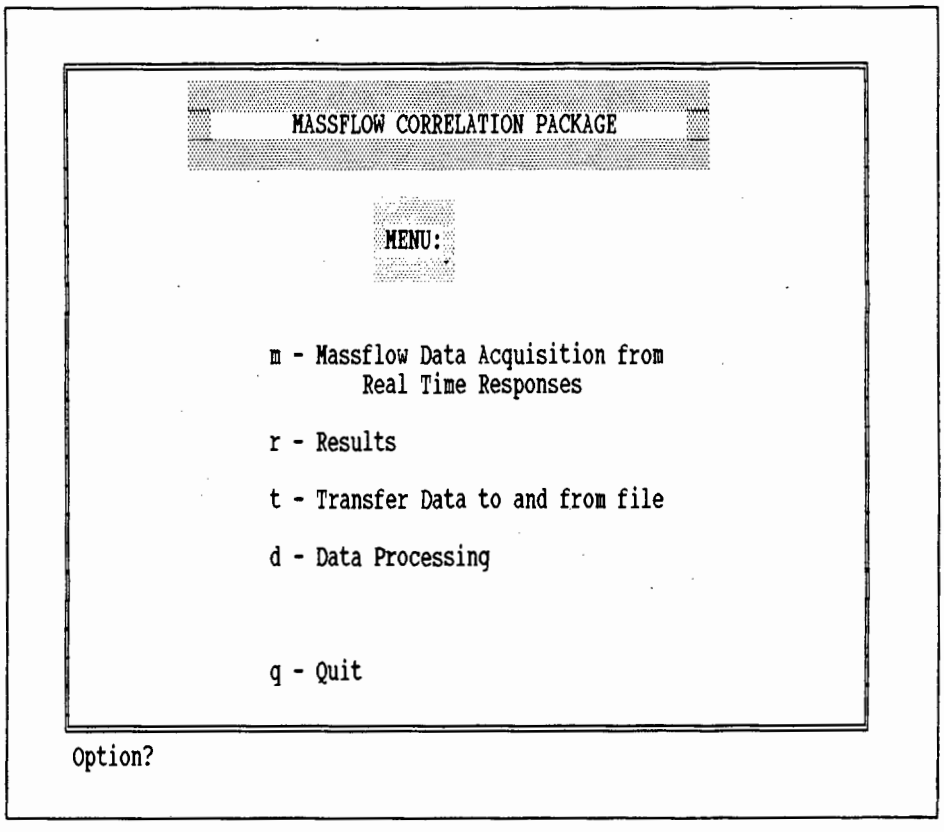


Figure D.6: Main menu of MFCM.EXE

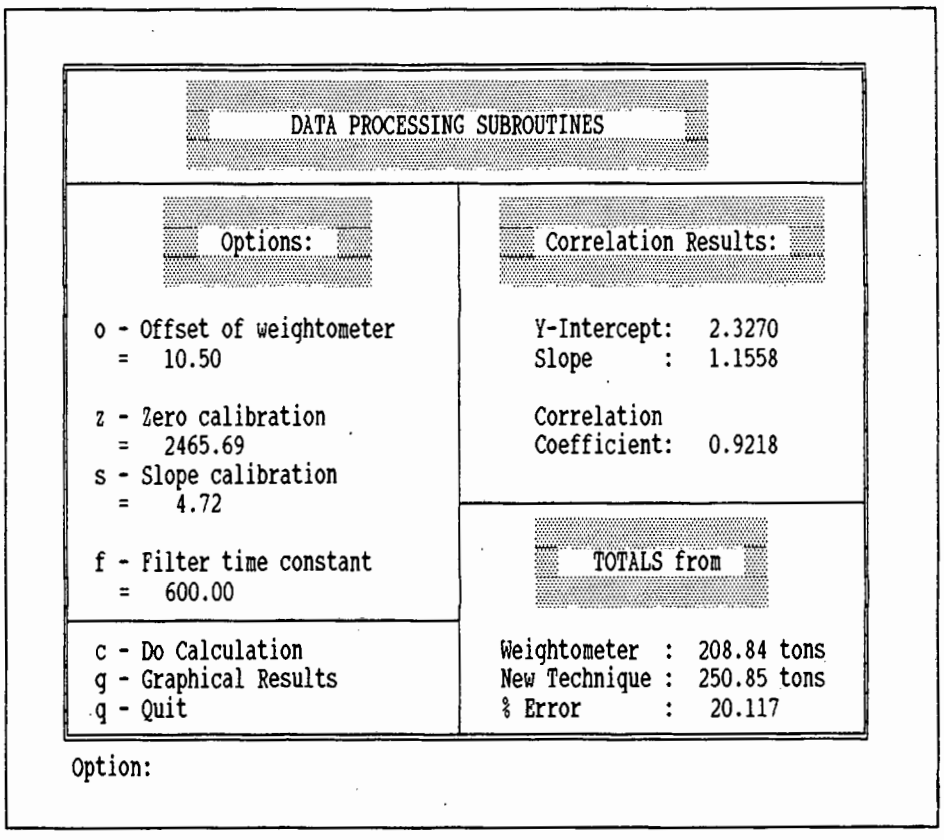


Figure D.7: Data Processing menu

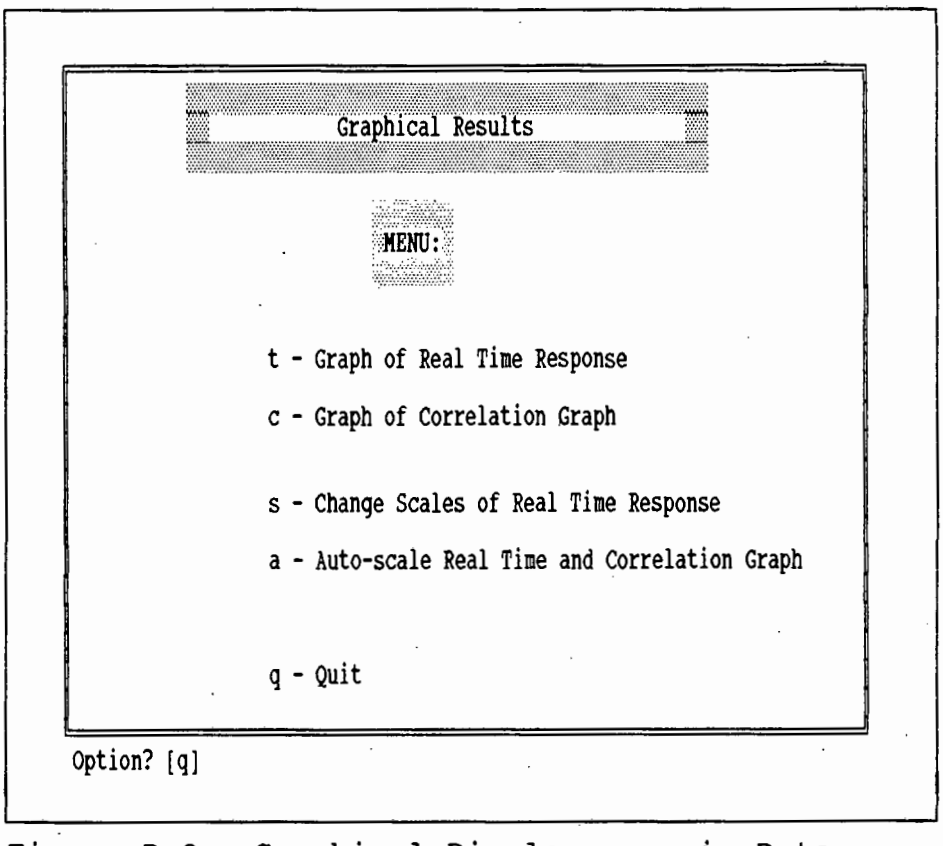


Figure D.8: Graphical Display menu in Data Processing option

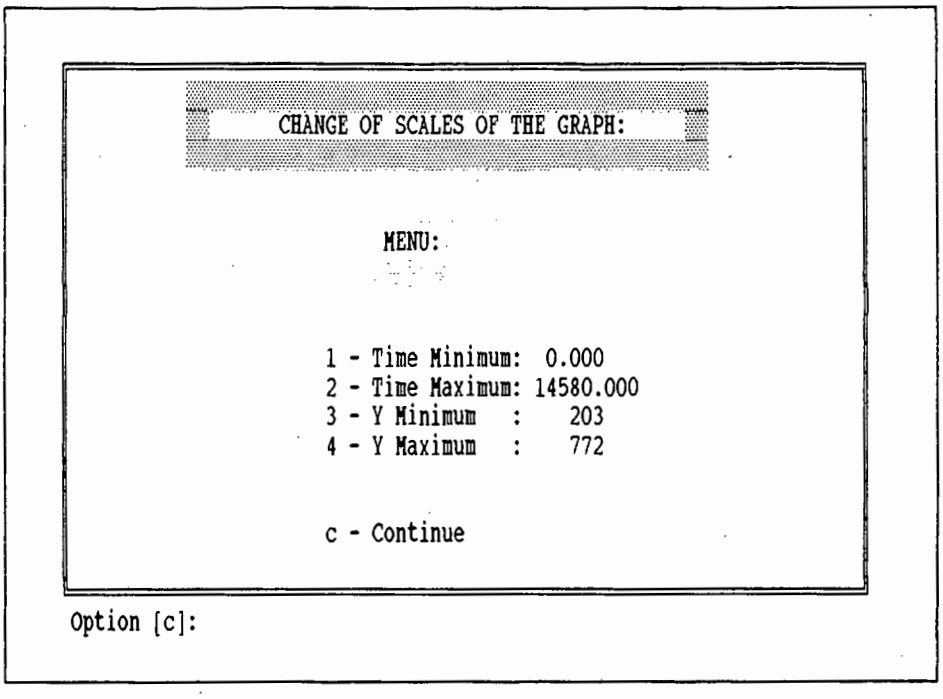


Figure D.9: Change Scale menu in Data Processing option

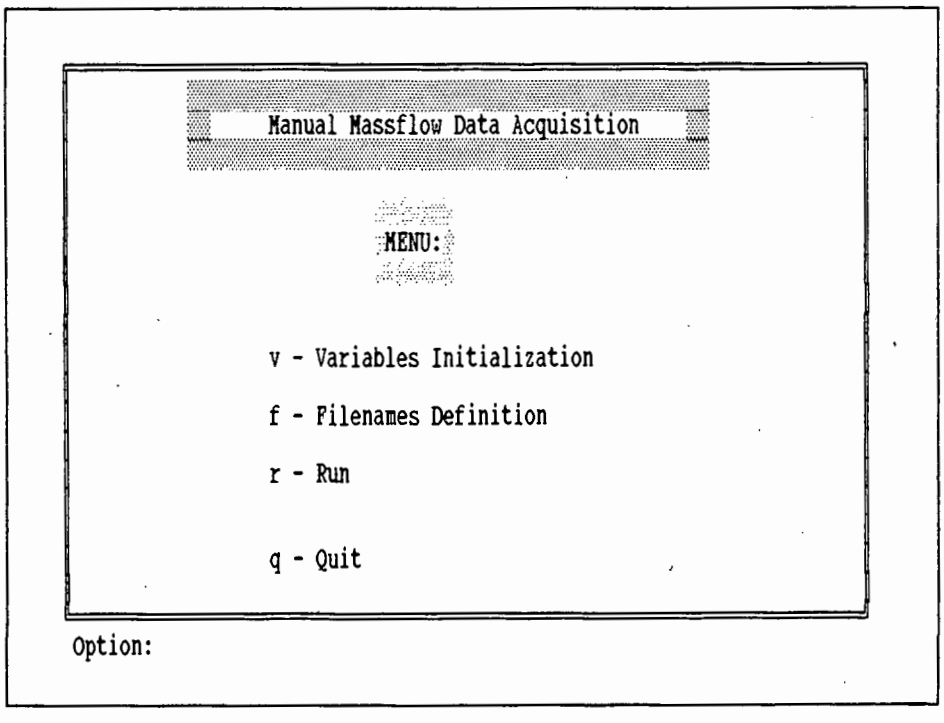


Figure D.10: Menu for Manual Massflow Data Acquisition using real time responses

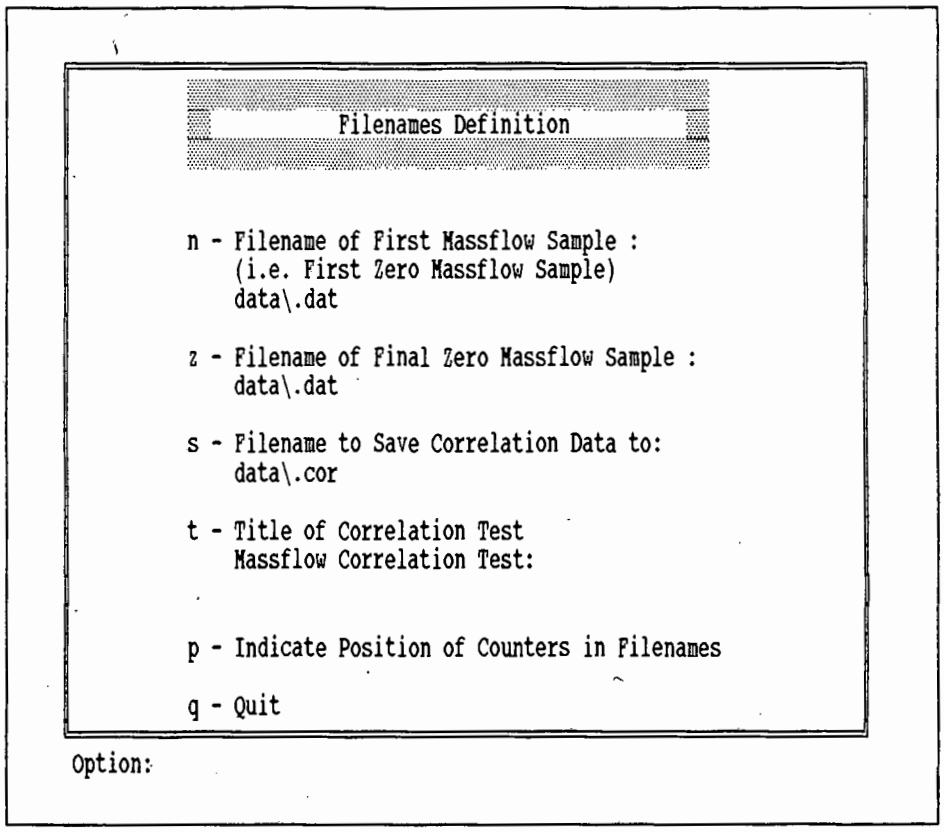


Figure D.11: Menu to define filenames in Manual Data Acquisition

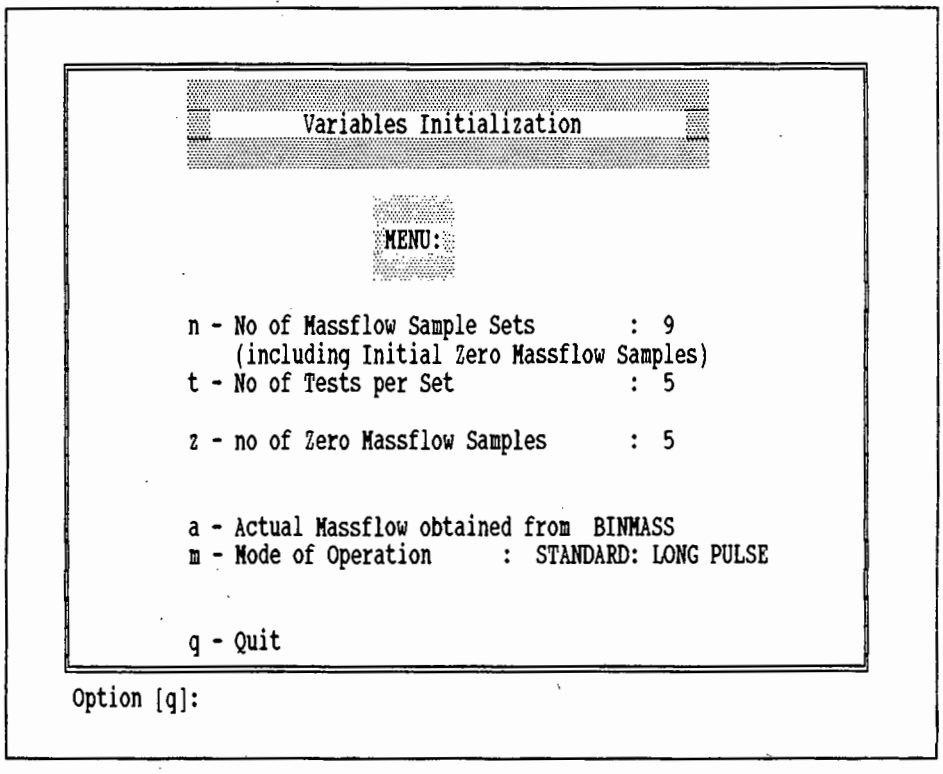


Figure D.12: Menu to define variables in Manual Data Acquisition

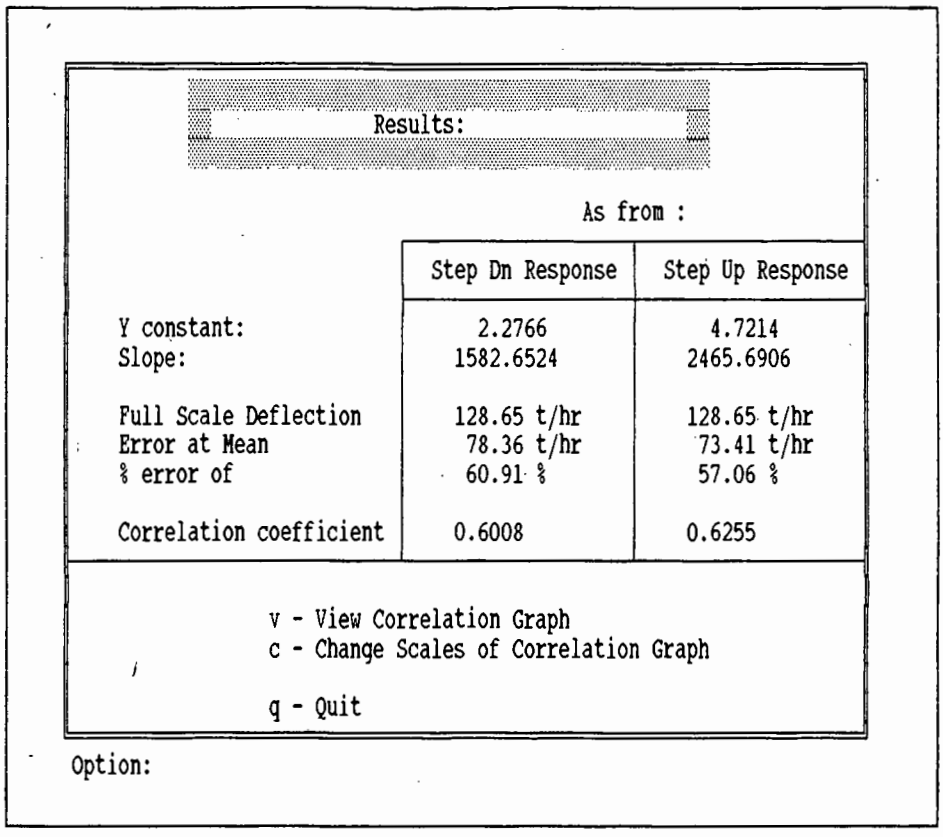


Figure D.13: Menu to display correlation results

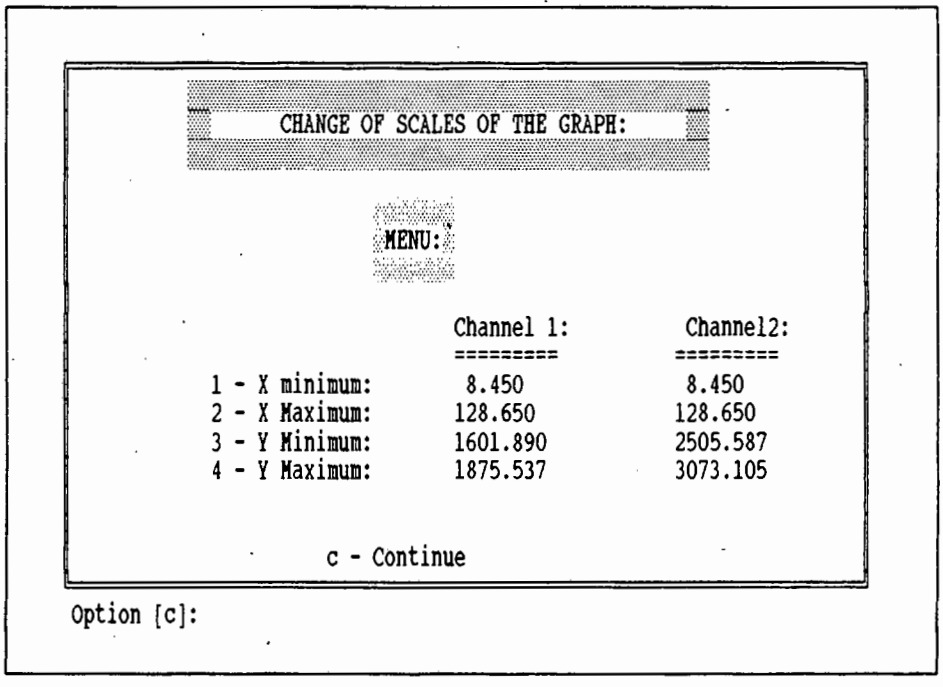


Figure D.14: Menu to change scales of correlation graph

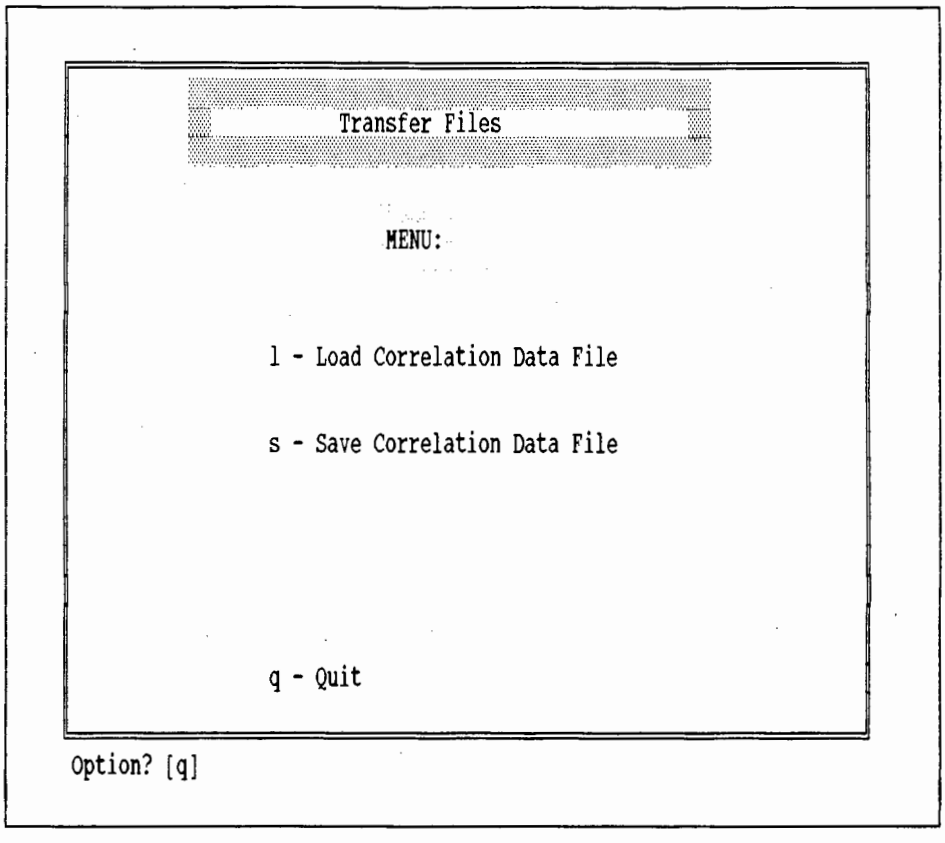


Figure D.15: Transfer Files menu

D.3: GRAF.EXE

D.3.1 Basic Menu Structure:

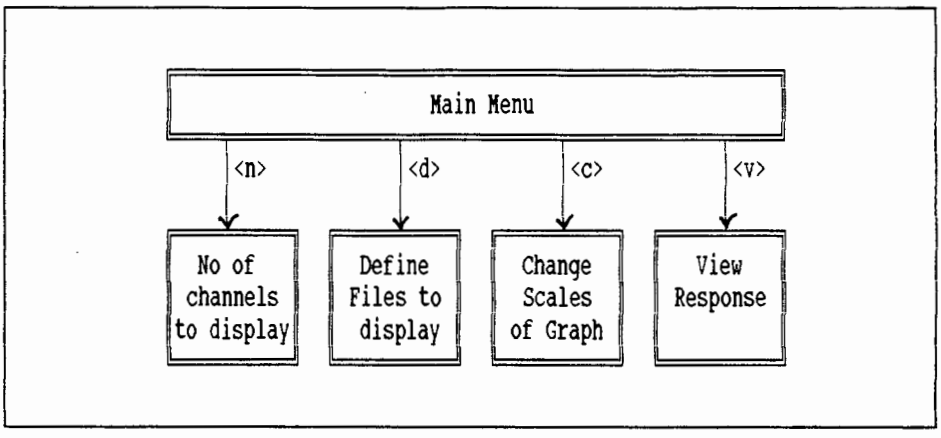


Figure D.16: Menu structure of GRAF.EXE

D.3.2 Description of Available Options:

No of channels to display: Specify the number of channels to be plotted

Define files to display: The following must be specified for each channel:

1) Label describing the response displayed in this channel.
2) The filename of the file which contains the response.
3) The channel in the file which contains the response

Change Scales of graph: Change the scales of the plot.

View Response: Plots the response in the specified way on the screen

D.3.3 Copies of the Menus used:

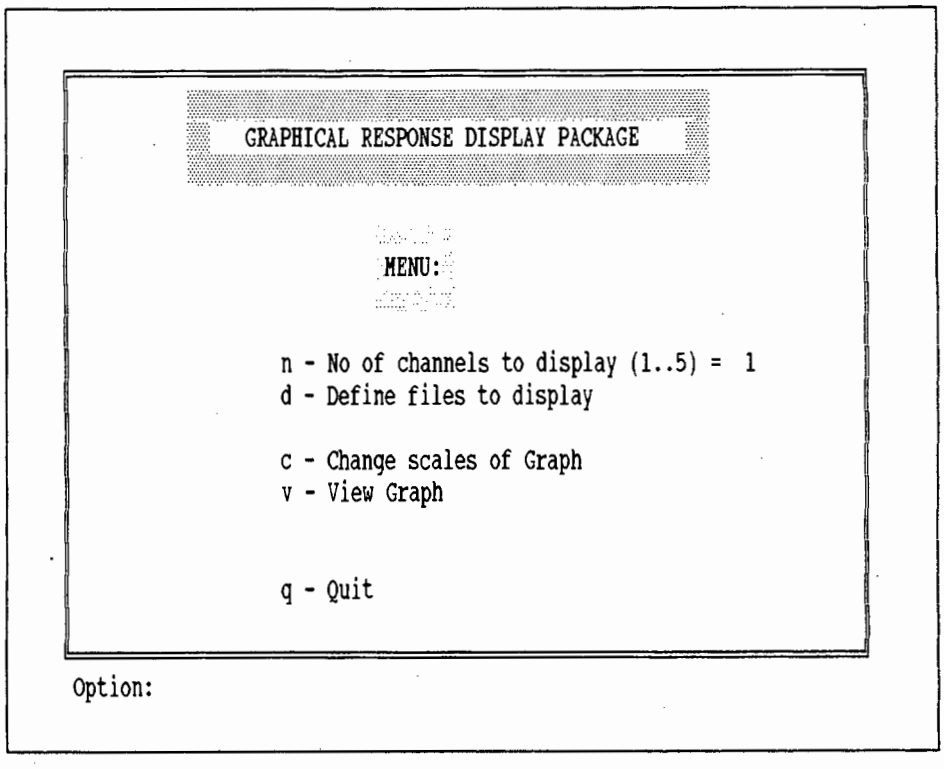


Figure D.17: Main menu of GRAF.EXE

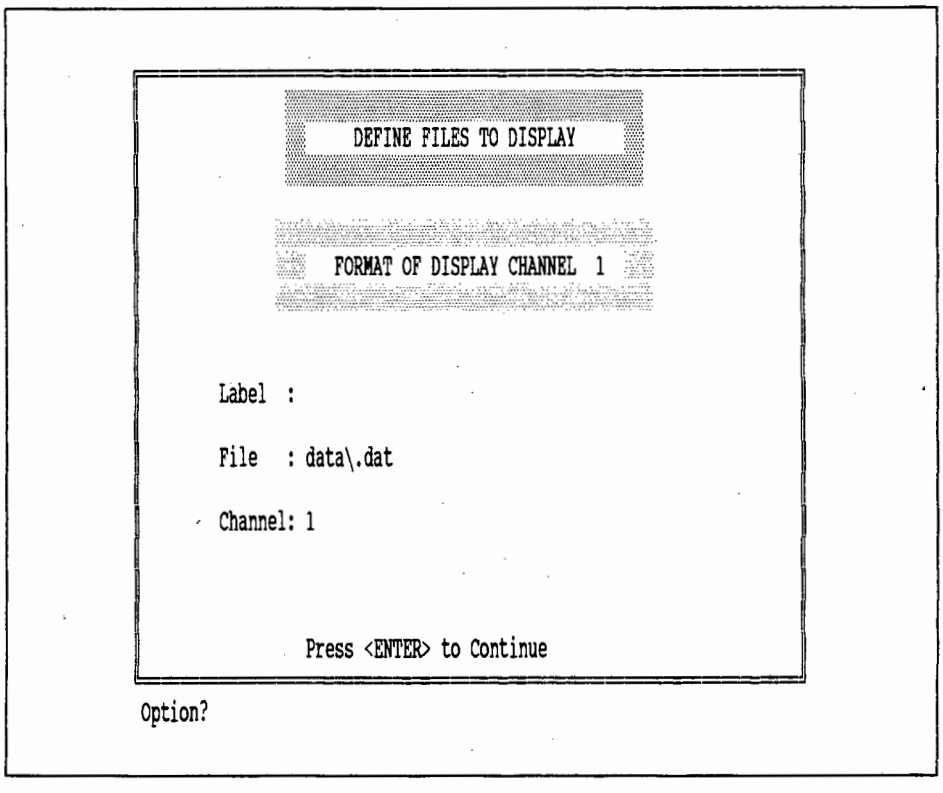


Figure D.18: Menu to define files to display

D.4: MFCA.EXE

D.4.1 Basic Menu Structure:

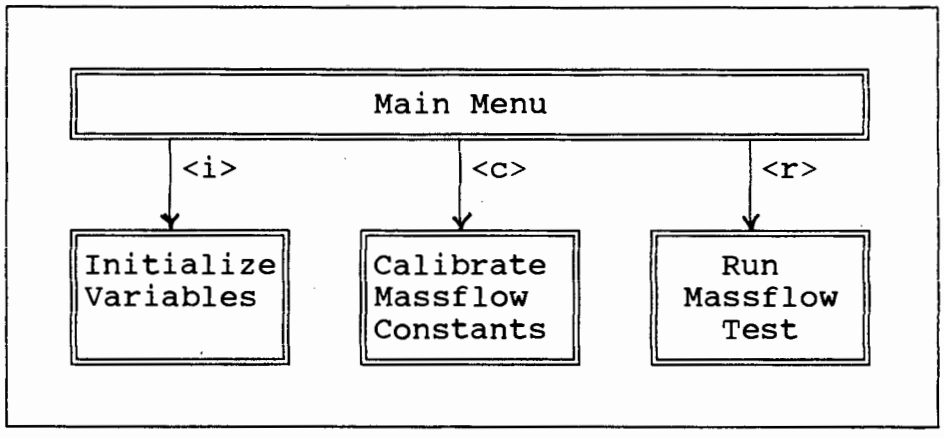


Figure D.19: Menu structure of MFCA.EXE

D.4.2 Description of Available Options:

Initialize Variables: The following Variables need to be specified:

- <n> - Total no of massflow samples
- <z> - No of zero massflow samples if reference massflow estimate is obtained from mass in the hopper bin
- <s> - Massflow sampling time
- <p> - Steady state setpoint
- <a> - Source of reference massflow estimate
- <m> - Toggle between mode of operation:
  - VSD: Belt stopped
  - VSD: Belt only slowed down
  - SSR: Belt stopped
  - SSR: Belt only slowed down
- <f> - Filename of where the massflow correlation data should be saved to
- <e> - Calibrate timing of data capture routine

Calibrate Massflow Constants: To run this option the password 'LEXI' needs to be entered. Thus the following constants can be changed:

<s> - [m/sec] per [PC count]
<p> - [Watts] per [PC count]
<l> - physical length of belt [m]
<e> - Error at mean (95% confidence bands) [tons/hour]
<x> - Offset of measurement to be subtracted [tons/hour]. This value can be determined automatically if the belt is run empty.
<y> - Slope of calibration curve [t/hr/t/hr]. This value can also be determined automatically by running the belt at full load, performing a massflow test, then doing a belt cut and entering the unit mass on the belt [kg/m] into the package. [NOTE: to determine the slope automatically the offset has to be calibrated first].

Run Massflow Test: The two flowcharts below illustrate the procedure of this option. The first flowchart explains the procedure if the reference massflow estimate is obtained from the mass in the hopper bin, and the second flowchart explains the procedure if the reference massflow is obtained from a massflow meter.

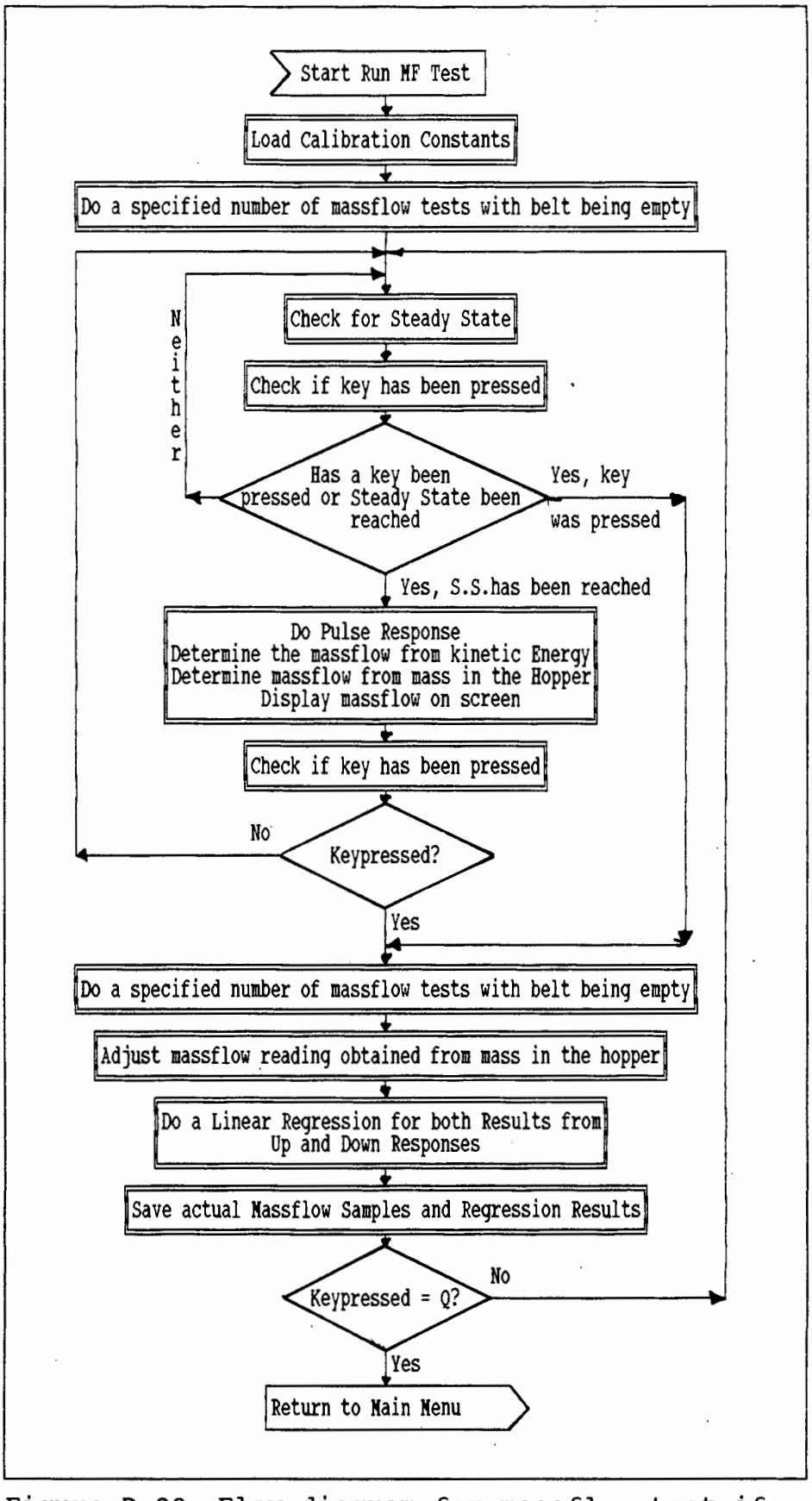


Figure D.20: Flow diagram for massflow test if reference massflow is obtained from mass in the hopper bin

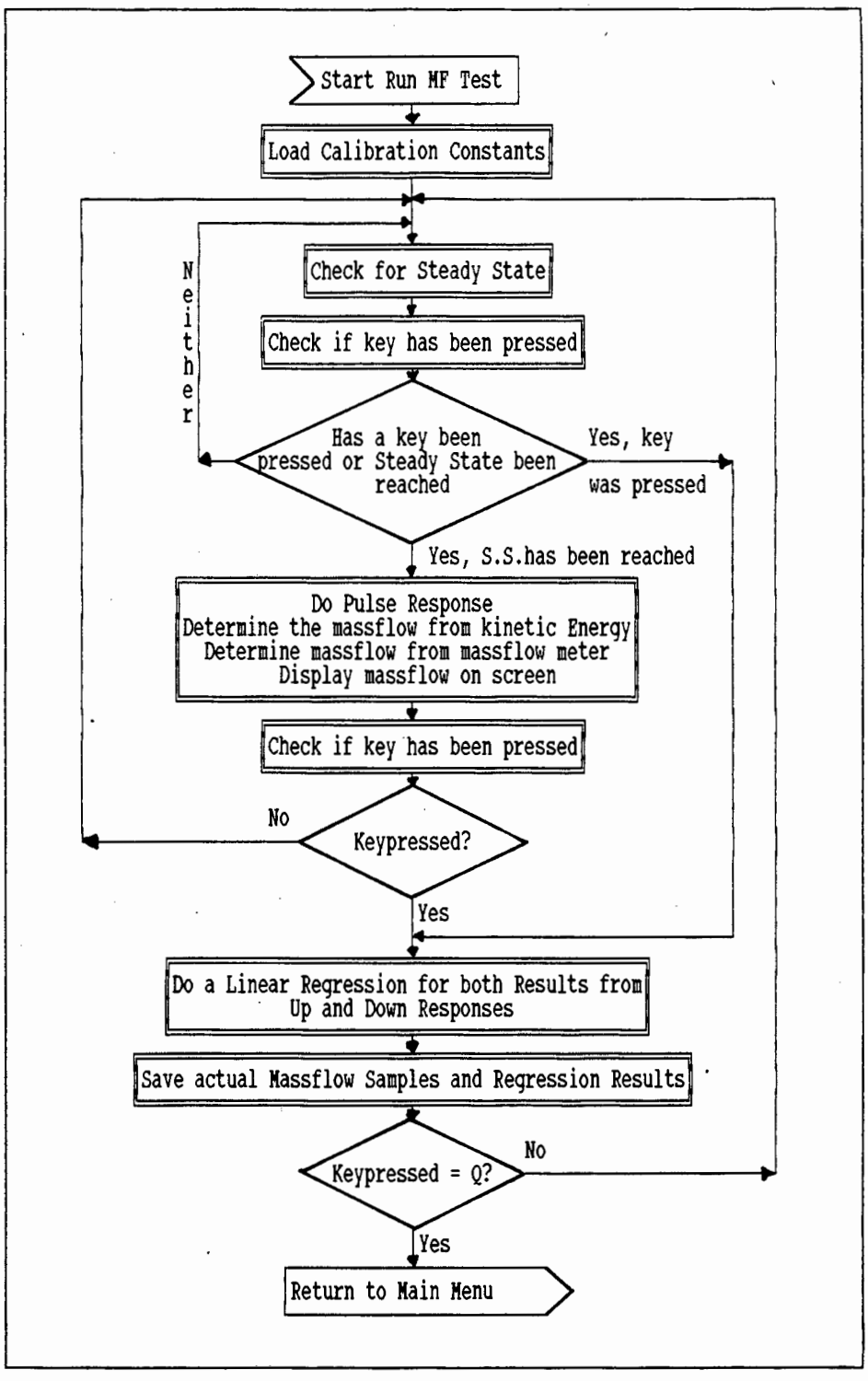


Figure D.21: Flow diagram for massflow test if reference massflow is obtained from a massflow meter

D.4.3 Copies of the Menus used:

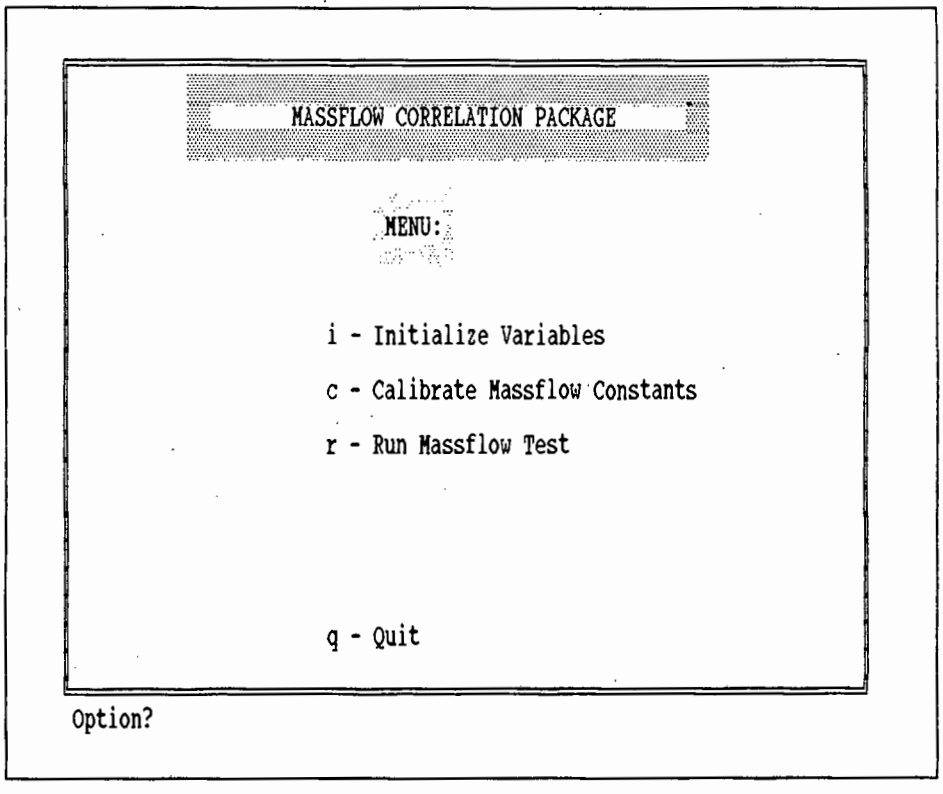


Figure D.22: Main menu of MFCA.EXE

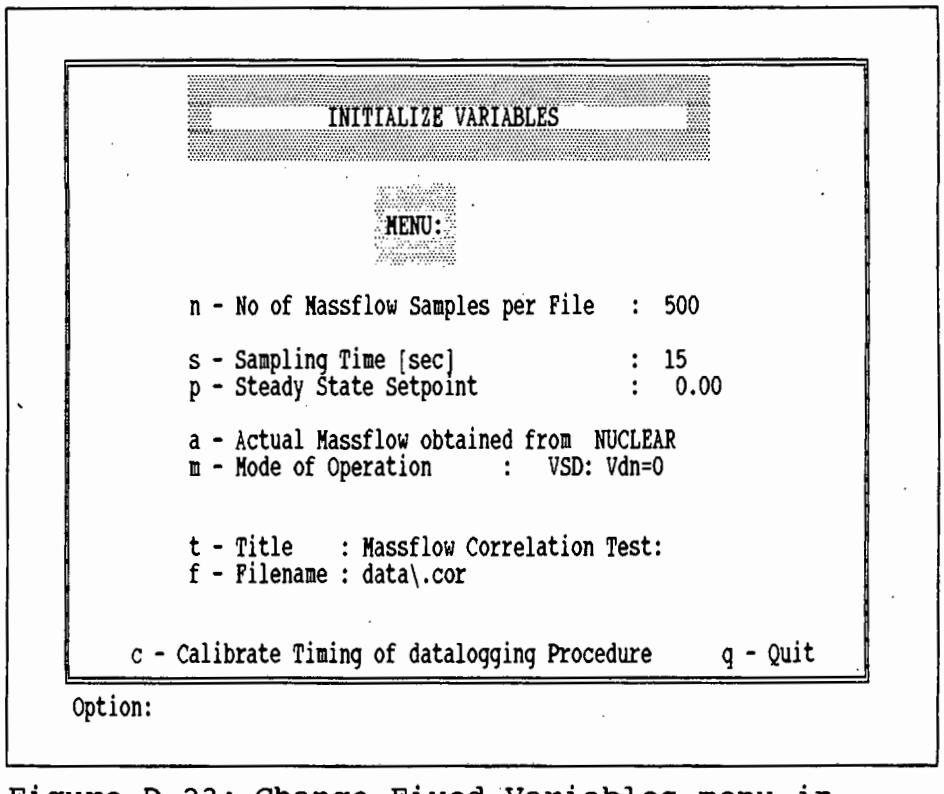


Figure D.23: Change Fixed Variables menu in MFCA.EXE

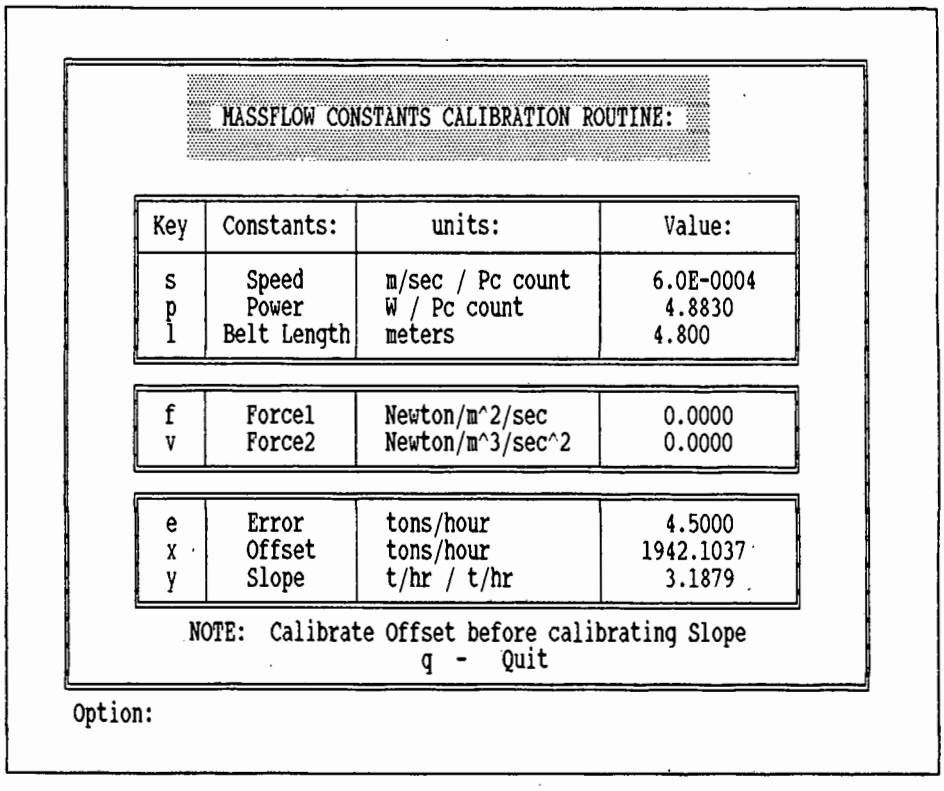


Figure D.24: Menu to calibrate massflow constants

D.5: MF.EXE

D.5.1 Basic Menu Structure:

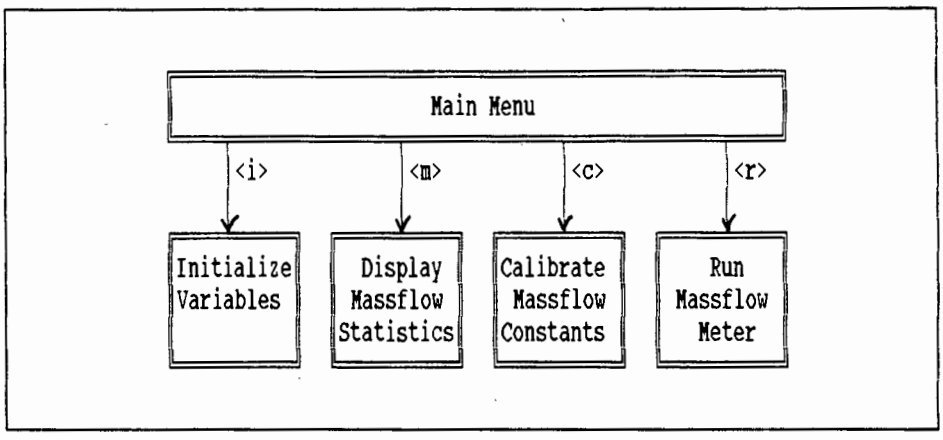


Figure D.25: Menu structure of MF.EXE

D.5.2 Description of Available Options:

Initialize Variables: The following Variables need to be specified:

<n> - Total no of massflow samples
<s> - Massflow sampling time
<p> - Steady state setpoint
<m> - Toggle between mode of operation:
- VSD: Belt stopped
- VSD: Belt only slowed down
- SSR: Belt stopped
- SSR: Belt only slowed down

<t> - Title of massflow test
<f> - Filename of where the massflow data should be saved to
<c> - Calibrate timing of data capture routine

Massflow Statistics: In this option the average, total, maximum and minimum massflow can be determined for a specified interval in time.

Calibrate Massflow Constants: Options in this menu are exactly the same as in MFCA.EXE.

Run Massflow Meter: The flowchart down below explains the procedure of this option.

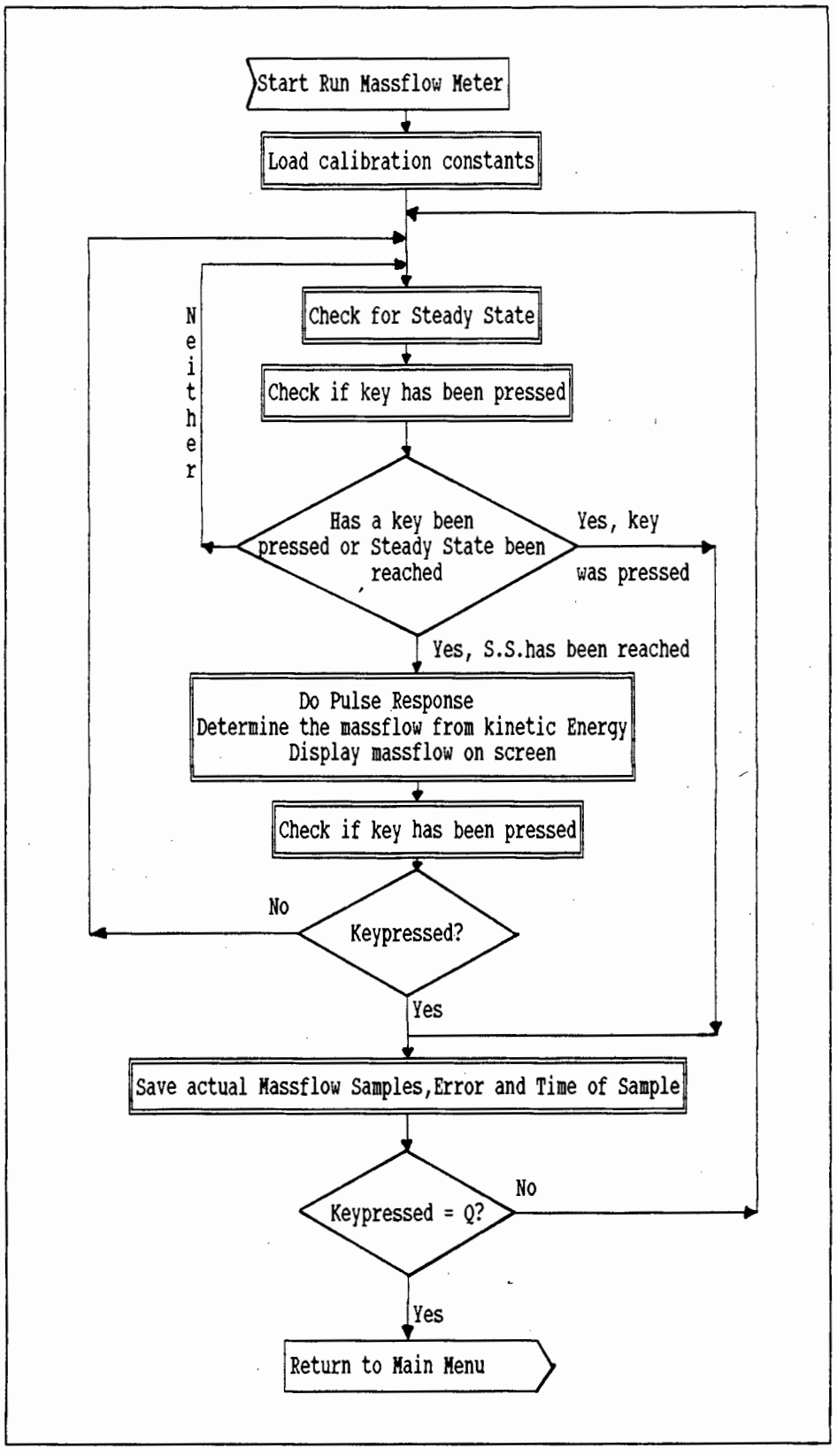


Figure D.26: Flow diagram for massflow correlation of Massflow

E.5.3 Copies of the Menus used:

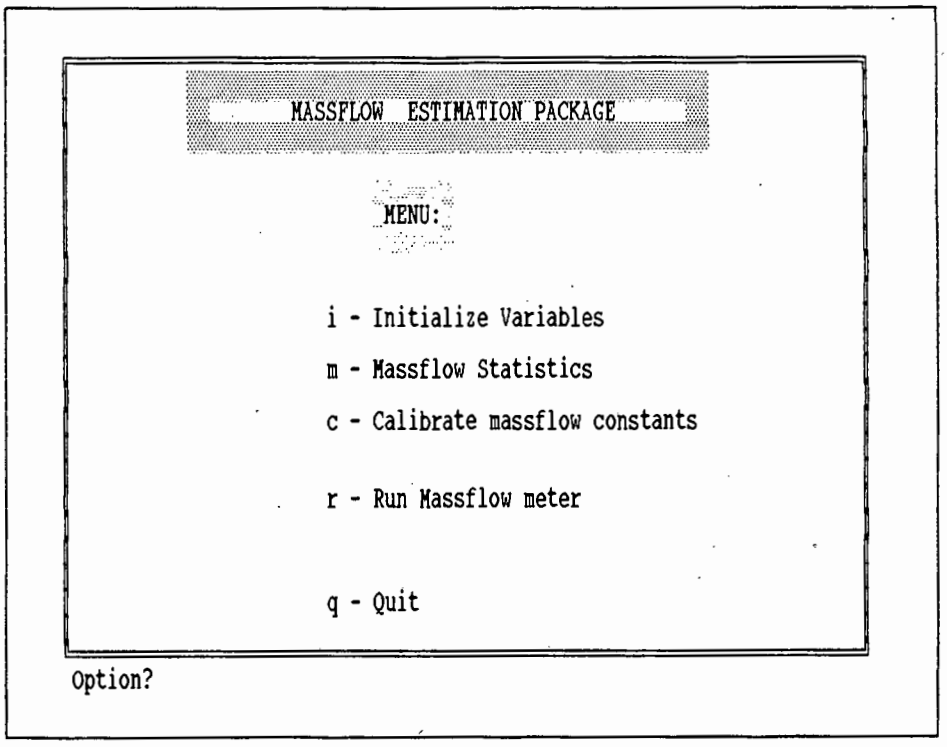


Figure D.27: Main menu of MF.EXE

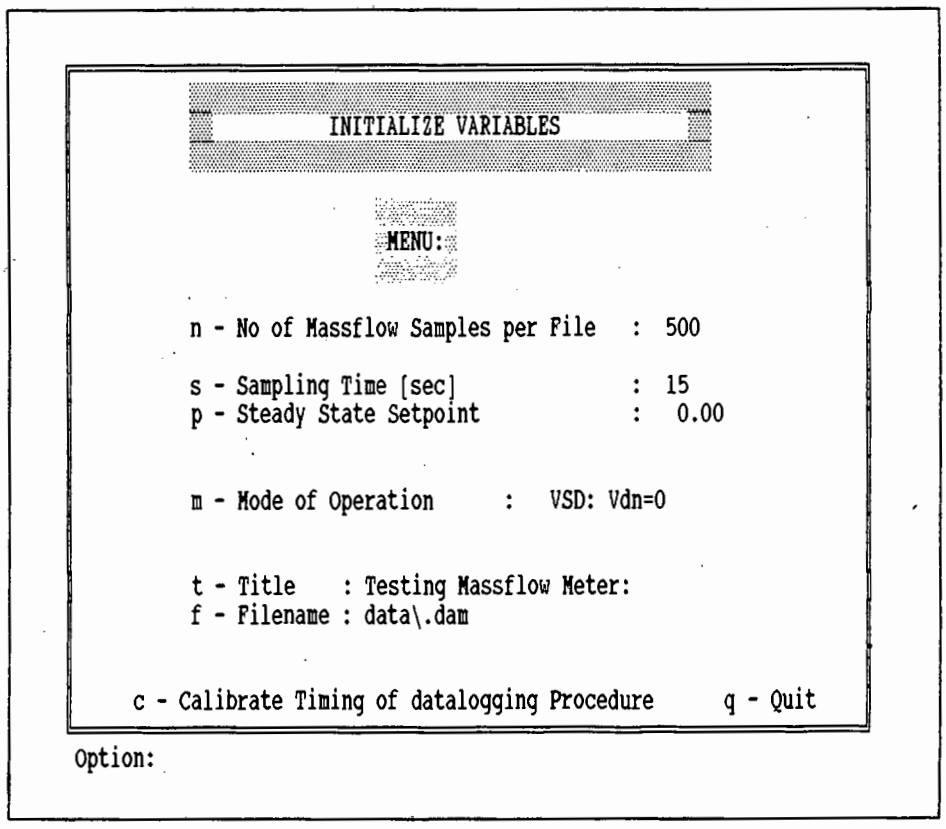


Figure D.28: Change Fixed Variables menu

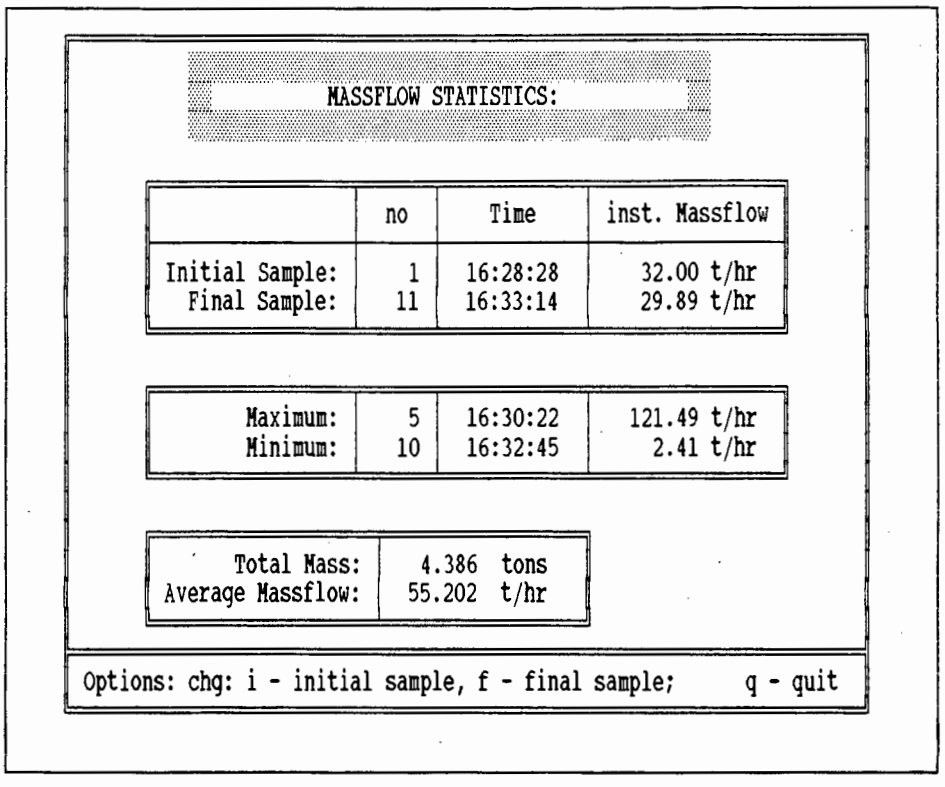


Figure D.29: Statistical Data display menu

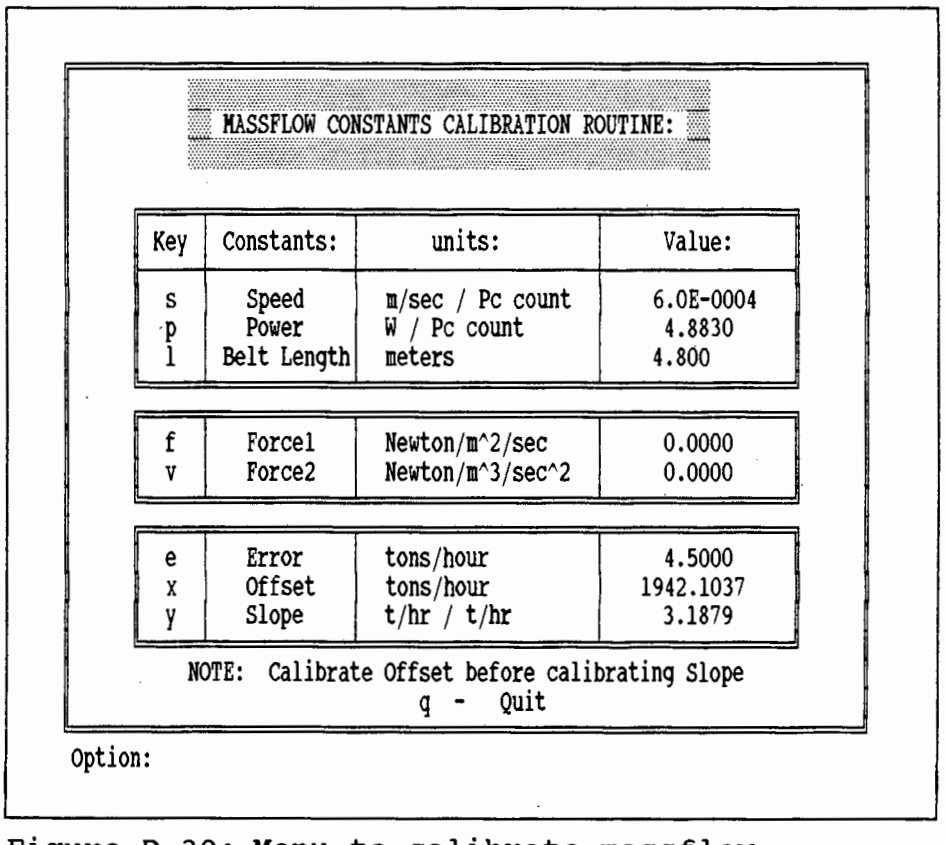


Figure D.30: Menu to calibrate massflow constants

APPENDIX E:
DATA FORMATS USED IN SOFTWARE

E.1 DATA A: REAL TIME RESPONSE DATA FORMAT

Filenames: *.dat

```
  501     1     3     100      -10.00000  (No of samples; First channel; No of channels; When to apply step;
   50.0000                               ( Sampling frequency )                                Size of step)
      SPEED      POWER    TONNAGE         ( Legends for channels 1 to 3)
MASSFLOW: 100% Speed Perturbation: C1099011     ( Title of response)
 1744   222    76                        ( Data logged: Channel 1; Channel 2; Channel 3; ..... )
 1754   214    75
 1765   211    71
 1760   219    72
 1746   225    77
 1747   223    75
 1750   225    69
 1764   232    69
 1772   232    74
         :
         :
         :
         :
( to no of Samples)
```

E.3 DATA C: MASSFLOW DATA FORMAT

Filenames: *.dam

```
Title: Testing Massflow Meter: Vxx9911
No of samples:      Error [t/hr]:
  11                    4.50

  Massflow    Time
  [t/hr]      [hrs:min:sec]
   32.00       16:28:28
   28.39       16:28:57
   29.80       16:29:25
   21.96       16:29:54
  121.49       16:30:22
  108.50       16:30:51
   94.31       16:31:19
   66.60       16:31:48
   48.97       16:32:16
    2.41       16:32:45
   29.89       16:33:14
                    :
          :
          :
          :
   ( to no of samples)
```

APPENDIX F:
CIRCUITS USED FOR LABORATORY TEST RIG

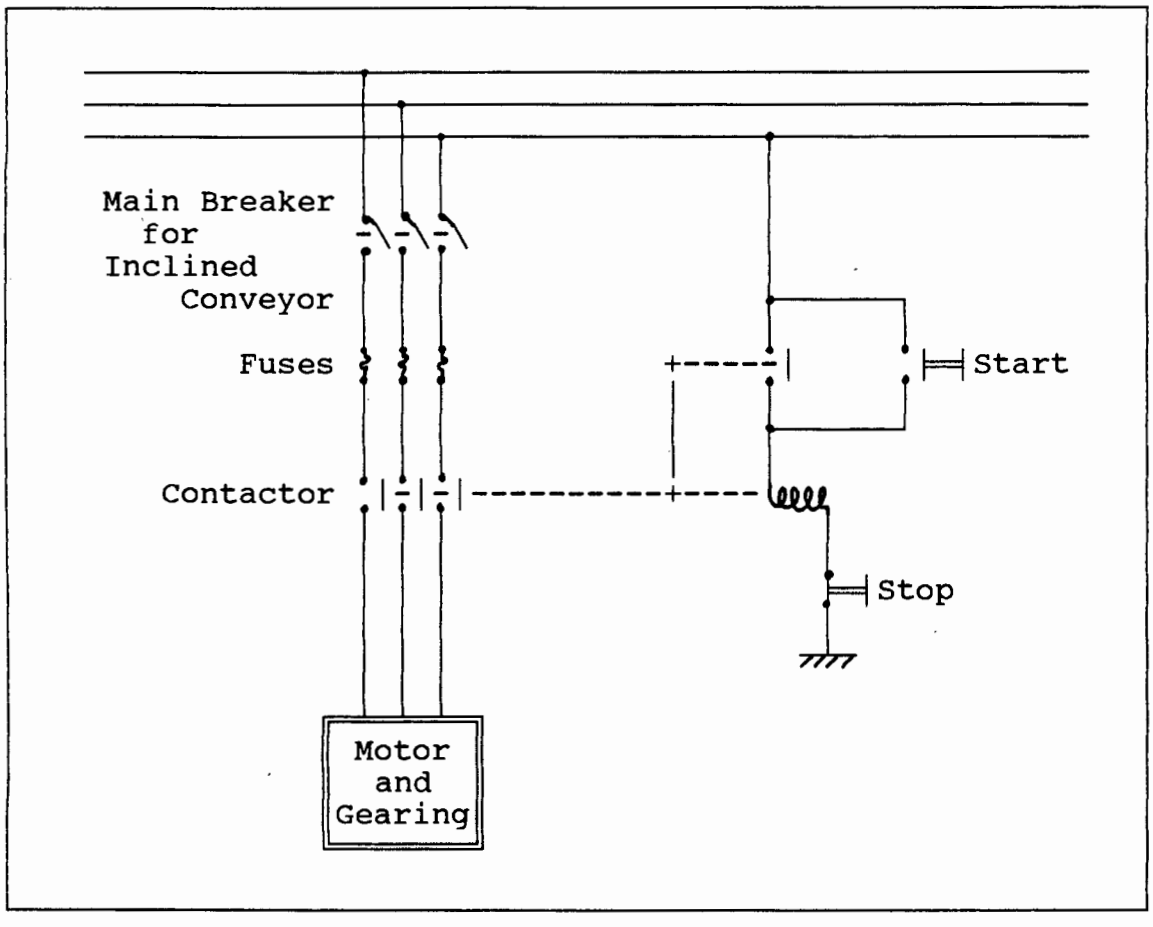


Figure F.1: Circuit diagram for the 380 V network of the inclined conveyor belt

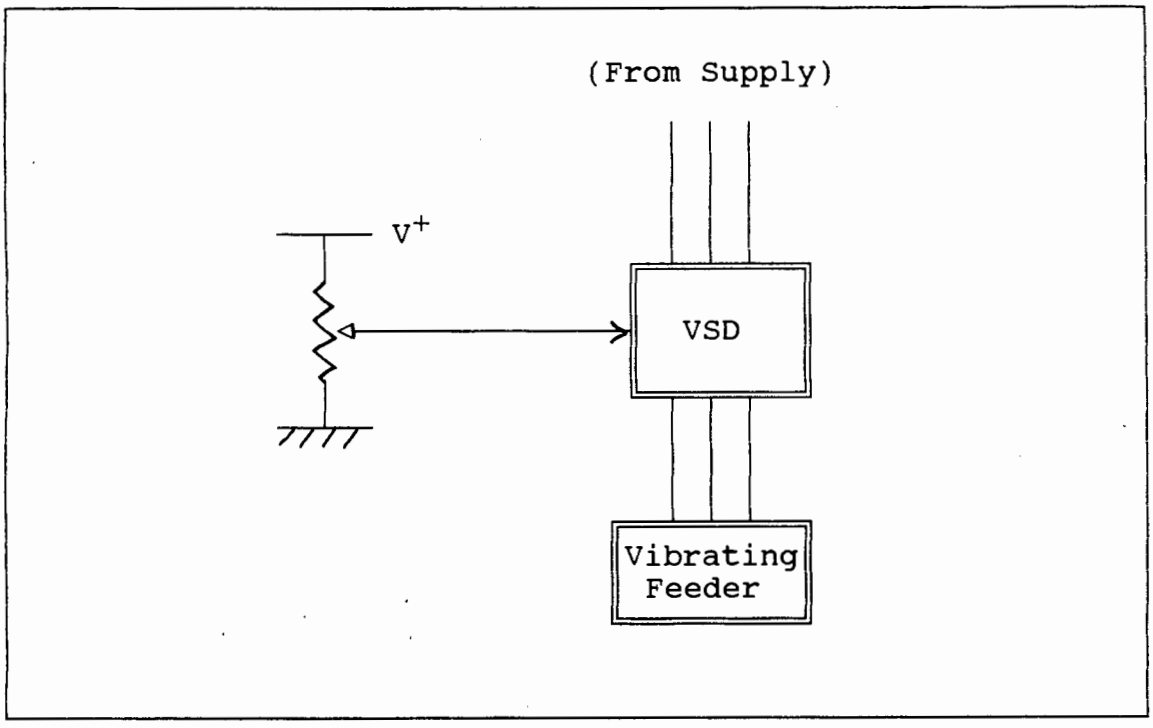


Figure F.2: Circuit for vibrating feeder of horizontal conveyor belt

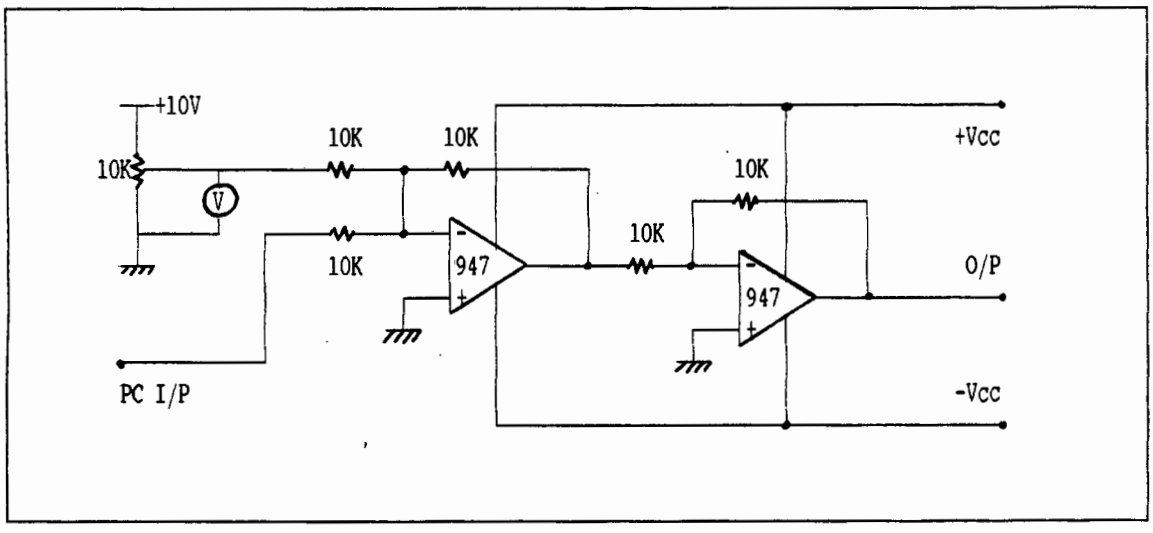


Figure F.3: Circuit diagram of op-amp adding circuit

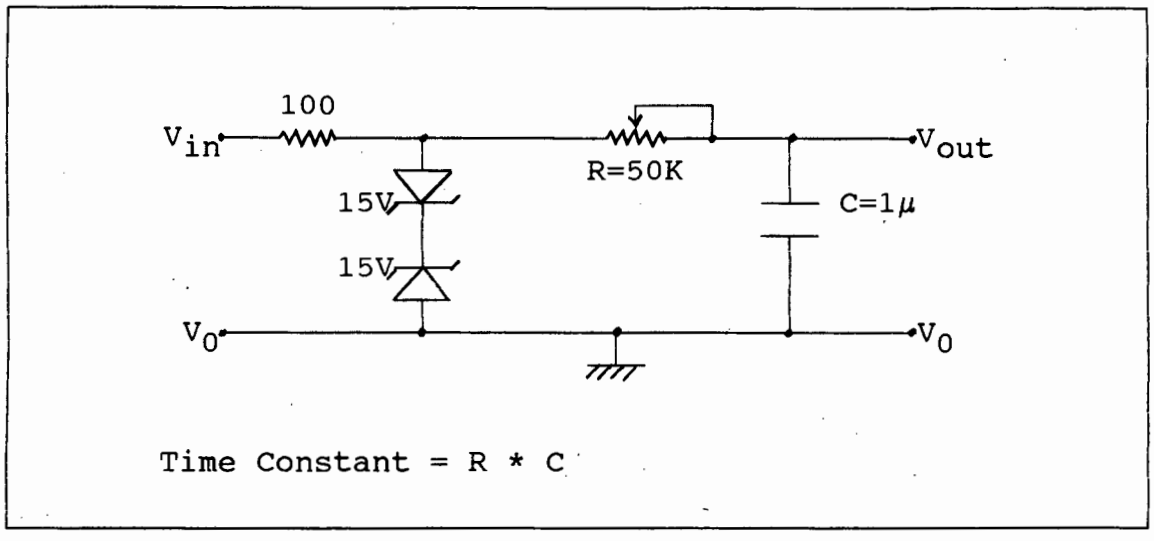


Figure F.4: Anti-aliasing filter and A/D protection

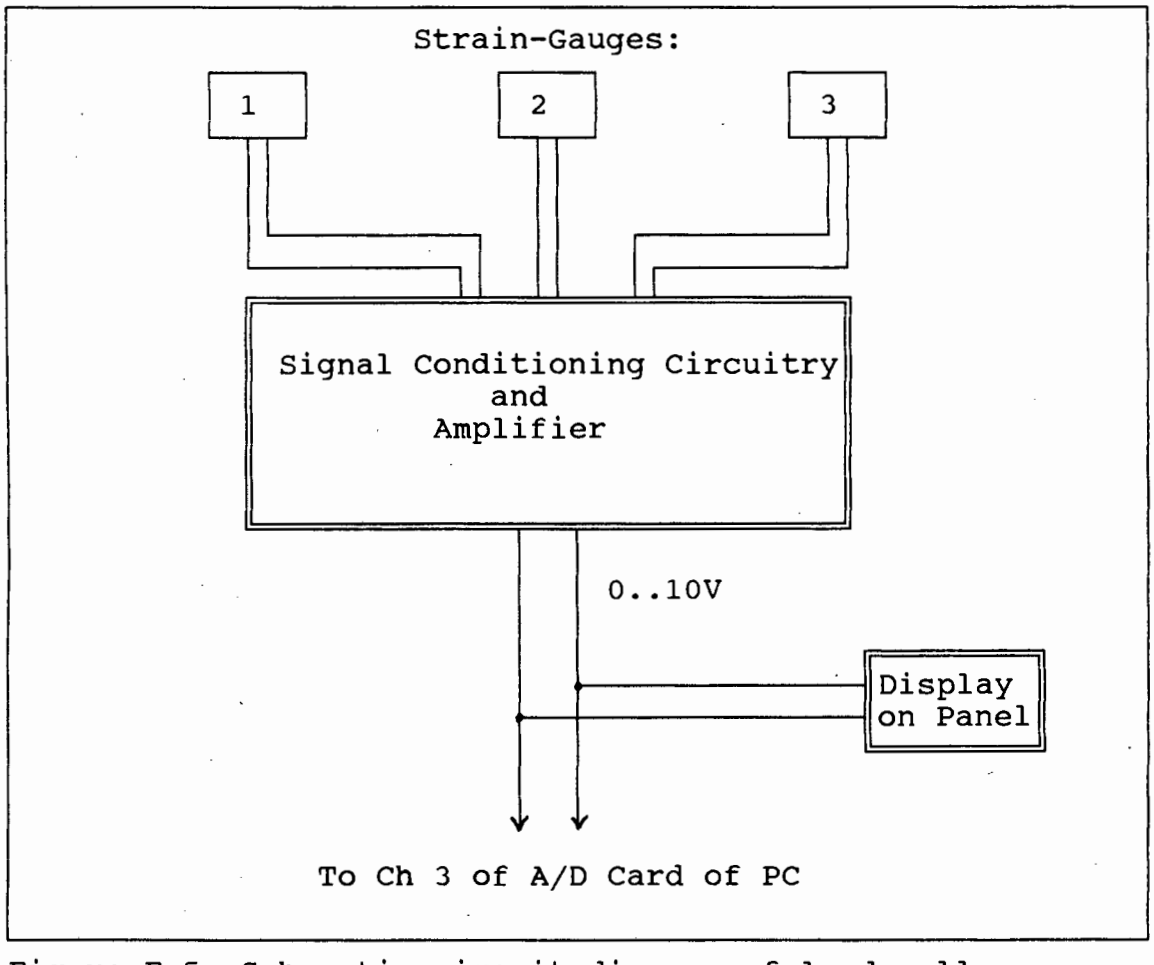


Figure F.5: Schematic circuit diagram of load cell amplifier

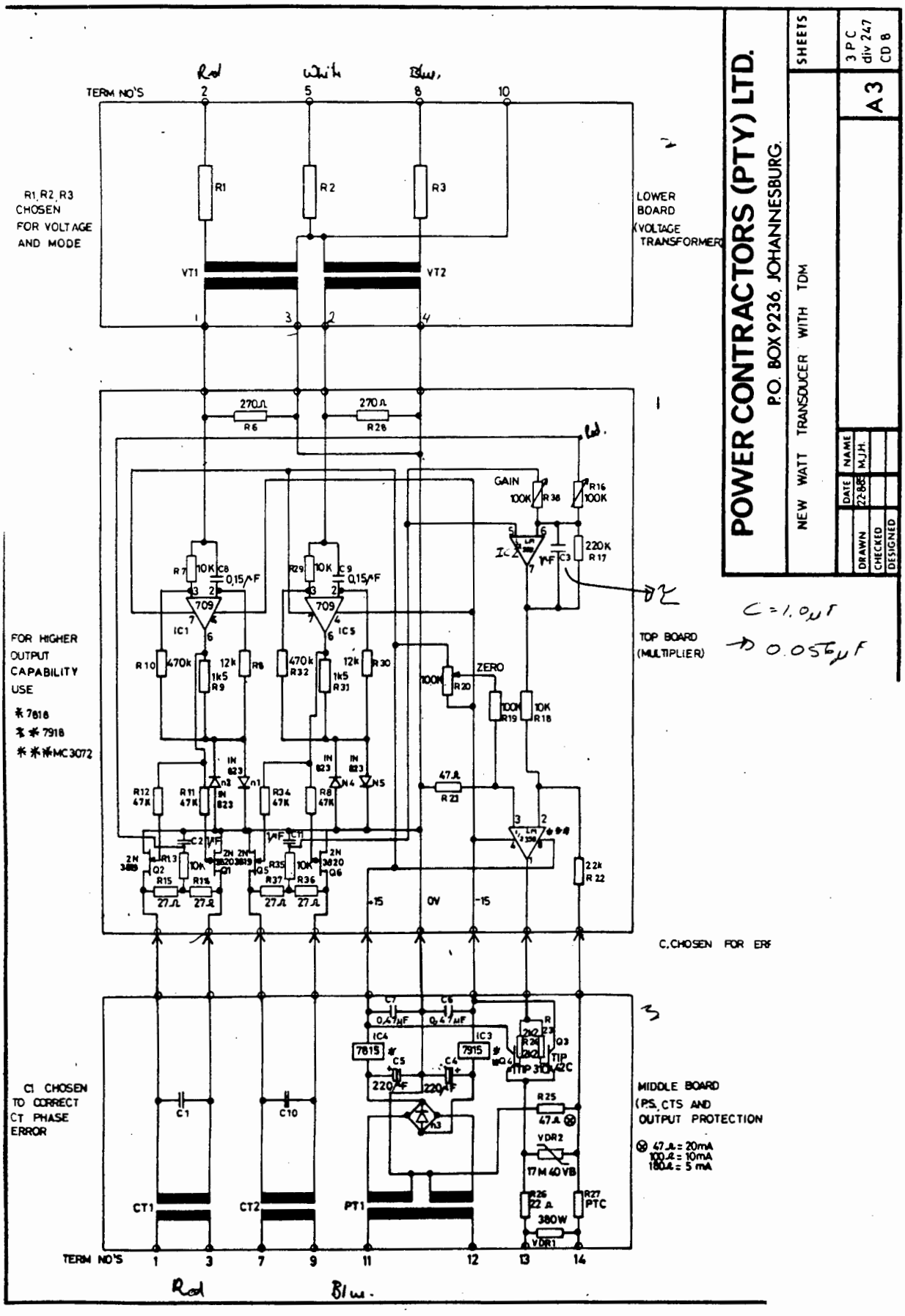


Figure F.6: Circuit diagram of power transducer

APPENDIX G:
CALIBRATION RESULTS OF LABORATORY TEST RIG

G.1 CALIBRATION AND REGRESSION RESULTS FOR SPEED

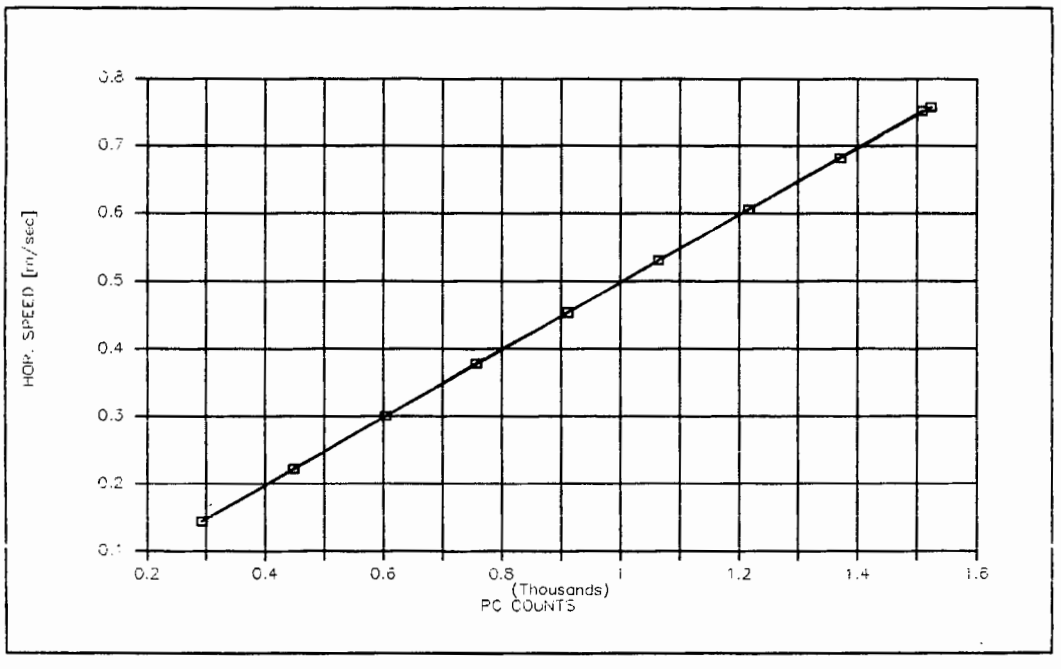


Figure G.1: Calibration of speed for laboratory test rig

HORIZONTAL SPEED / PC COUNT

Regression Output:

Constant	-0.00060
Std Err of Y Est	0.000679
R Squared	0.999991
No. of Observations	10
Degrees of Freedom	8
X Coefficient(s)	0.000498809
Std Err of Coef.	0.000000515

Table G.1: Regression results from speed calibration

G.2 CALIBRATION AND REGRESSION RESULTS FOR POWER

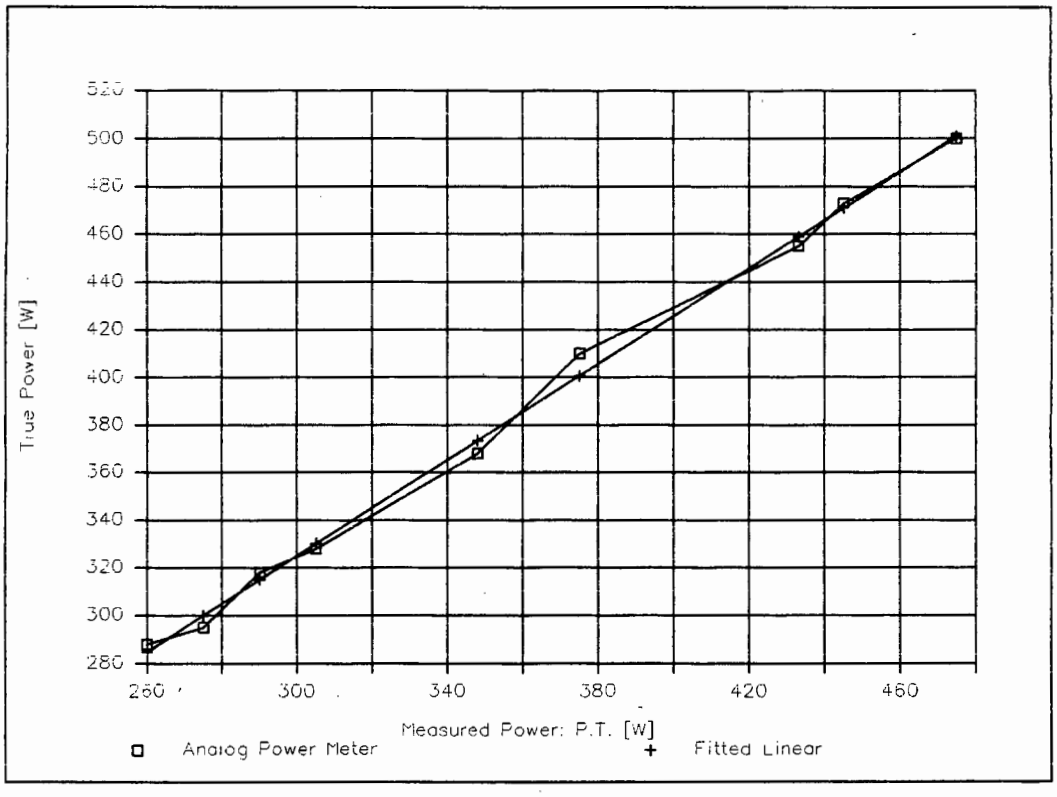


Figure G.2: Calibration graph for power transducer when system is operating with VSD

TRUE POWER vs. PT

Regression Output:

Constant	23.452651
Std Err of Y Est	5.1633661
R Squared	0.9964062
No. of Observations	9
Degrees of Freedom	7
X Coefficient(s)	1.0055914
Std Err of Coef.	0.0228258

Table G.2: Regression results for calibration of PT when system is running with the VSD

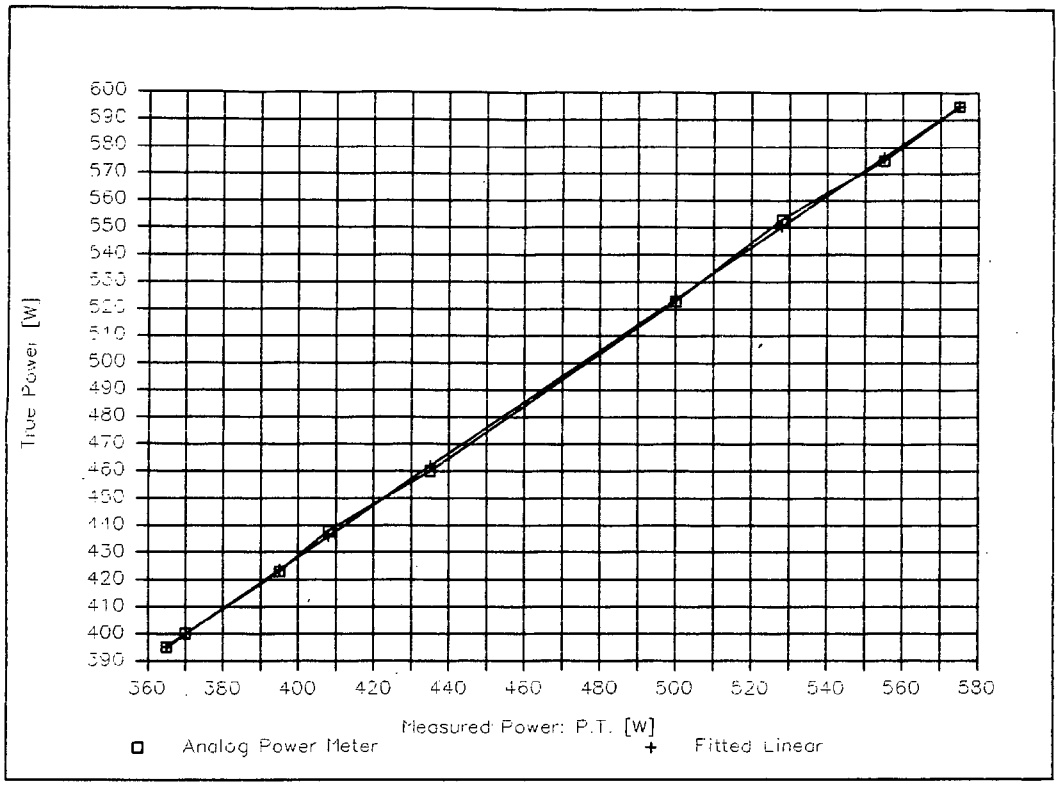


Figure G.3: Calibration graph for power transducer when system is operating in Direct Mode

TRUE POWER vs. PT

Regression Output:

Constant	47.29349
Std Err of Y Est	1.512725
R Squared	0.999668
No. of Observations	9
Degrees of Freedom	7
X Coefficient(s)	0.952882
Std Err of Coef.	0.006561

Table G.3: Regression results for calibration of PT when system is running in Direct Mode

G.3 CALIBRATION AND REGRESSION RESULTS FOR REFERENCE MASSFLOW

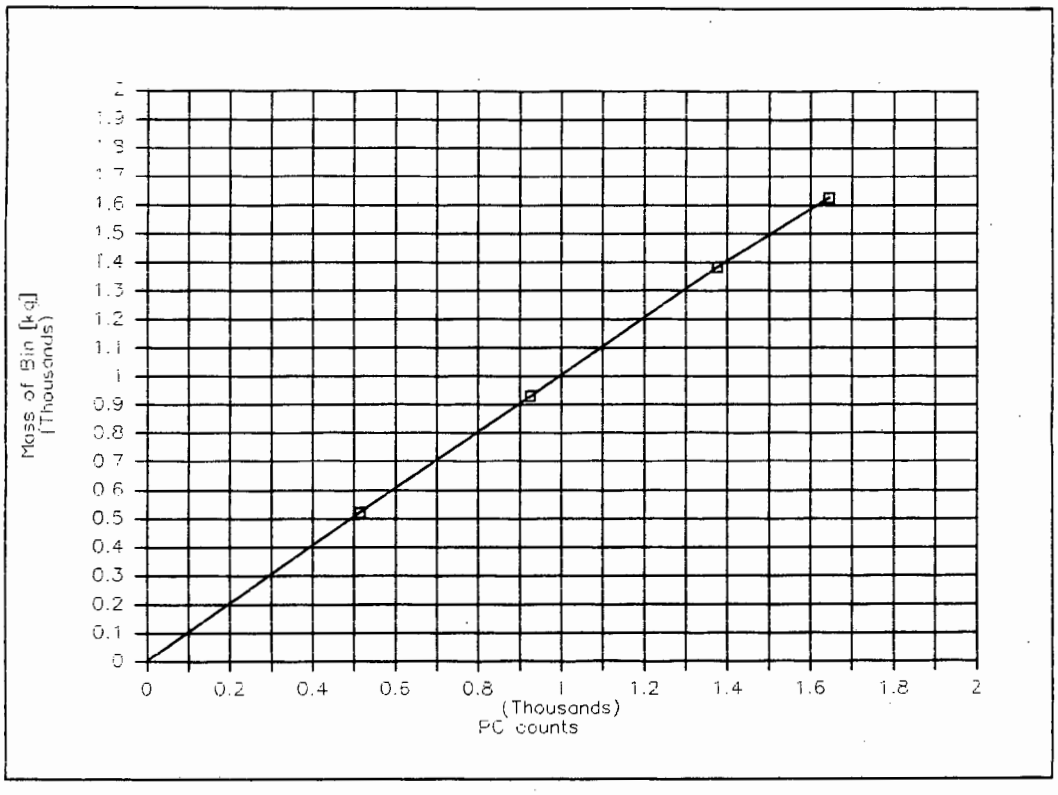


Figure G.4: Calibration graph of mass in hopper bin

MASS of BIN CALIBRATION Regression Output:	
Constant	9.635015
Std Err of Y Est	10.69089
R Squared	0.999800
No. of Observations	5
Degrees of Freedom	3
X Coefficient(s)	0.990929
Std Err of Coef.	0.008087

Table G.4: Regression results of calibration of mass of bin

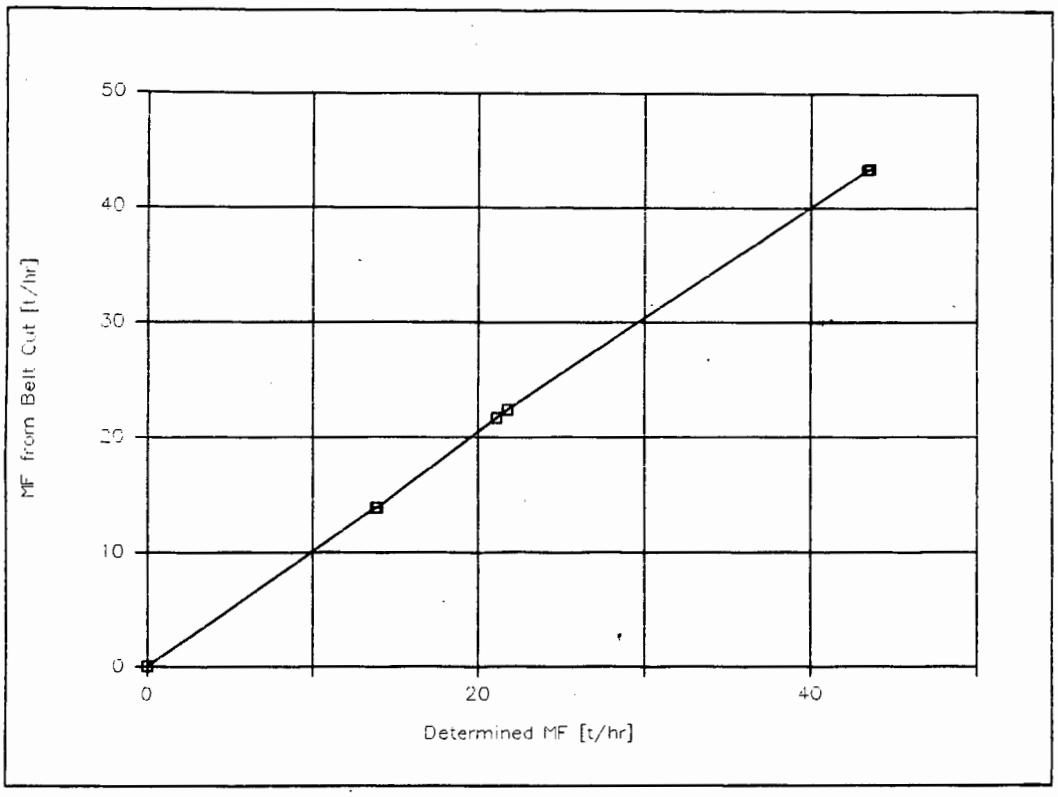


Figure G.5: Confirming calibration for massflow on horizontal conveyor belt

Regression Output:	
Constant	0.176521
Std Err of Y Est	0.331109
R Squared	0.999667
No. of Observations	8
Degrees of Freedom	6
X Coefficient(s)	0.998780
Std Err of Coef.	0.007437

Table G.5: Regression results from massflow confirmation

G.4 LOAD FRICTION MODEL

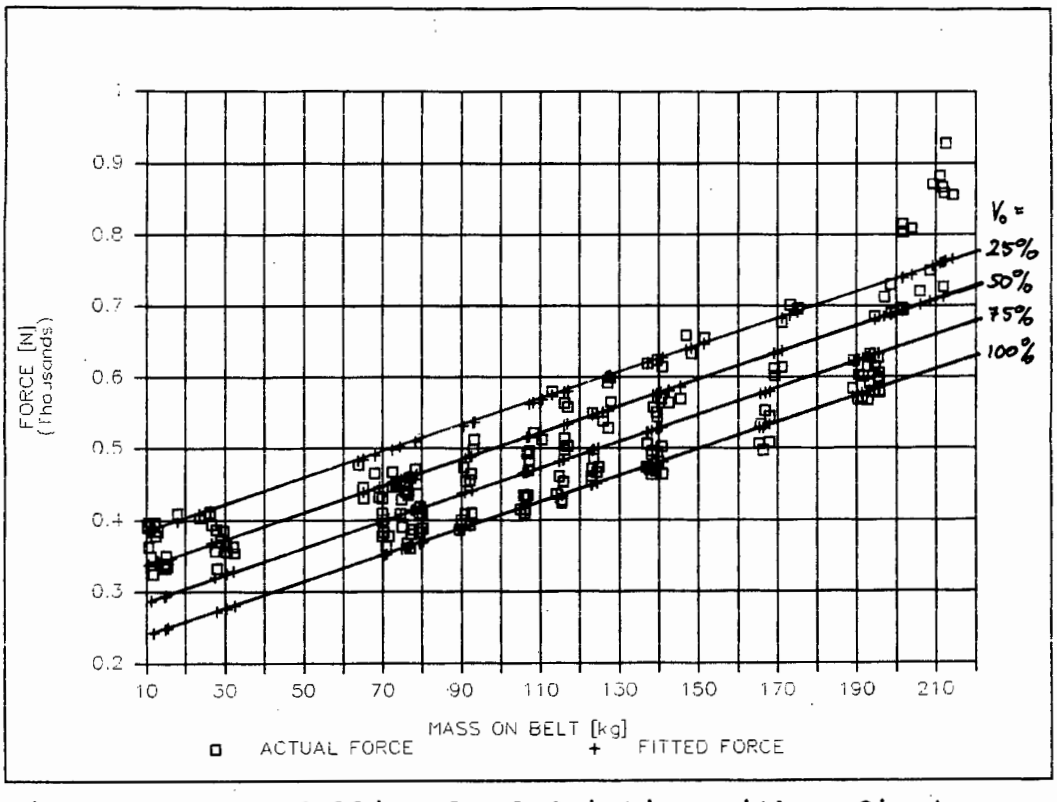


Figure G.6: Modelling load friction with a first order model

1st Order Regression

Regression Output:

Constant		415.8226
Std Err of Y Est		36.27267
R Squared		0.919345
No. of Observations		180
Degrees of Freedom		177
X Coefficient(s)	1.847472	-255.950
Std Err of Coef.	0.047076	13.02915

Table G.6: Regression results of modelling the load friction with a first order model

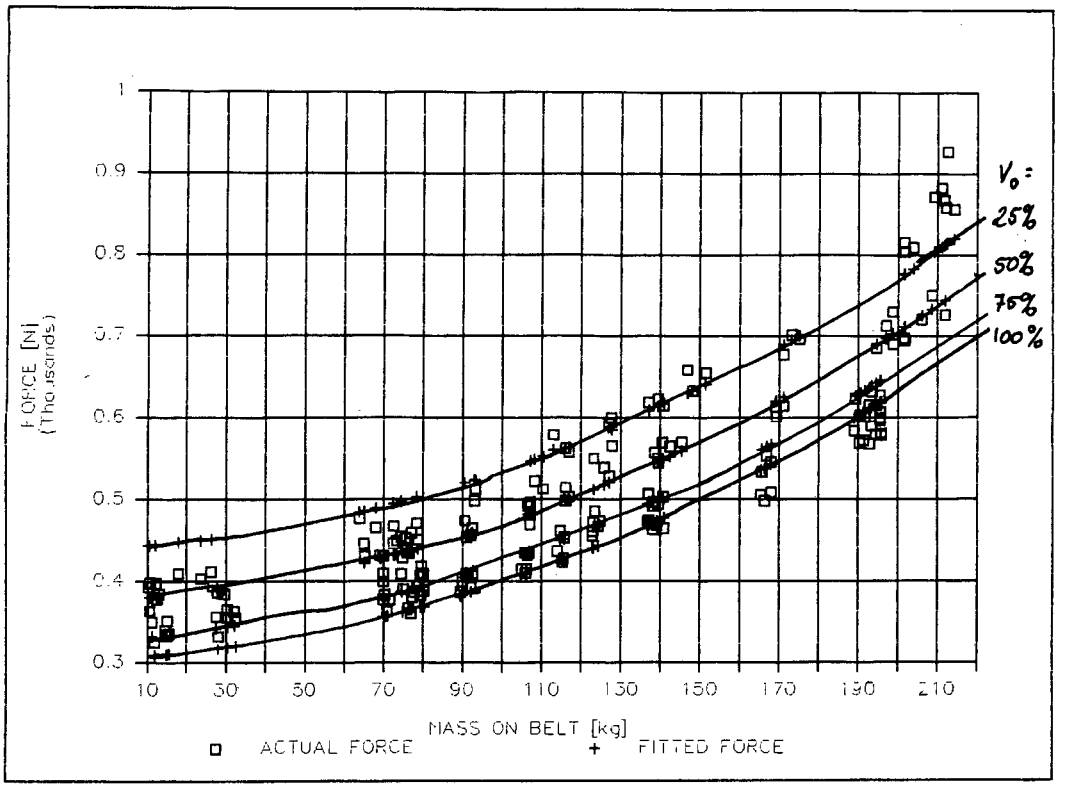


Figure G.7: Modelling load friction with a second order model

2nd Order Regression

Regression Output:

Constant	533.3329			
Std Err of Y Est	24.47488			
R Squared	0.963694			
No. of Observations	180			
Degrees of Freedom	175			
X Coefficient(s)	0.249630	-523.975	0.006933	293.4309
Std Err of Coef.	0.124967	50.51729	0.000527	51.91040

Table G.7: Regression results of modelling the load friction with a second order model

G.5 CALCULATION OF TIME CONSTANT OF TEST RIG

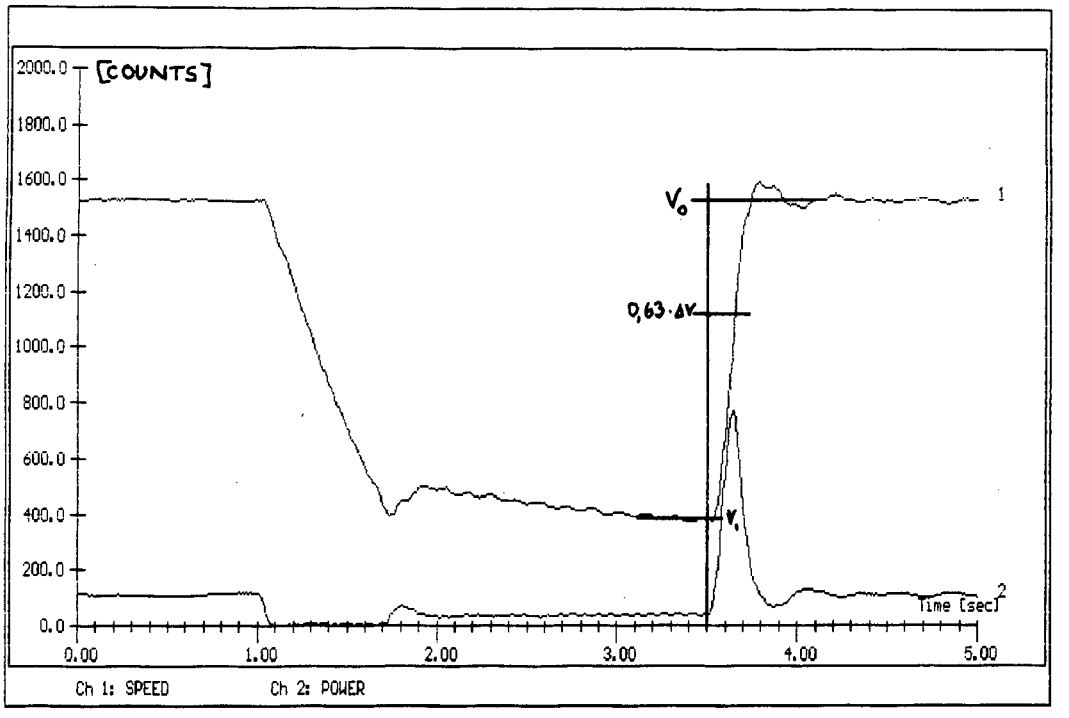


Figure G.8: Typical pulse response of laboratory test rig

Calculation of Time Constant:

$v(t_0) = v_0 = 1524$ counts

$v(t_1) = v_1 = 388$ counts

$\triangle v = v_0 - v_1 = 1136$ counts

$0.63 * \triangle v = 716$ counts

$v_x = v_0 + (0.63 * \triangle v) = 1103$ counts

$\Rightarrow t_x = 3.45$ sec

$\Rightarrow \tau_p = t_x - t_0 = 0.15$ sec $= 150$ msec

Hence a τ_p can be approximated as 100 msec. Hence τ_s is set to 10 msec and τ_f is set to 20msec.

APPENDIX H:
RESULTS FROM LABORATORY TESTS

H.1 SPEED AND POWER RESPONSES FROM TESTS

H.1.1 Various Speed Steps applied with VSD

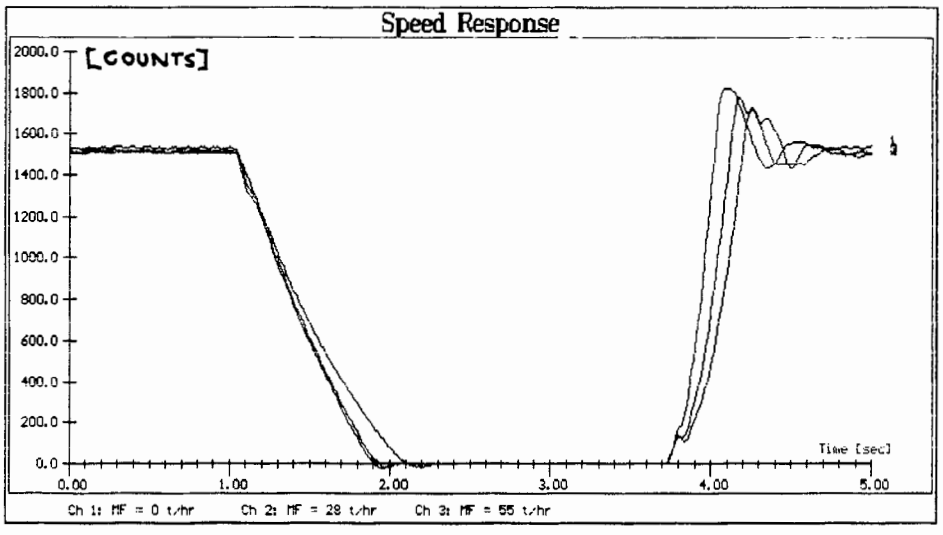


Figure H.1: Speed response of system; Initial setpoint = 100%; Step = 100%

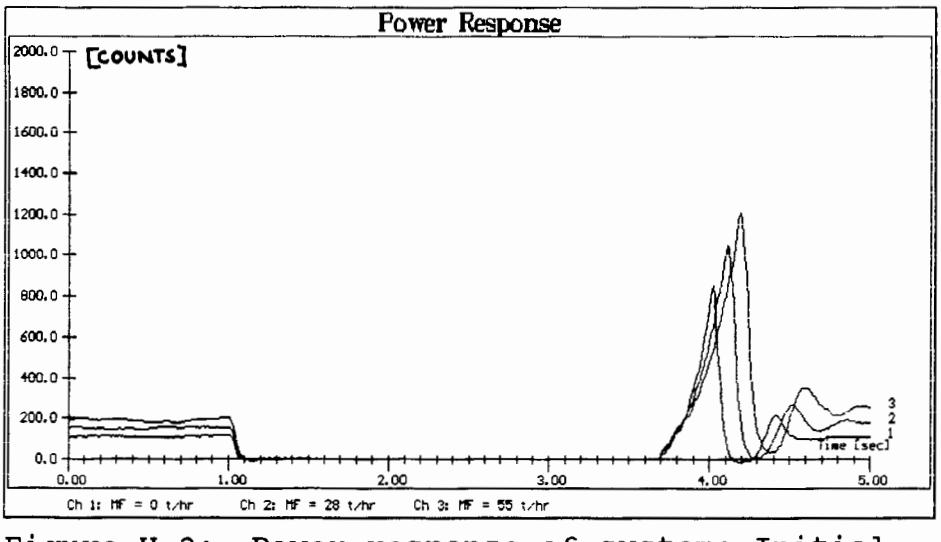


Figure H.2: Power response of system; Initial setpoint = 100%; Step = 100%

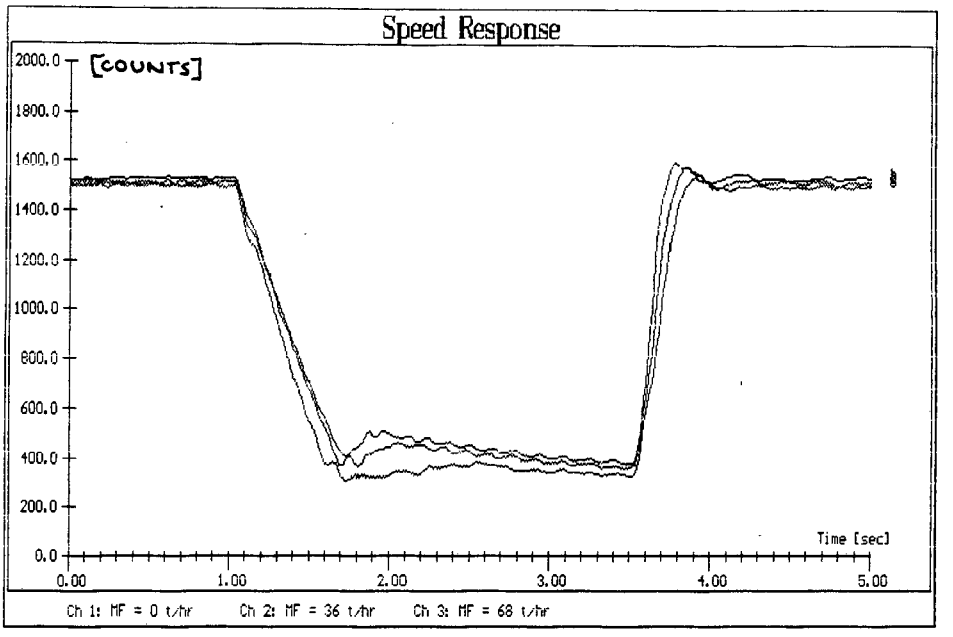


Figure H.3: Speed response of system; Initial setpoint = 100%; Step = 75%

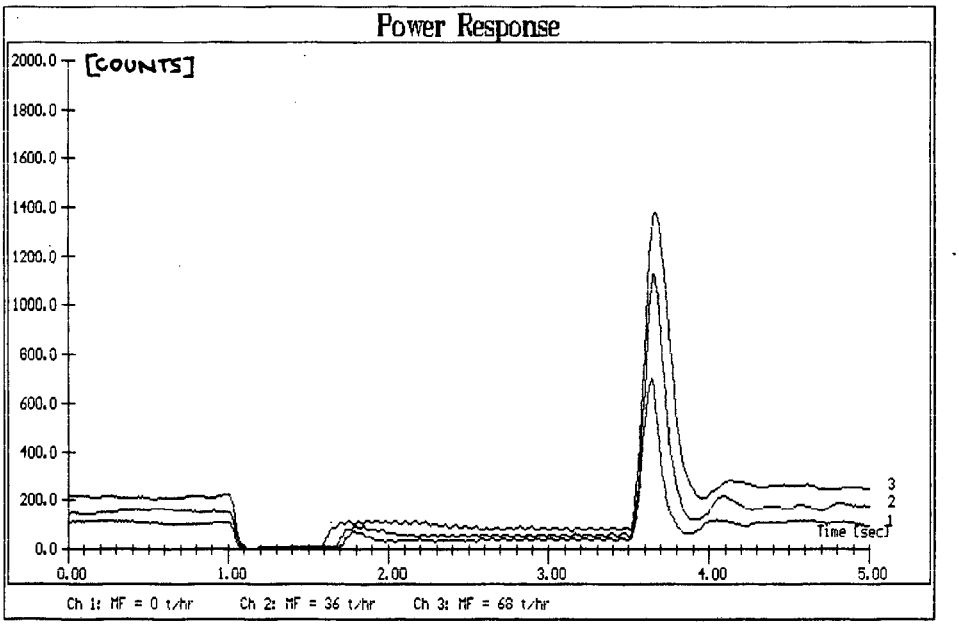


Figure H.4: Power response of system; Initial setpoint = 100%; Step = 75%

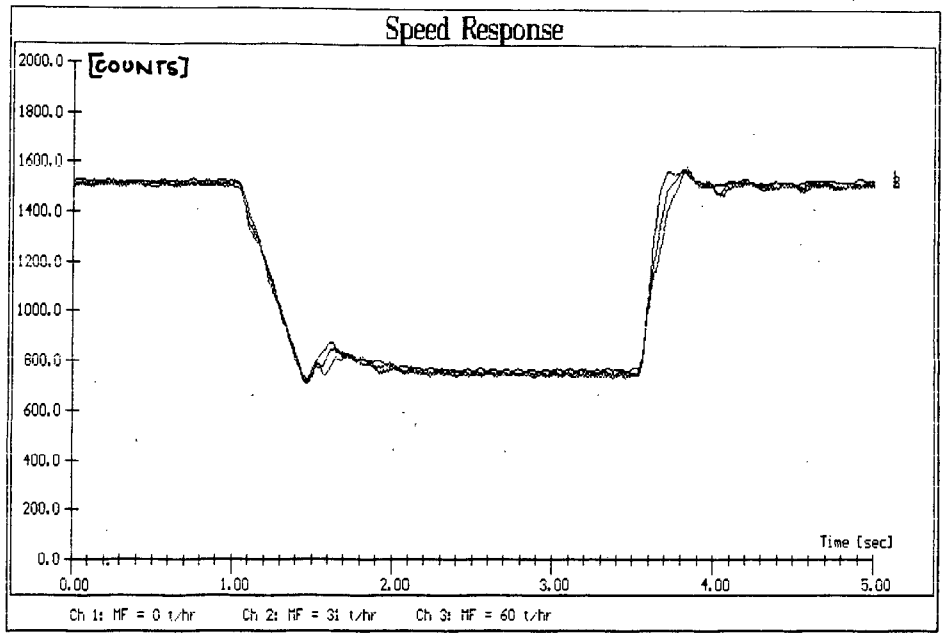


Figure H.5: Speed response of system; Initial setpoint = 100%; Step = 50%

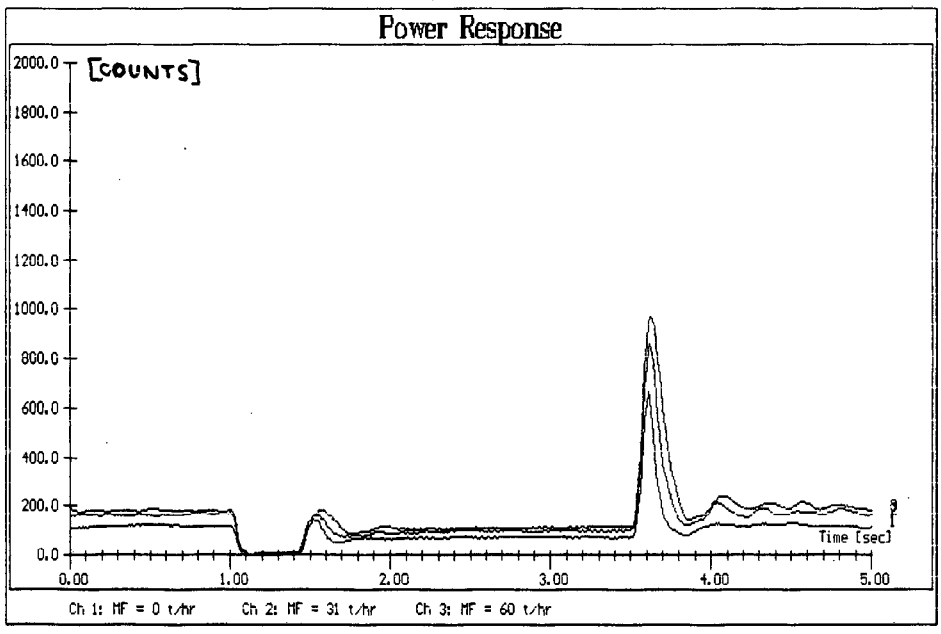


Figure H.6: Power response of system; Initial setpoint = 100%; Step = 50%

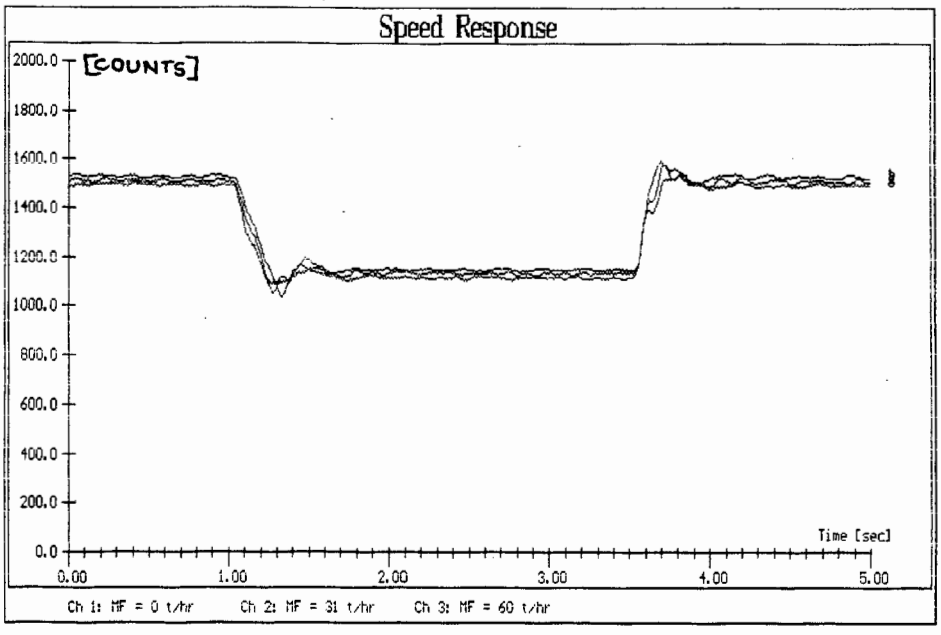


Figure H.7: Speed response of system; Initial setpoint = 100%; Step = 25%

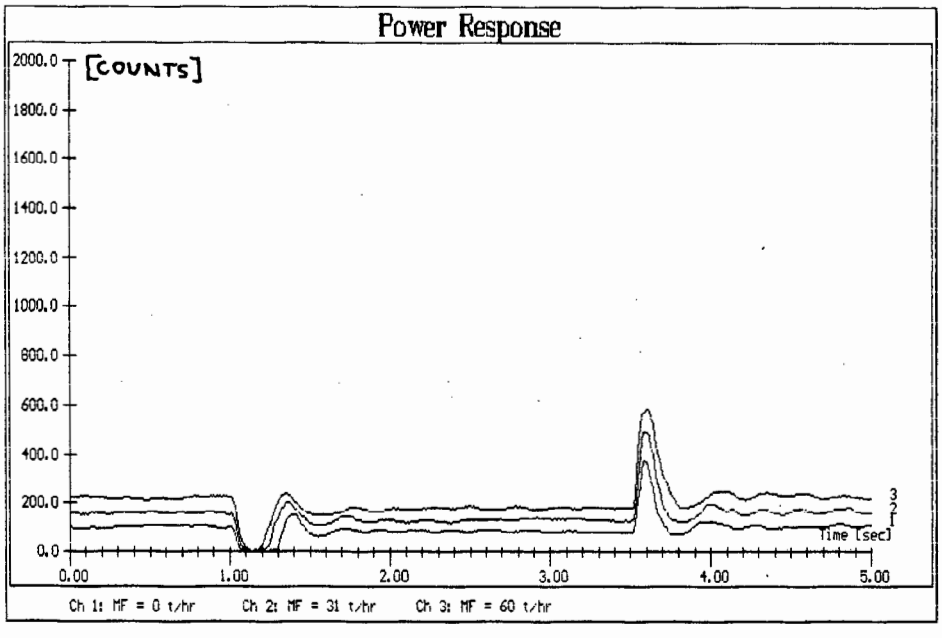


Figure H.8: Power response of system; Initial setpoint = 100%; Step = 25%

H.1.1 Various Initial Speed Setpoints using VSD to Perturb the system

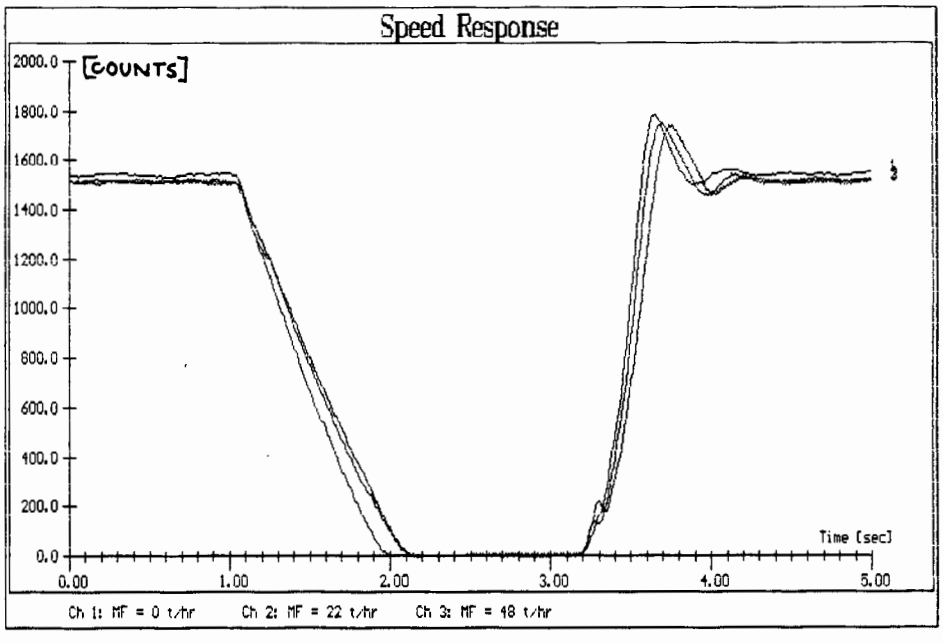


Figure H.9: Speed response of system; Initial setpoint = 100%; Step = 100%

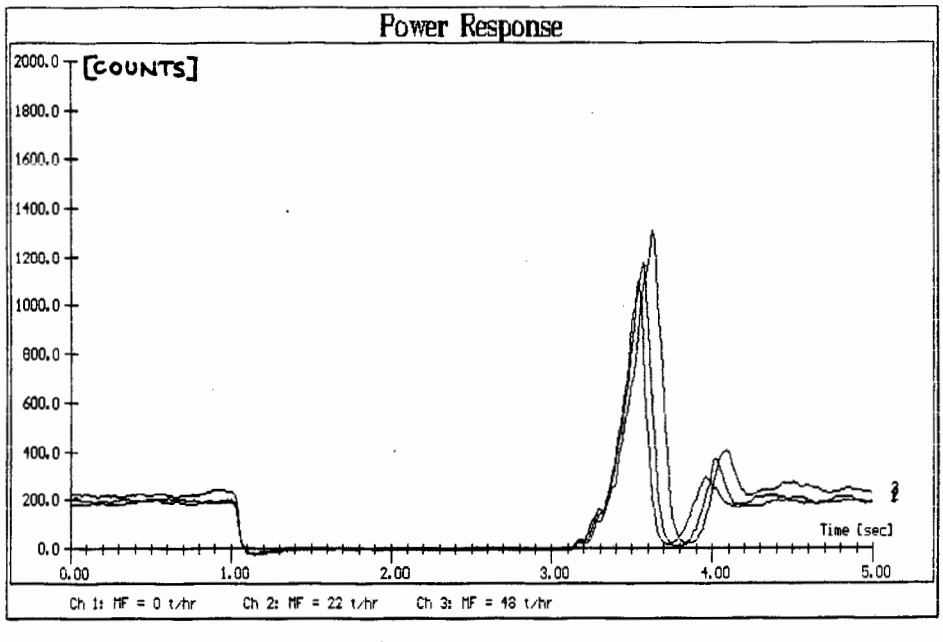


Figure H.10: Power response of system; Initial setpoint = 100%; Step = 100%

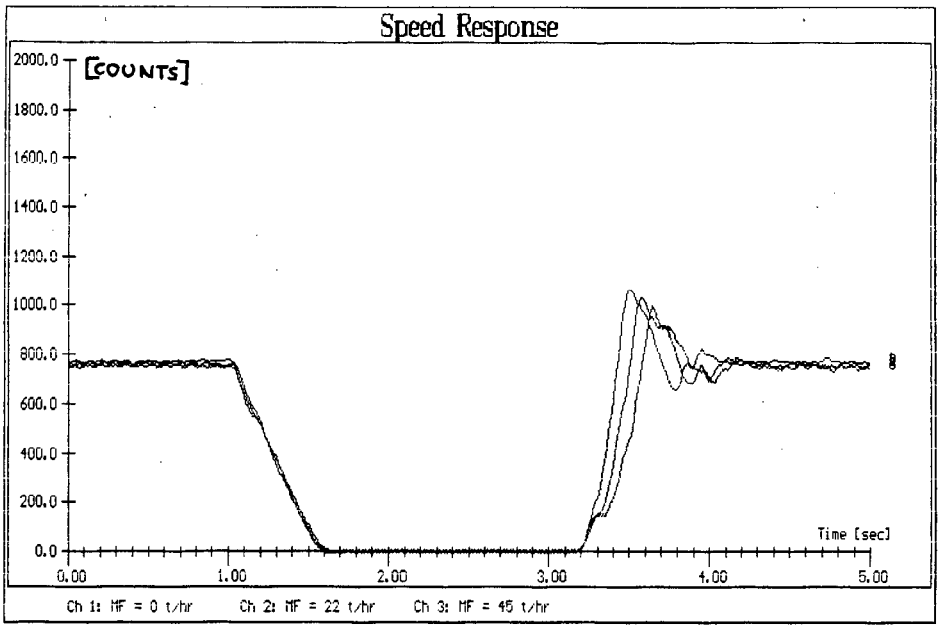


Figure H.11: Speed response of system; Initial setpoint = 50%; Step = 50%

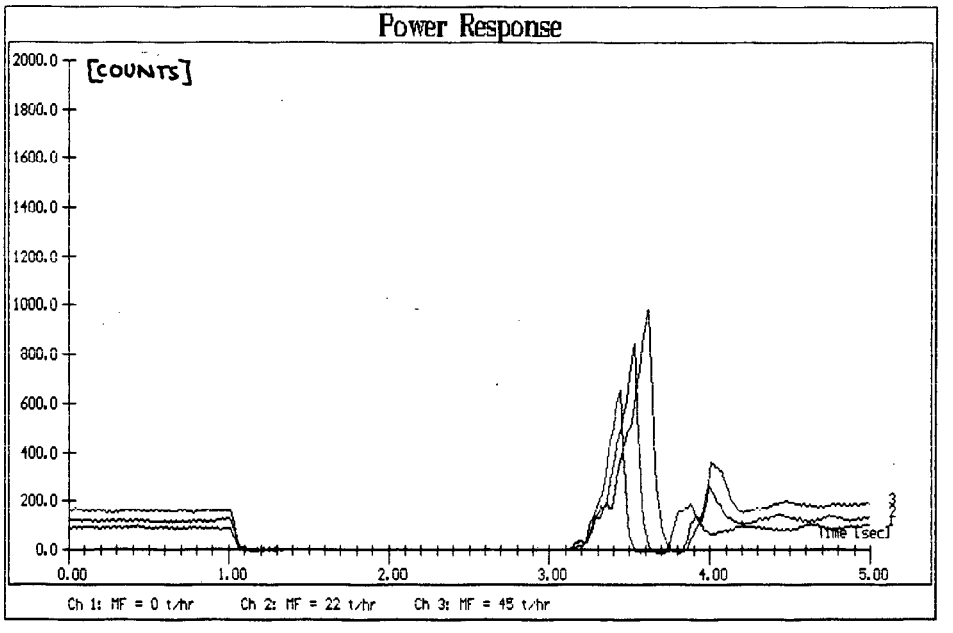


Figure H.12: Power response of system; Initial setpoint = 50%; Step = 50%

H.2 CORRELATION GRAPHS AND REGRESSION RESULTS

H.2.1 Constant Initial Speed; Different Steps

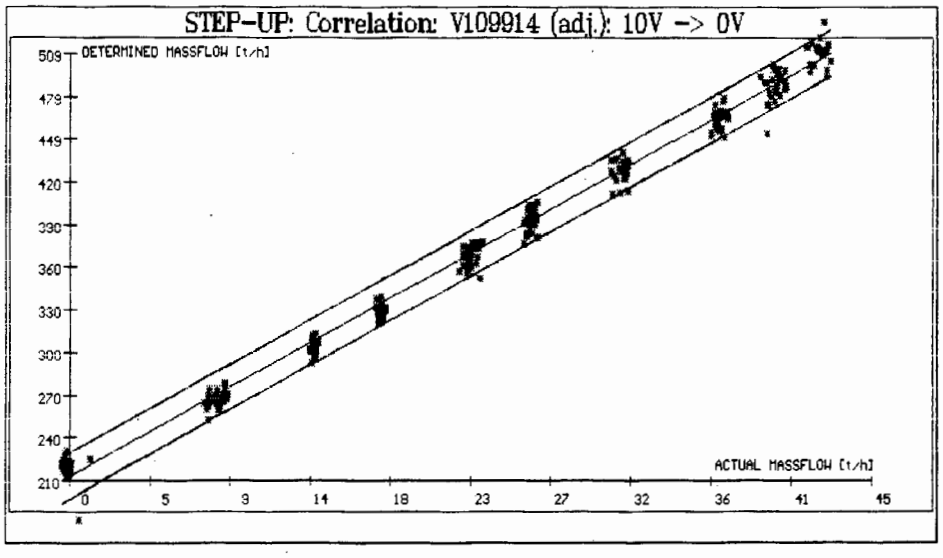


Figure H.13: Correlation graph: Initial speed setpoint = 100%; Step = 100%

Results:

	As from : Step Dn Response	Step Up Response
Y constant:	1.0404	6.9305
Slope:	97.1450	213.4396
Full Scale Deflection	42.68 t/hr	42.68 t/hr
Error at Mean	7.01 t/hr	2.28 t/hr
% error of	16.42 %	5.34 %
Correlation coefficient	0.9655	0.9962

v - View Correlation Graph
c - Change Scales of Correlation Graph

q - Quit

Option: [q]

Table H.1: Regression results: Initial speed setpoint = 100% ; Step = 100%

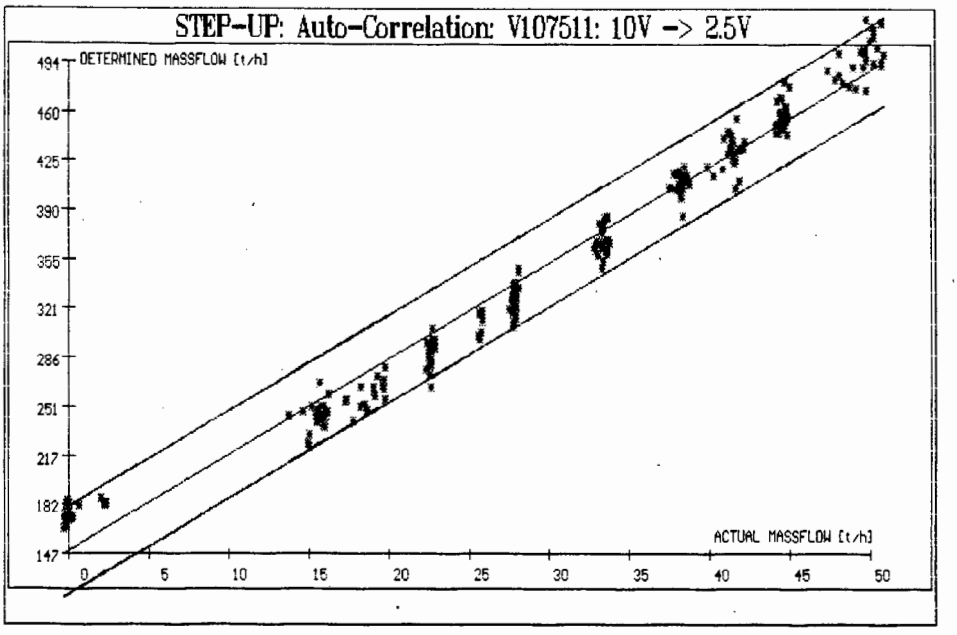


Figure H.14: Correlation graph: Initial speed setpoint = 100% ; Step = 75%

Results:

As from :

	Step Dn Response	Step Up Response
Y constant:	0.1989	6.7719
Slope:	91.5103	149.6775
Full Scale Deflection	50.89 t/hr	50.89 t/hr
Error at Mean	67.90 t/hr	4.54 t/hr
% error of	133.42 %	8.92 %
Correlation coefficient	0.4069	0.9879

v - View Correlation Graph
c - Change Scales of Correlation Graph

q - Quit

Option: [q]

Table H.2: Regression results: Initial speed setpoint = 100% ; Step = 75%

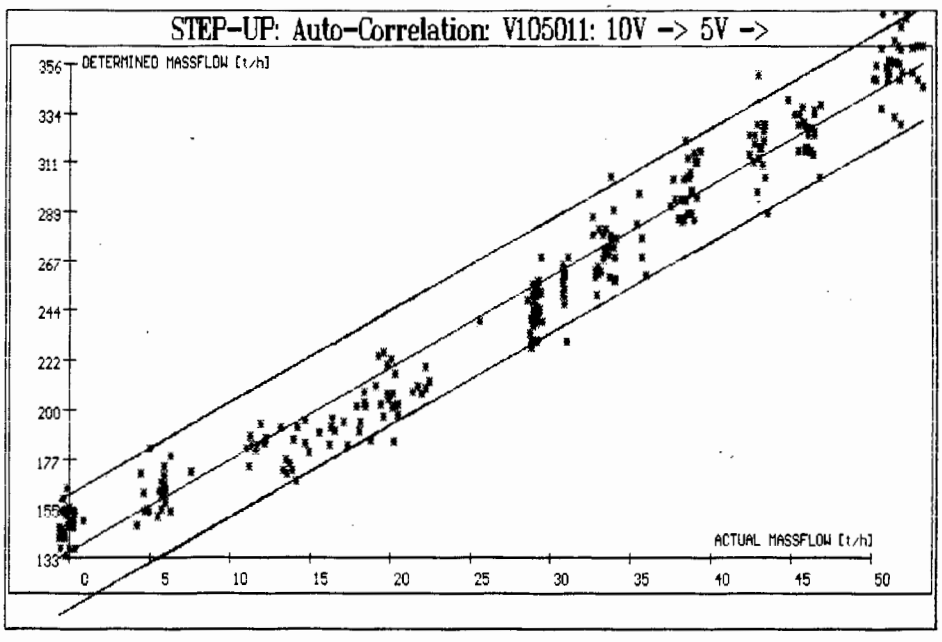


Figure H.15: Correlation graph: Initial speed setpoint = 100% ; Step = 50%

Results:

	As from : Step Dn Response	Step Up Response
Y constant:	0.4913	4.1473
Slope:	78.1247	135.3784
Full Scale Deflection	53.24 t/hr	53.24 t/hr
Error at Mean	26.14 t/hr	6.24 t/hr
% error of	49.09 %	11.71 %
Correlation coefficient	0.7802	0.9820

v - View Correlation Graph
c - Change Scales of Correlation Graph

q - Quit

Option: [q]

Table H.3: Regression results: Initial speed setpoint = 100% ; Step = 50%

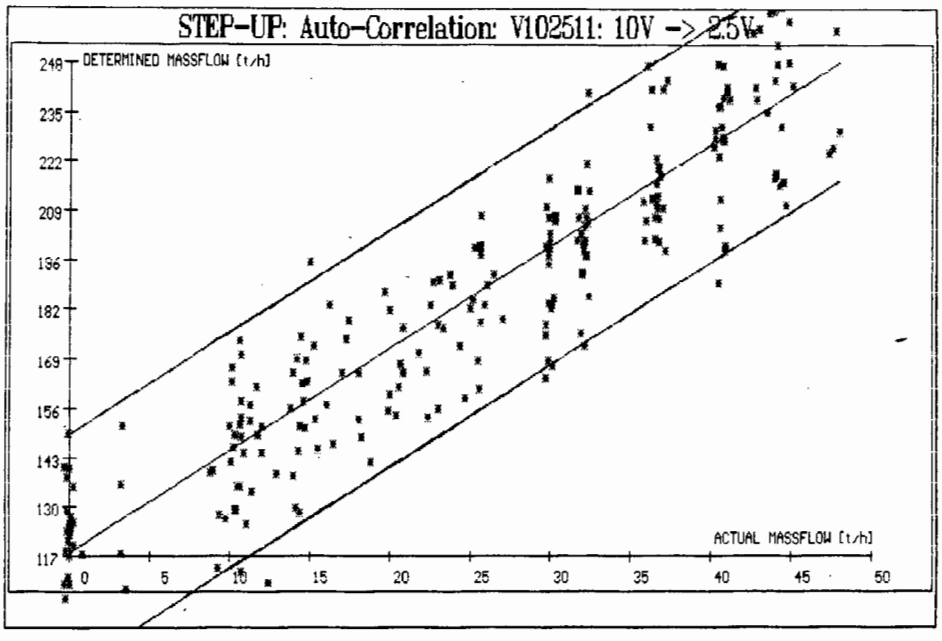


Figure H.16: Correlation graph: Initial speed setpoint = 100% ; Step = 25%

Results:

	As from : Step Dn Response	Step Up Response
Y constant:	0.7294	2.7274
Slope:	49.4534	117.3974
Full Scale Deflection	48.02 t/hr	48.02 t/hr
Error at Mean	25.54 t/hr	11.58 t/hr
% error of	53.18 %	24.11 %
Correlation coefficient	0.7405	0.9241

v - View Correlation Graph
c - Change Scales of Correlation Graph

q - Quit

Option: [q]

Table H.4: Regression results: Initial speed setpoint = 100% ; Step = 25%

H.2.2 Differing Initial Speed; Total Step

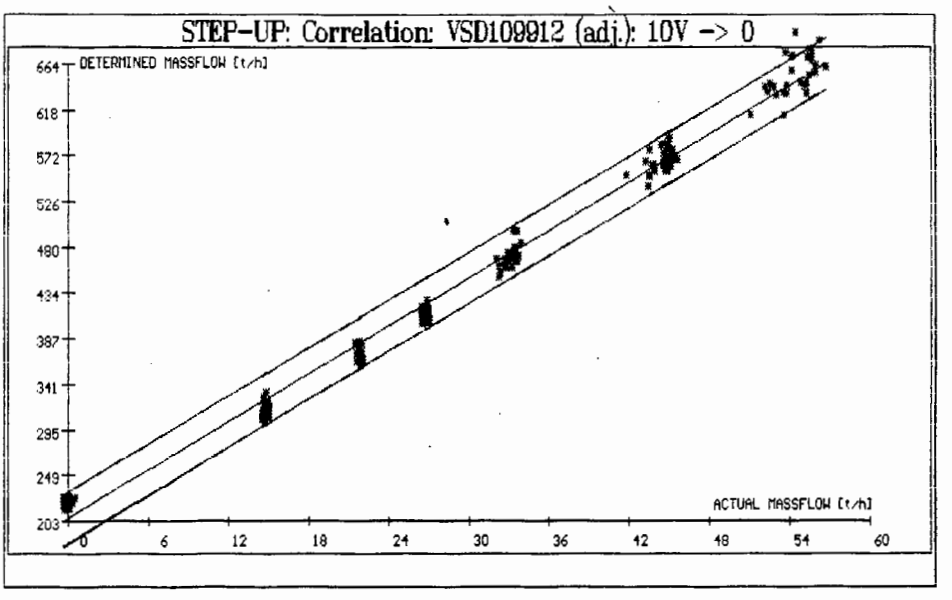


Figure H.17: Correlation graph: Initial speed setpoint = 100%; Step = 100%

Results:

As from :

	Step Dn Response	Step Up Response
Y constant:	0.9858	8.0725
Slope:	109.9347	206.2537
Full Scale Deflection	56.72 t/hr	56.72 t/hr
Error at Mean	8.13 t/hr	3.20 t/hr
% error of	14.34 %	5.64 %
Correlation coefficient	0.9721	0.9955

v - View Correlation Graph
c - Change Scales of Correlation Graph

q - Quit

Option: [q]

Table H.5: Regression results: Initial speed setpoint = 100%; Step = 100%

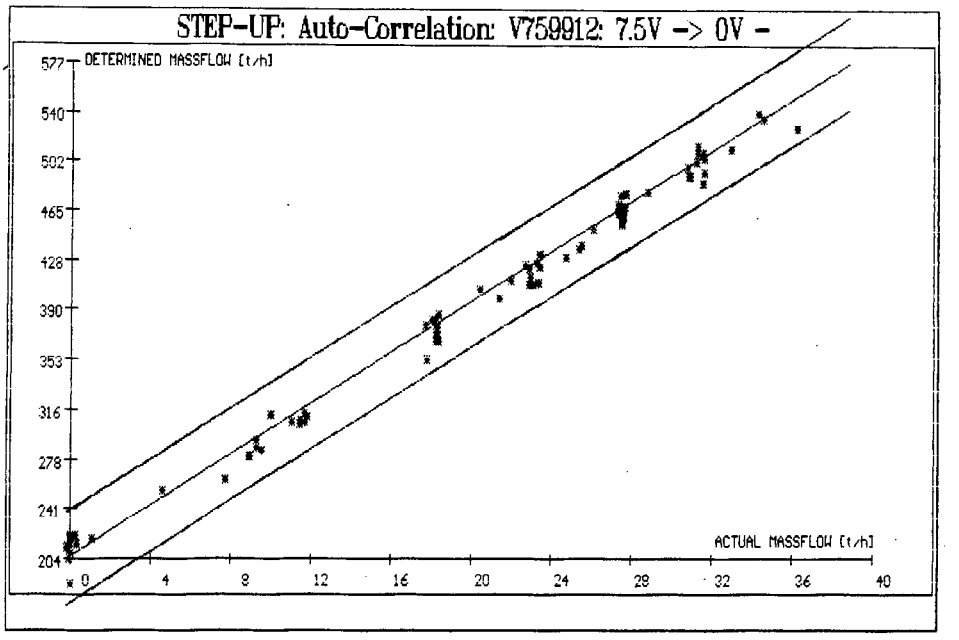


Figure H.18: Correlation graph: Initial speed setpoint = 75% ; Step = 75%

Results:

As from :

	Step Dn Response	Step Up Response
Y constant:	1.0830	9.5797
Slope:	75.9730	205.2282
Full Scale Deflection	38.80 t/hr	38.80 t/hr
Error at Mean	7.99 t/hr	3.57 t/hr
% error of	20.60 %	9.21 %
Correlation coefficient	0.9443	0.9880

v - View Correlation Graph
c - Change Scales of Correlation Graph

q - Quit

Option: [q]

Table H.6: Regression results: Initial speed setpoint = 75%; Step = 75%

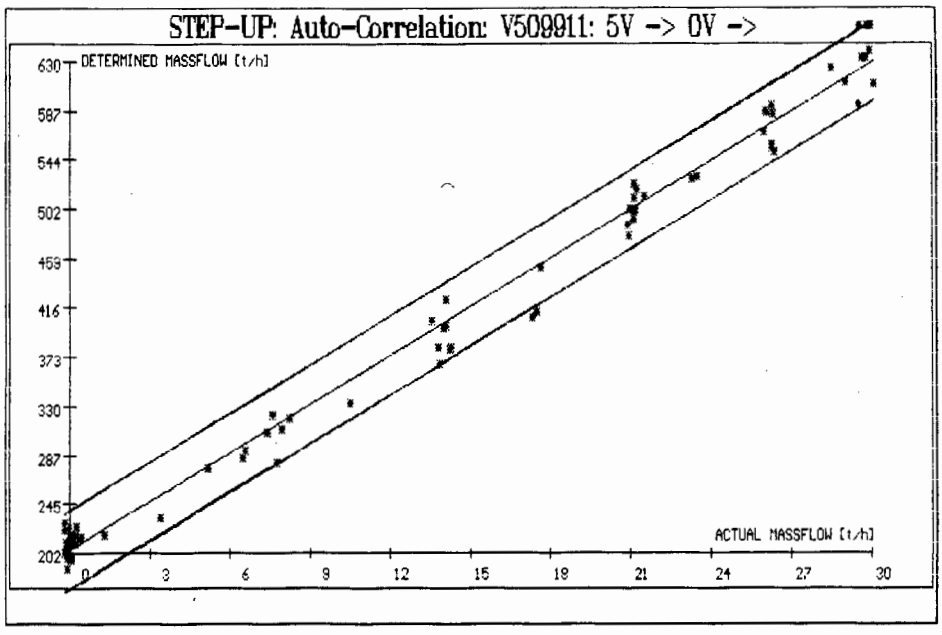


Figure H.19: Correlation graph: Initial speed setpoint = 50% ; Step = 50%

Results:

As from :

	Step Dn Response	Step Up Response
Y constant:	1.1733	14.1461
Slope:	40.7756	204.5264
Full Scale Deflection	30.08 t/hr	30.08 t/hr
Error at Mean	4.83 t/hr	2.42 t/hr
% error of	16.04 %	8.03 %
Correlation coefficient	0.9784	0.9944

v - View Correlation Graph
c - Change Scales of Correlation Graph

q - Quit

Option: [q]

Table H.7: Regression results: Initial speed setpoint = 50% ; Step = 50%

H.2.3 SSR: Stopping the HCB

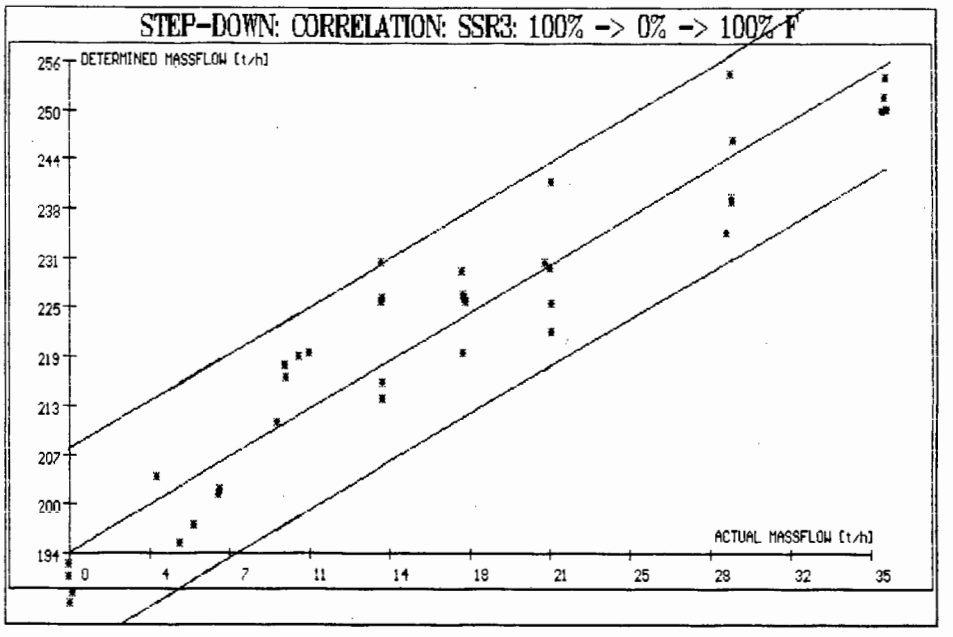


Figure H.20: Correlation graph for step -down response: Perturbation with SSR

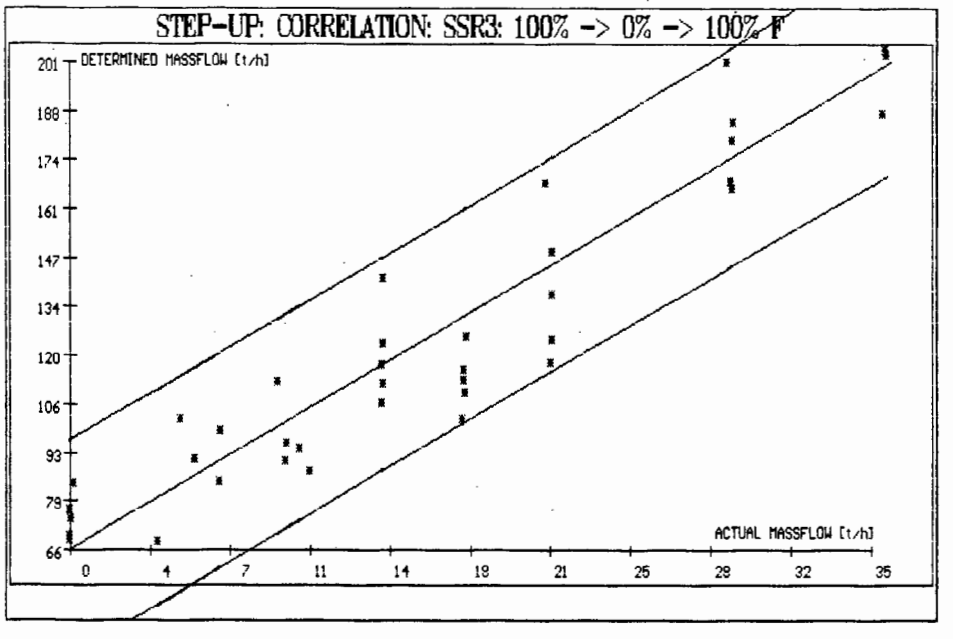


Figure H.21: Correlation graph for step -up response: Perturbation with SSR

Results:

	As from : Step Dn Response	Step Up Response
Y constant:	1.7276	3.7771
Slope:	194.2205	65.9635
Full Scale Deflection	35.86 t/hr	35.86 t/hr
Error at Mean	7.43 t/hr	7.92 t/hr
% error of	20.72 %	22.08 %
Correlation coefficient	0.9535	0.9477

v - View Correlation Graph
c - Change Scales of Correlation Graph

q - Quit

Option: [q]

Table H.8: Regression results: Perturbation with SSR

H.2.4 SSR: Slowing HCB down

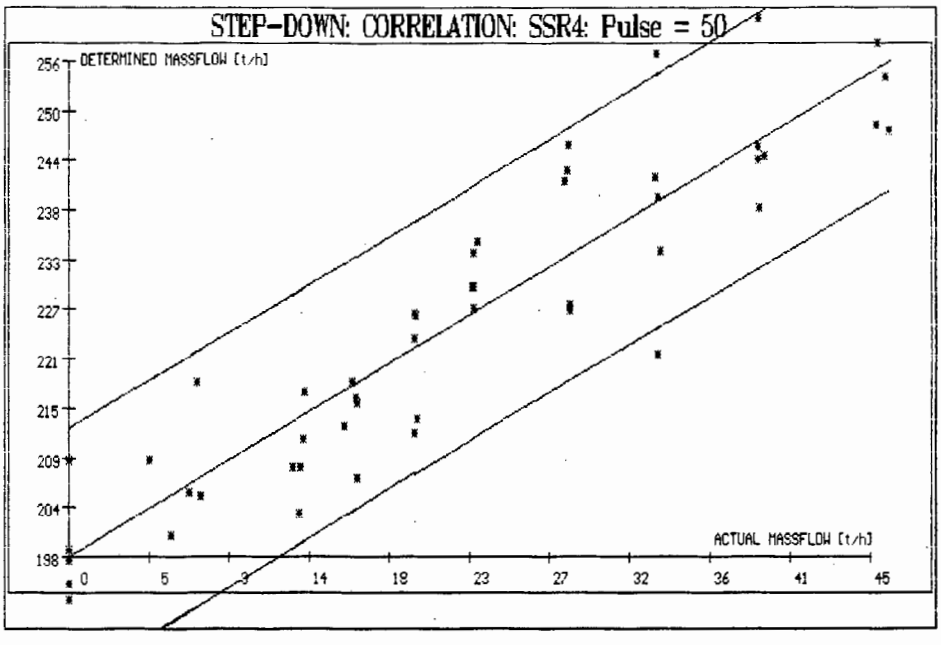


Figure H.22: Correlation graph for step-down response: Perturbation with SSR; System only slowed down

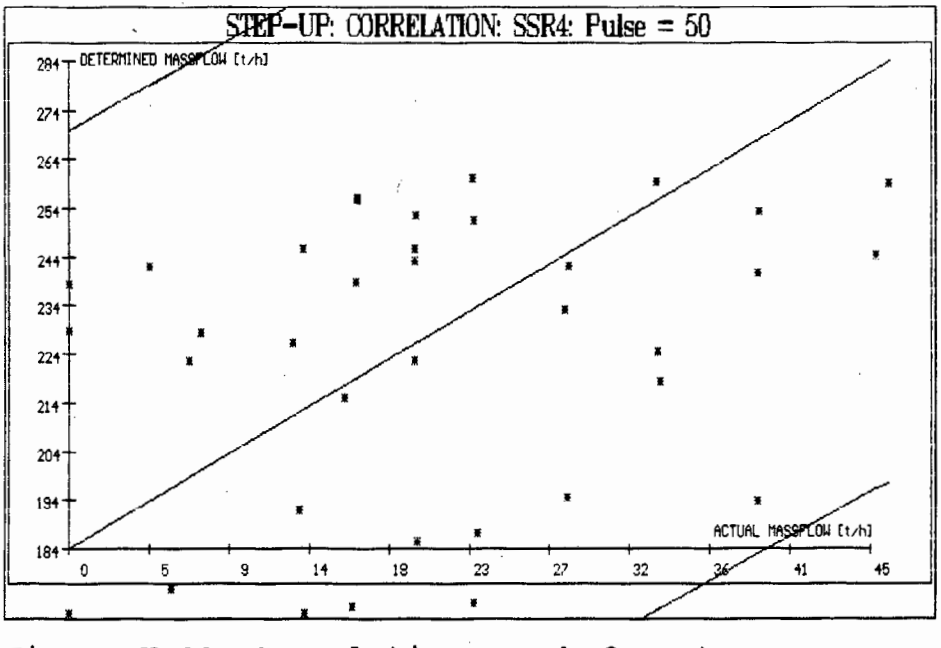
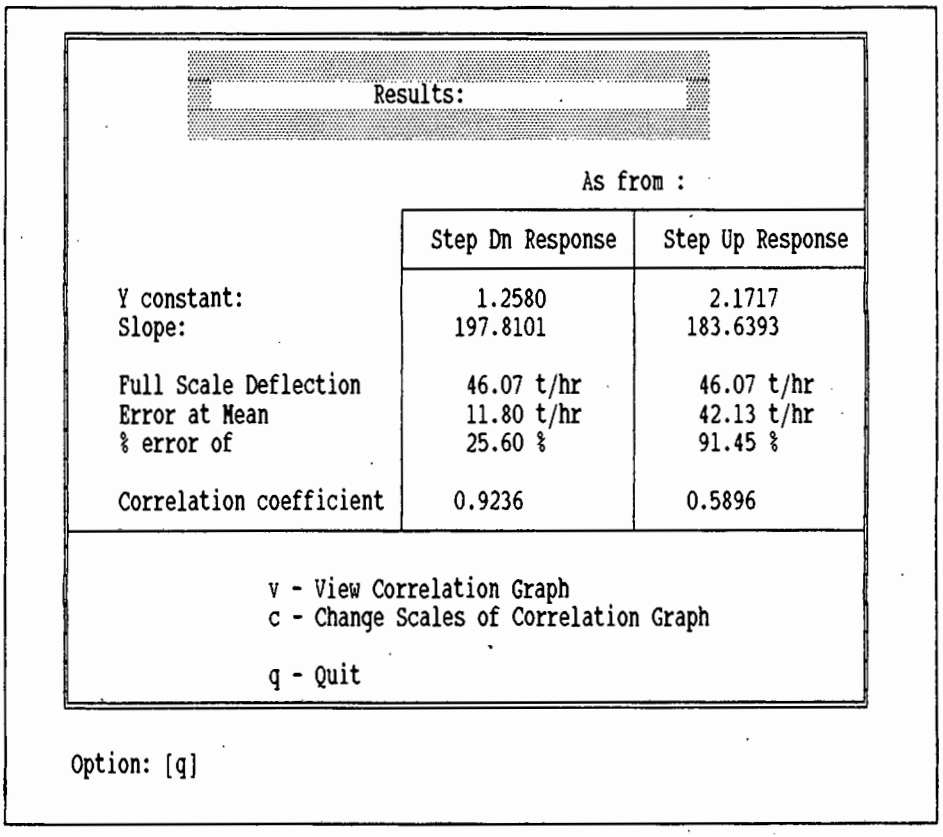


Figure H.23: Correlation graph for step-up response: Perturbation with SSR; System only slowed down



Results:

As from :

	Step Dn Response	Step Up Response
Y constant:	1.2580	2.1717
Slope:	197.8101	183.6393
Full Scale Deflection	46.07 t/hr	46.07 t/hr
Error at Mean	11.80 t/hr	42.13 t/hr
% error of	25.60 %	91.45 %
Correlation coefficient	0.9236	0.5896

v - View Correlation Graph
c - Change Scales of Correlation Graph

q - Quit

Option: [q]

Table H.9: Regression results: Perturbation with SSR; System only slowed down

H.2.5 Effect of Filtering the Samples

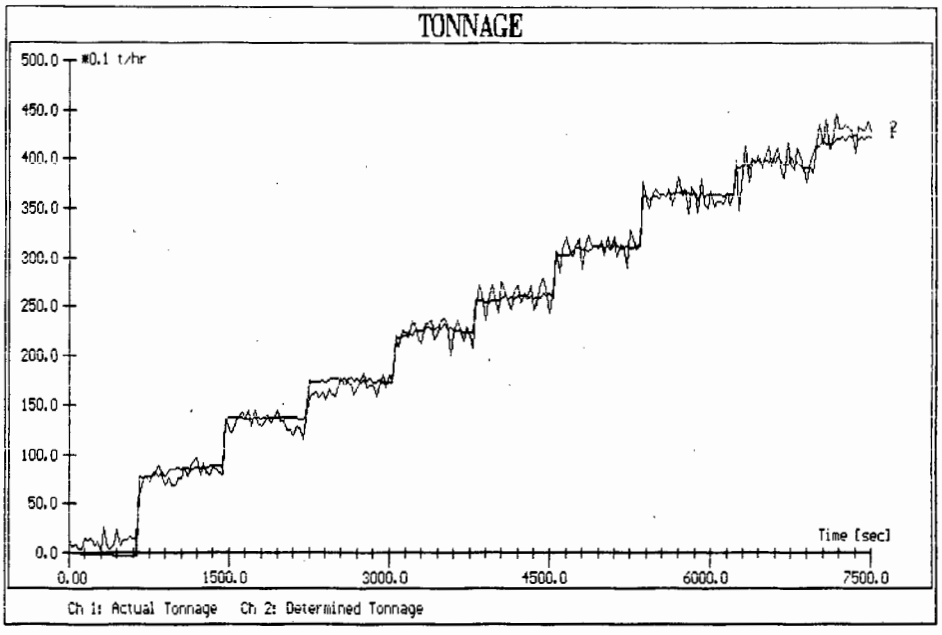


Figure H.24: Time plot of estimated and reference massflow for τ_f = 6s

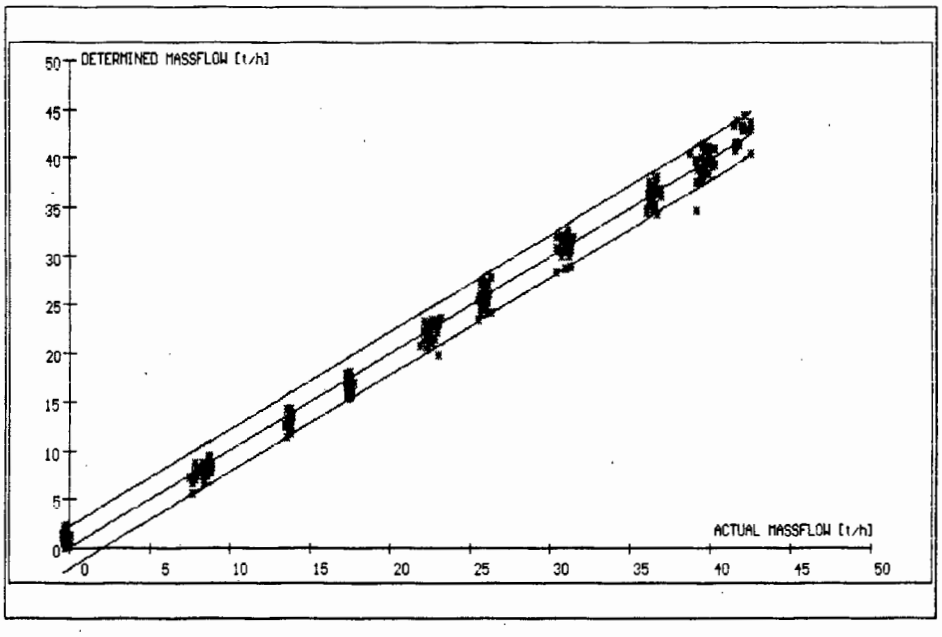


Figure H.25: Correlation graph of estimated versus reference massflow for τ_f = 6s

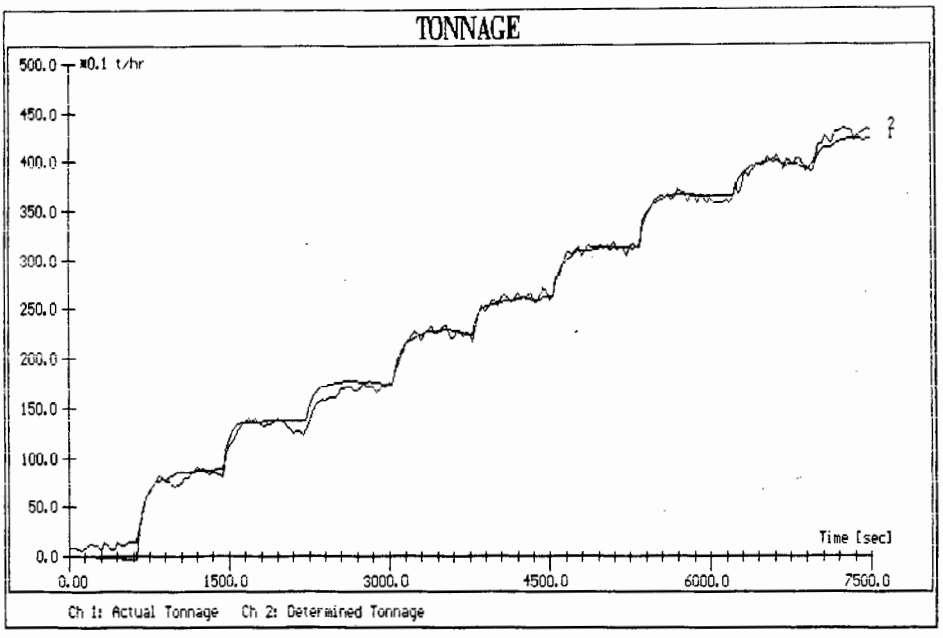


Figure H.26: Time plot of estimated and reference massflow for τ_f = 60s

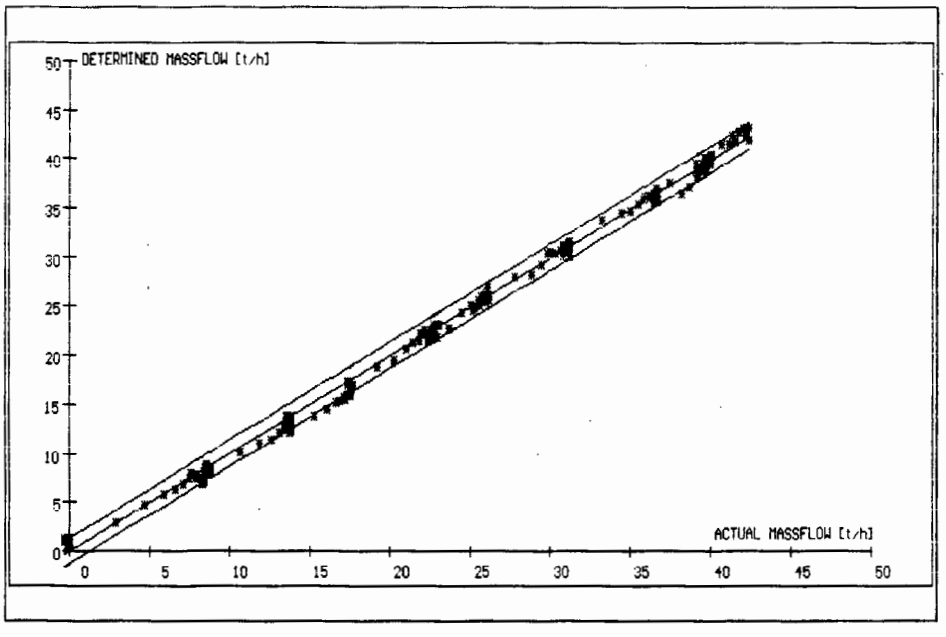


Figure H.27: Correlation graph of estimated versus reference massflow for τ_f = 60s

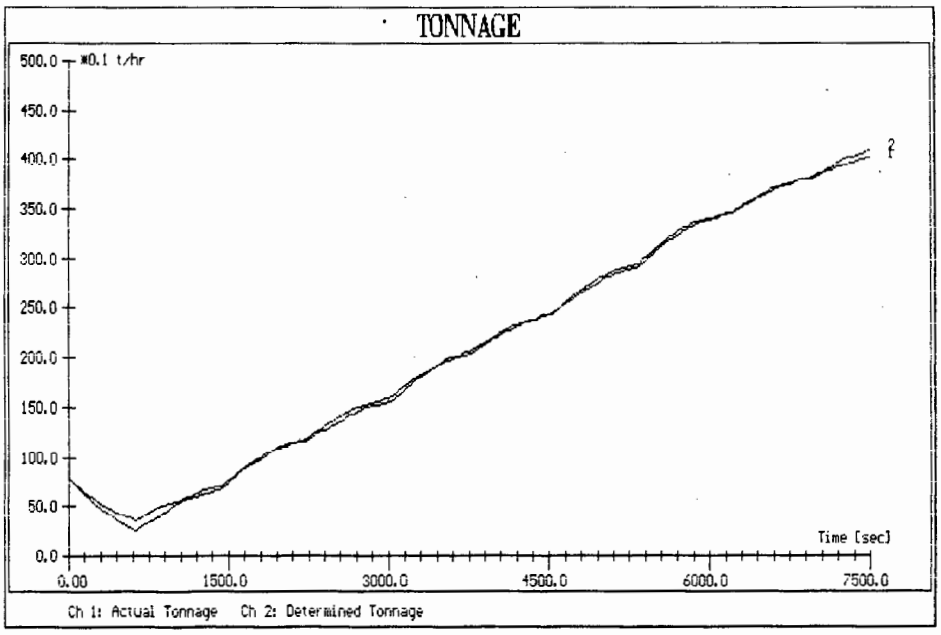


Figure H.28: Time plot of estimated and reference massflow for τ_f = 600

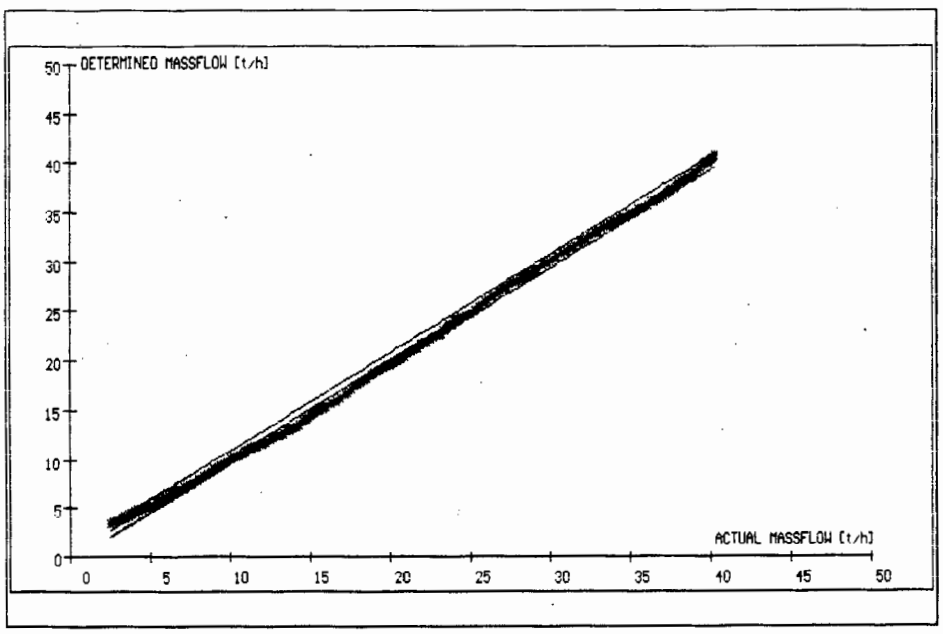


Figure H.29: Correlation graph of estimated versus reference massflow for τ_f = 600s

APPENDIX I:
CIRCUITS USED FOR
MINE TEST RIG

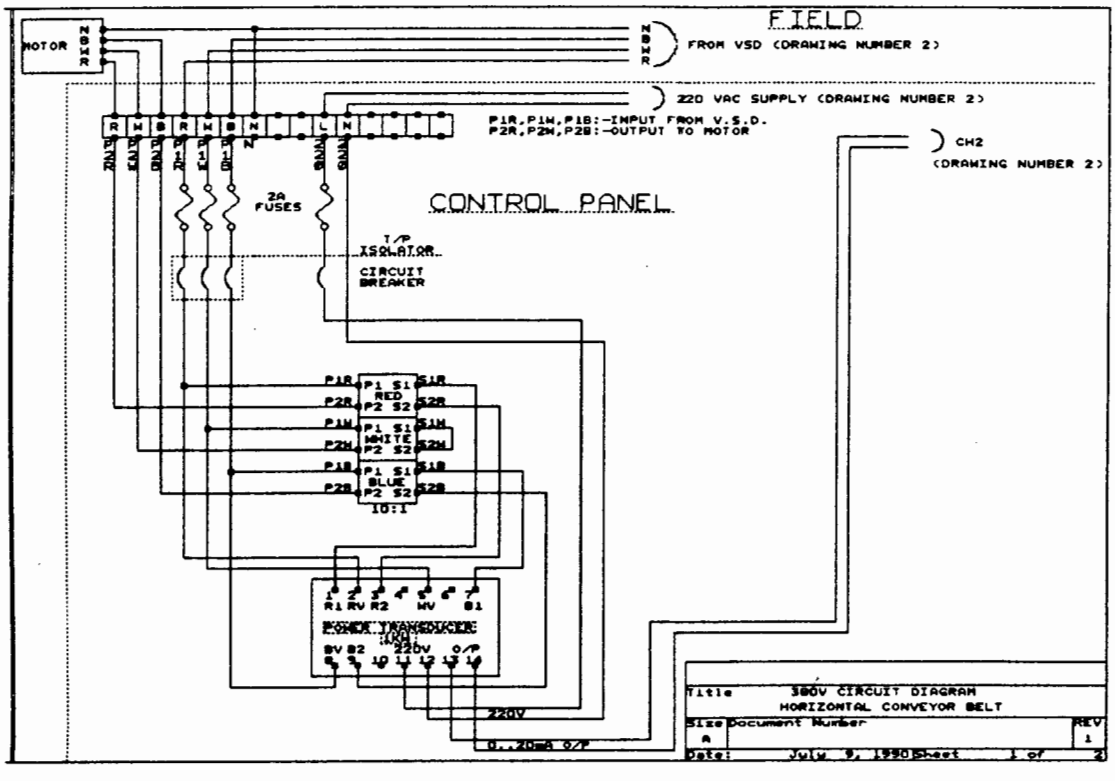


Figure I.1: Detailed circuit diagram of 380V network

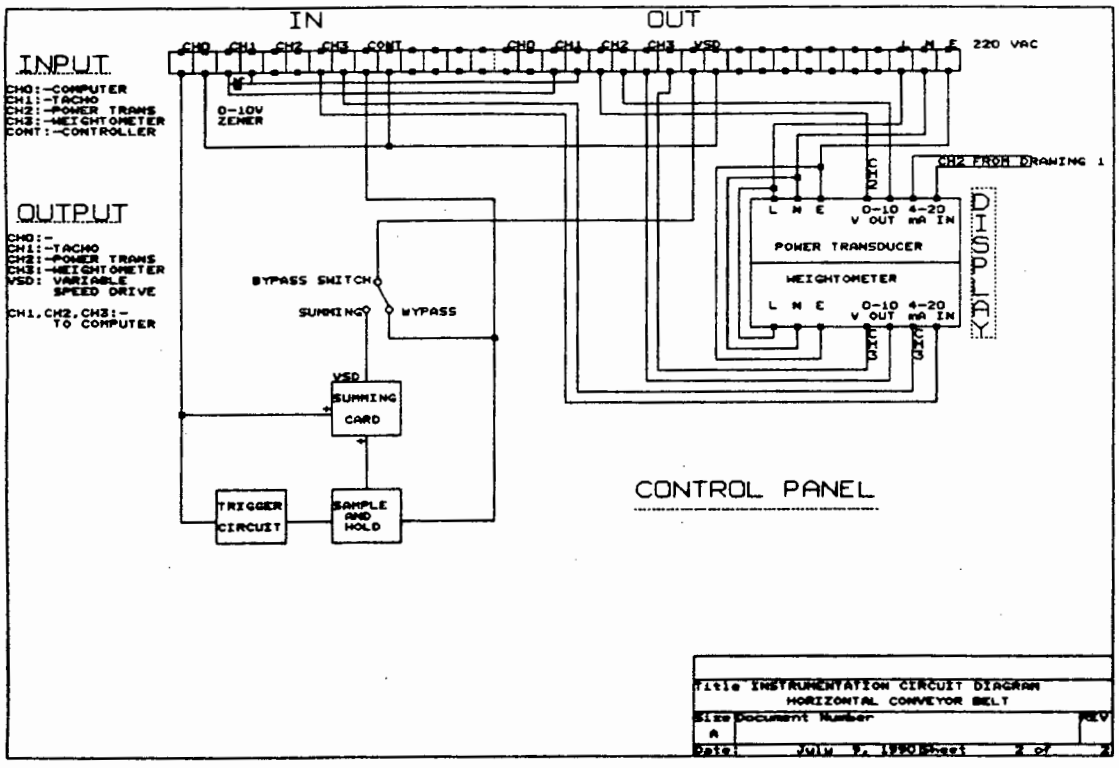


Figure I.2: Detailed circuit diagram of instrumentation circuit

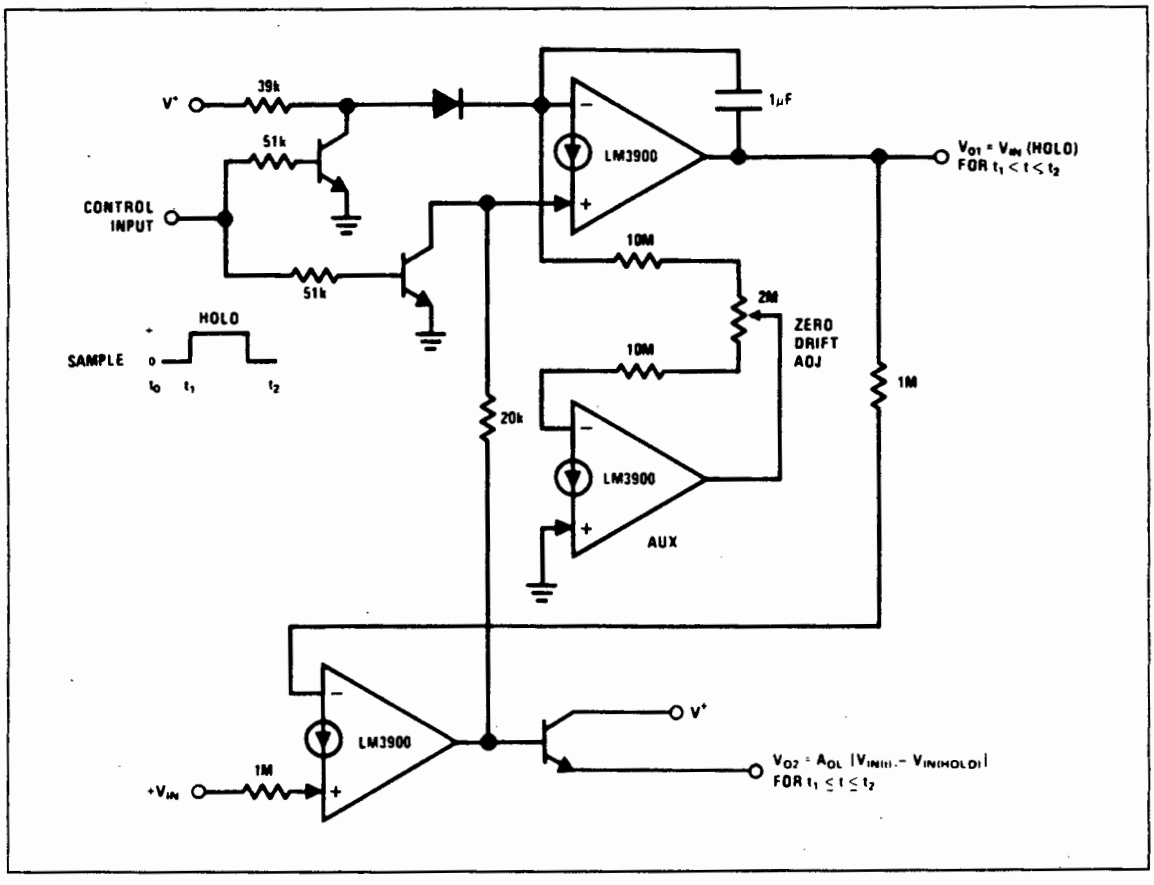


Figure I.3: Circuit diagram of Sample and Hold circuit (Linear 1 Databook [10])

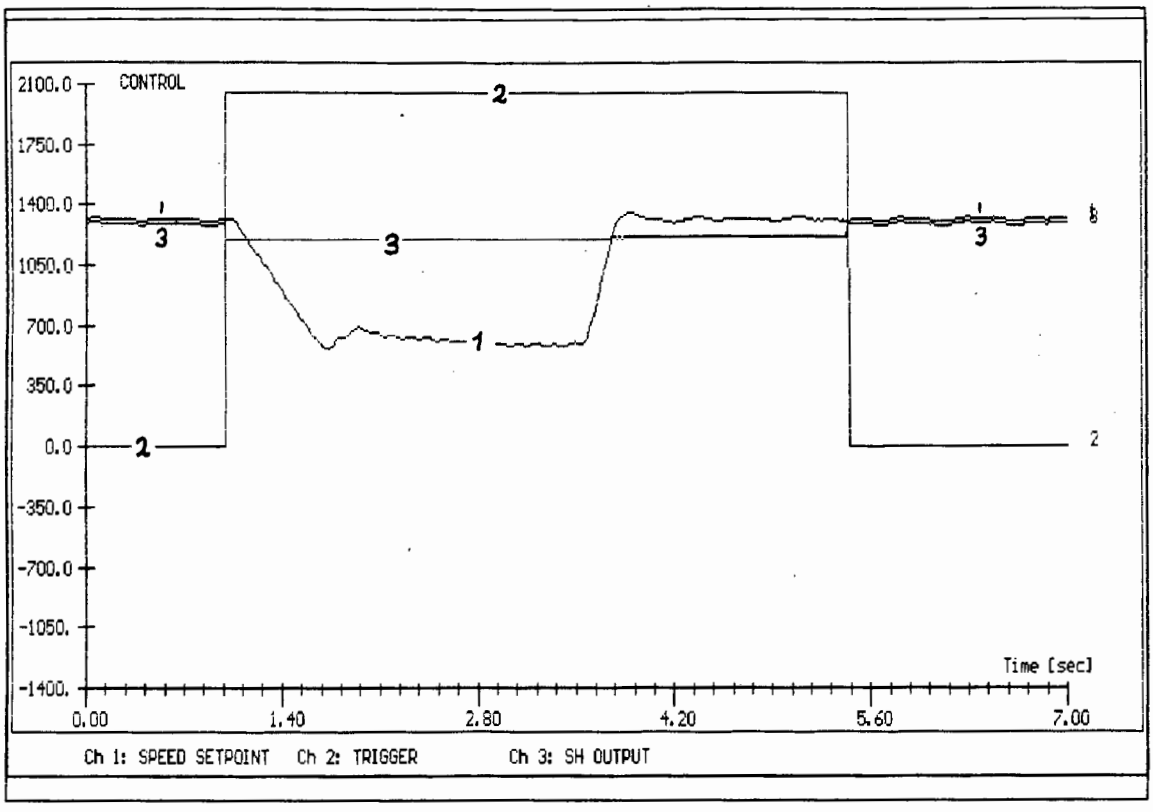


Figure I.4: Timing diagram for Sample and Hold circuit

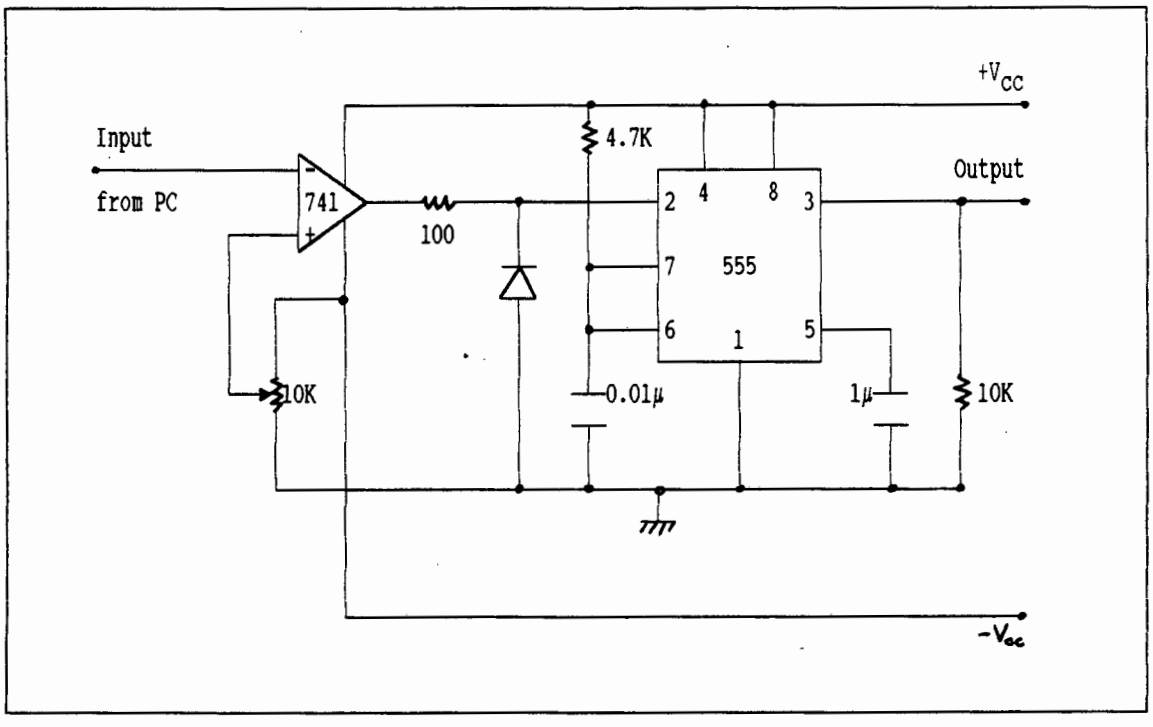


Figure I.5: Circuit diagram for trigger signal

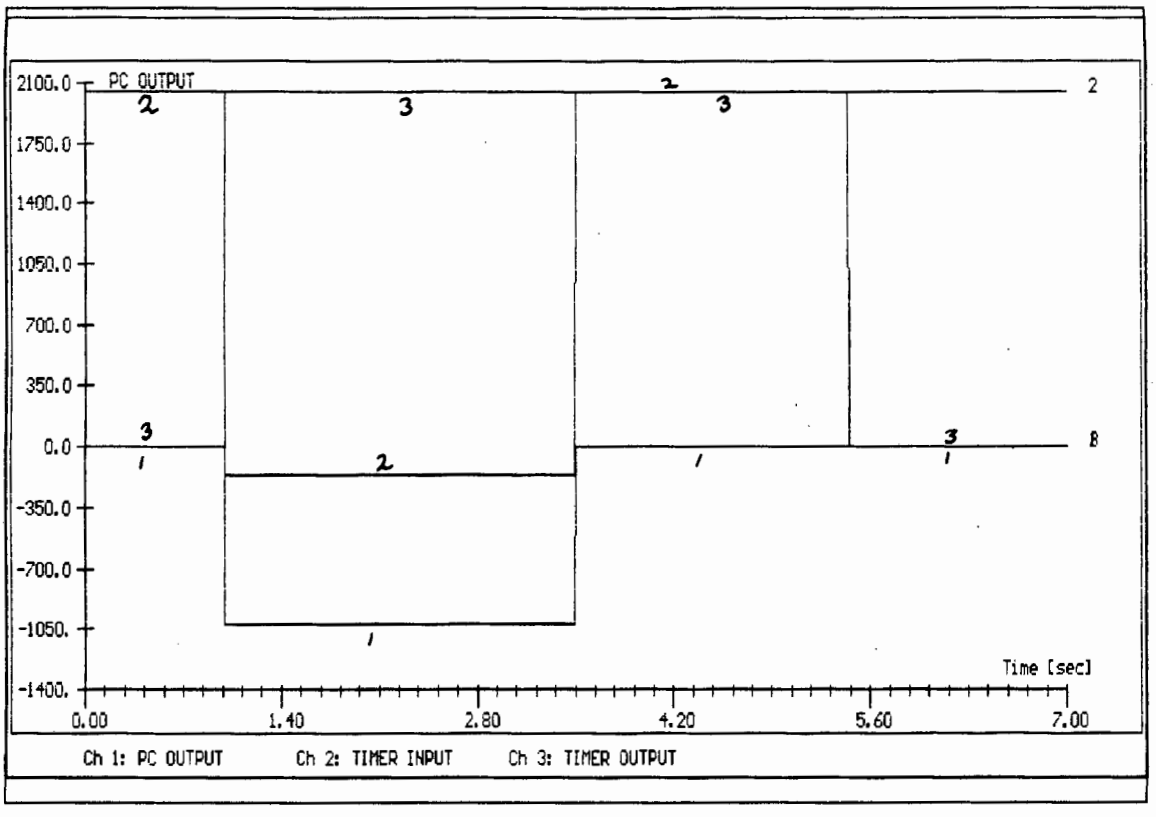


Figure I.6: Timing diagram for trigger signal circuit

APPENDIX J:
CALIBRATION RESULTS FOR MINE TEST RIG

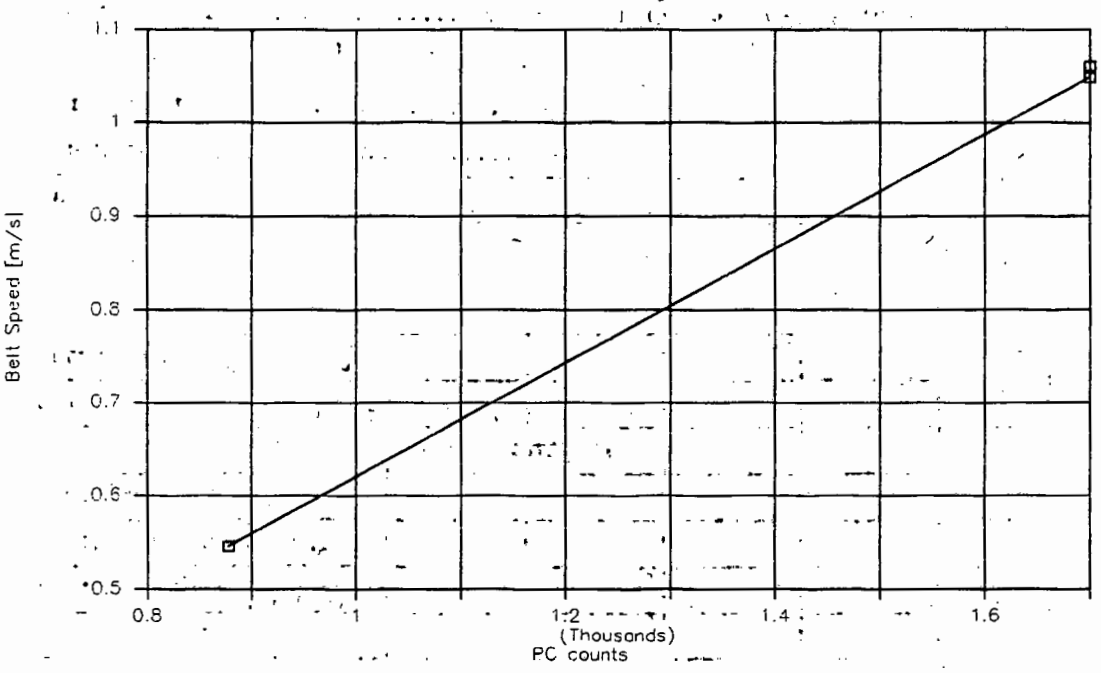


Figure J.1: Correlation graph for speed of conveyor belt

Regression Output:	
Constant	0.004391
Std Err of Y Est	0.006
R Squared	0.999721
No. of Observations	4
Degrees of Freedom	2
X Coefficient(s)	0.00061800
Std Err of Coef.	0.000007299

Table J.1: Regression results for speed calibration

DETERMINING THE TIME CONSTANT OF THE PLANT

Below a typical pulse response of the system is shown:

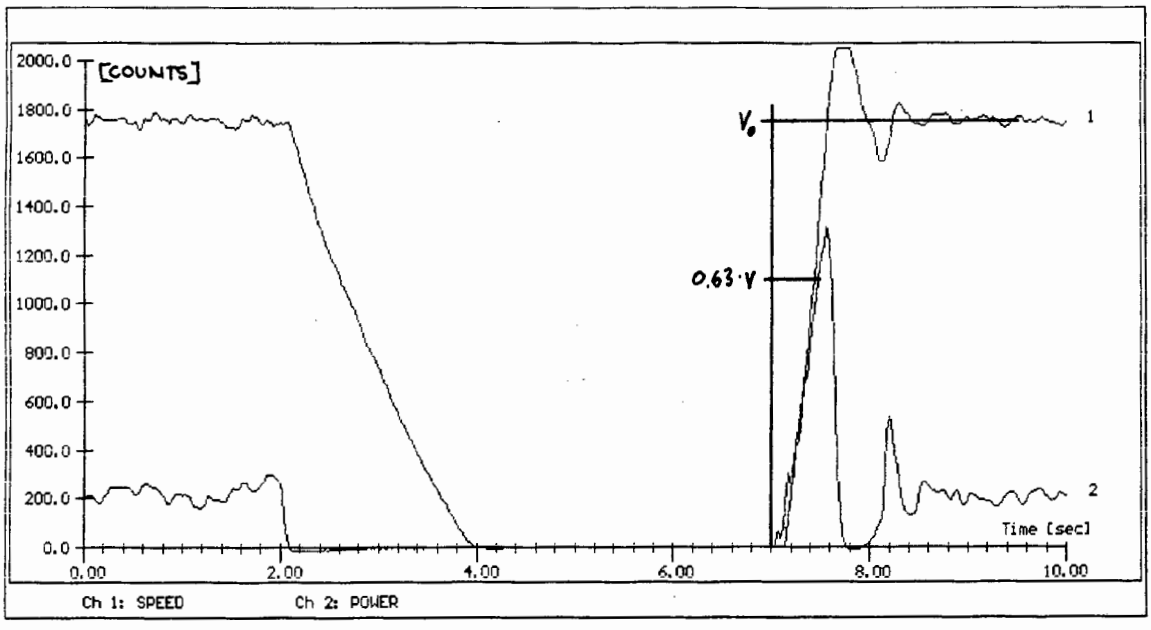


Figure J.2: Typical pulse response of test plant on a mine

The time constant is equal to that time, that the system needs to reach 63% of the new setpoint after a step was applied.

Thus

$v_0 = v(t_0) = 1755$ counts

$v_1 = v(t_1) = 0$ counts

$\triangle v = v_1 - v_0 = 1755$ counts

$0.63 * \triangle v = 1106$ counts

$\Rightarrow \quad v_x = v_0 + (0.63 * \triangle v) = 1106$ counts

$t_x = 7.42$ sec

Thus

$\tau_p = t_x - t_0 = 0.42$ sec

τ_p was thus approximated with 0.2 seconds or 200 msec. Hence τ_s was taken as 20 msec and τ_f as 40 msec.

APPENDIX K:
TEST RESULTS FROM MINE TEST RIG

K.1 REAL TIME PULSE RESPONSES

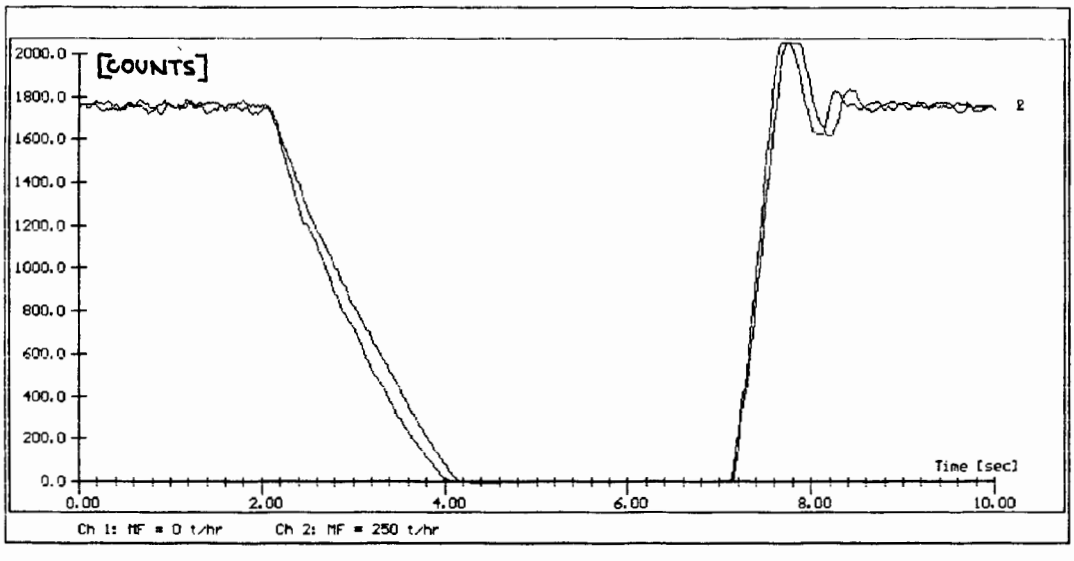


Figure K.1: Speed pulse response for v_0 = 100%

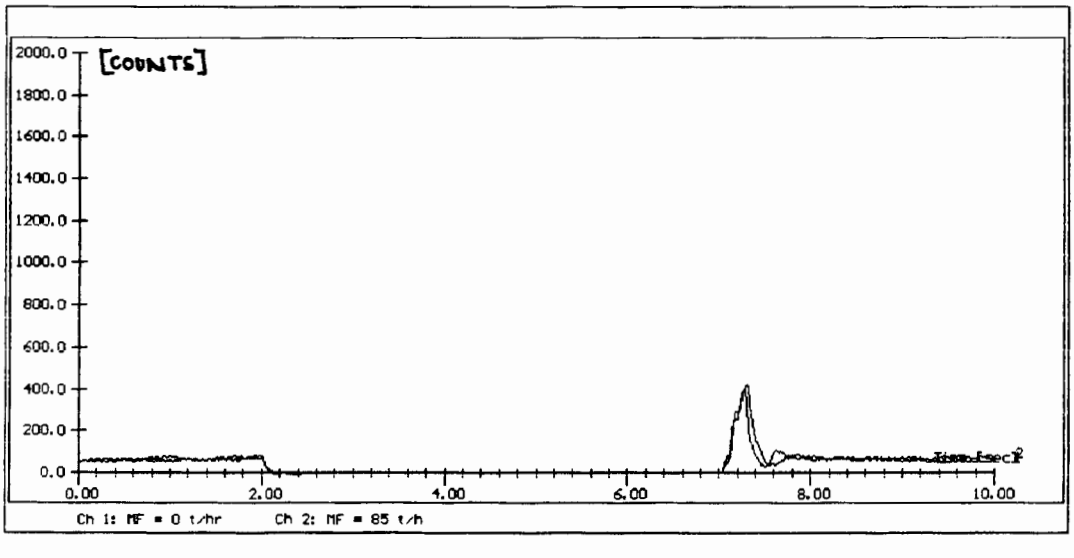


Figure K.2: Power pulse response for v_0 = 100%

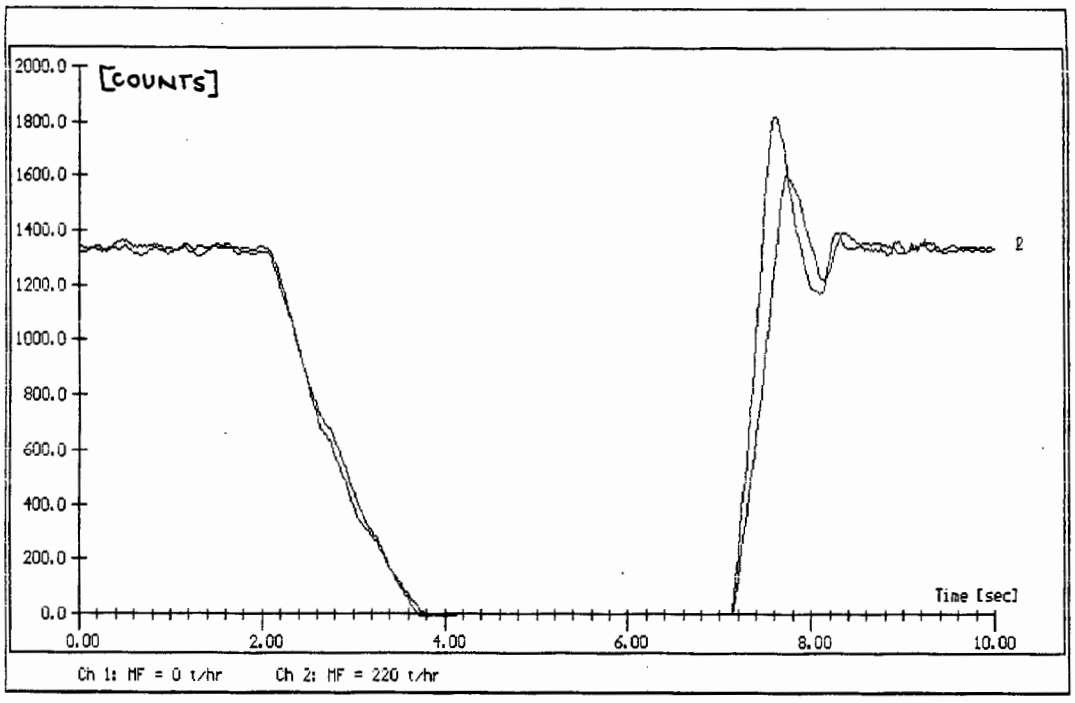


Figure K.3: Speed pulse response for v_o = 75%

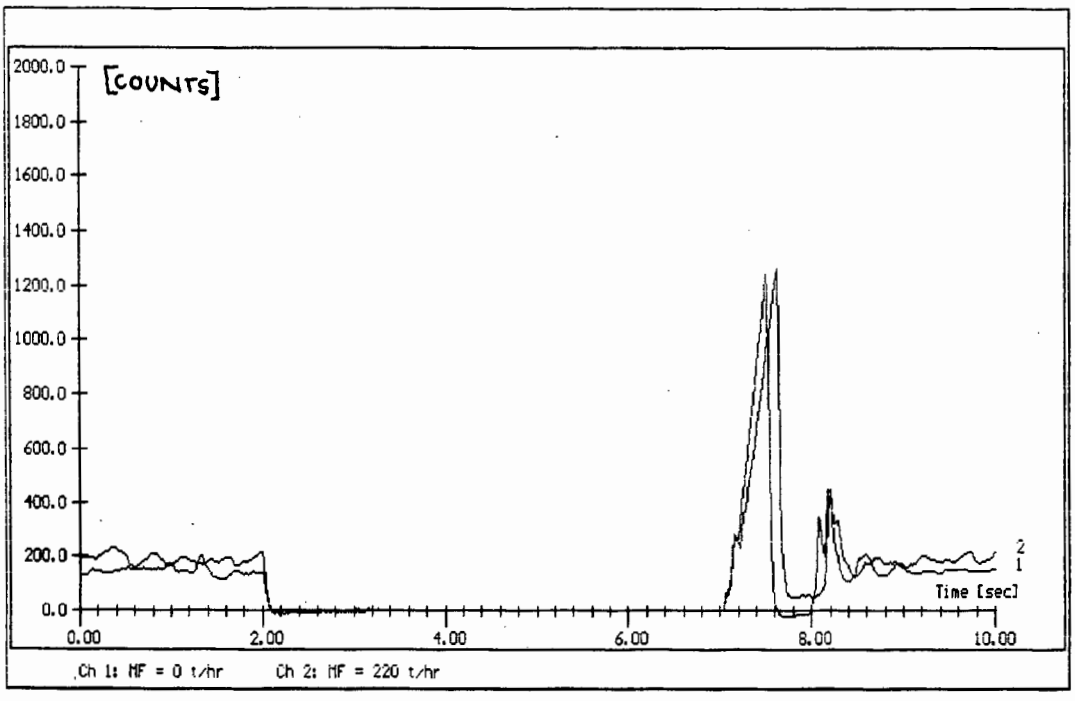


Figure K.4: Power pulse response for v_o = 75%

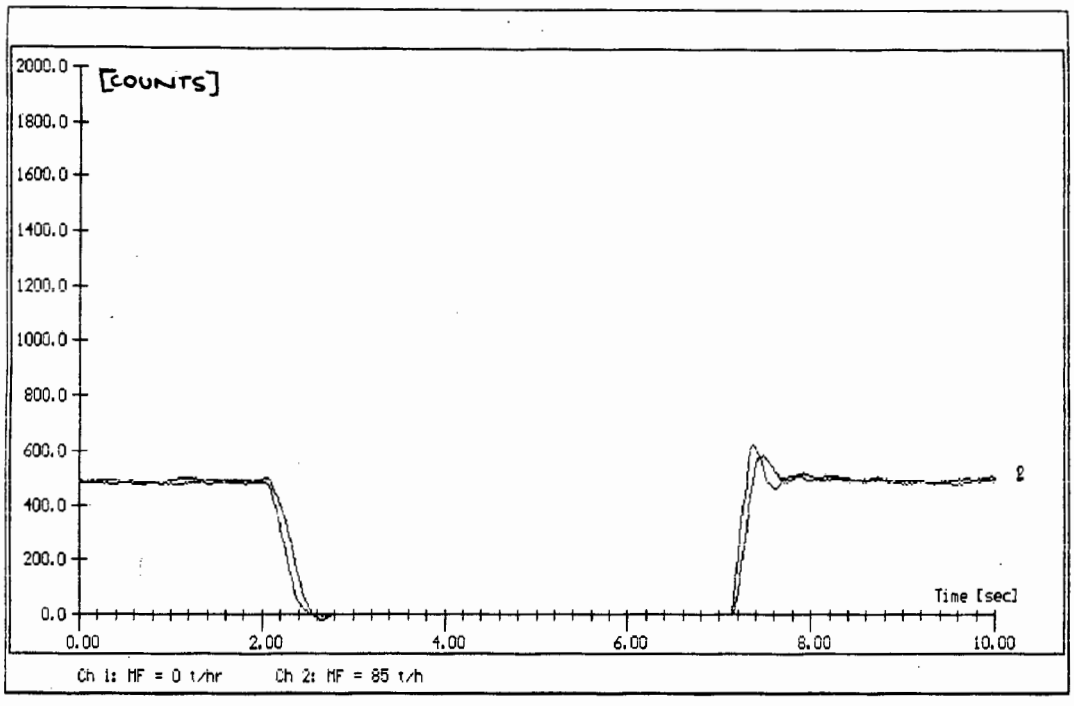


Figure K.5: Speed pulse response for v_0 = 25%

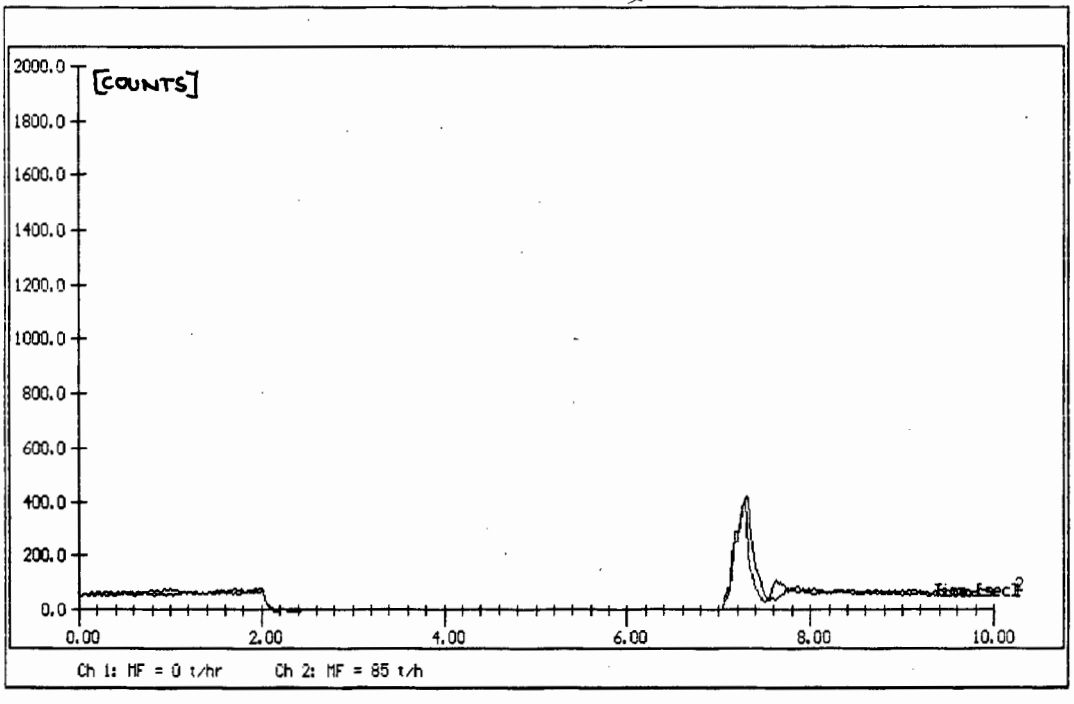


Figure K.6: Power pulse response for v_0 = 25%

K.2 CORRELATION CURVES FOR DIFFERENT FILTER TIME CONSTANTS

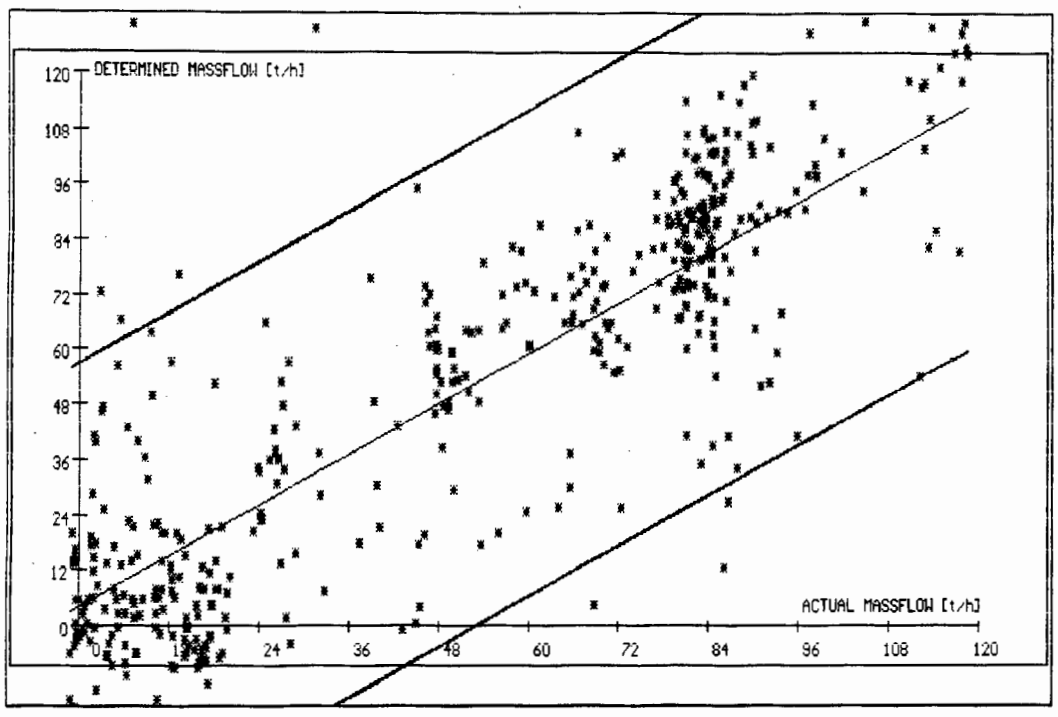


Figure K.7: Correlation curve for τ_f = 60 s

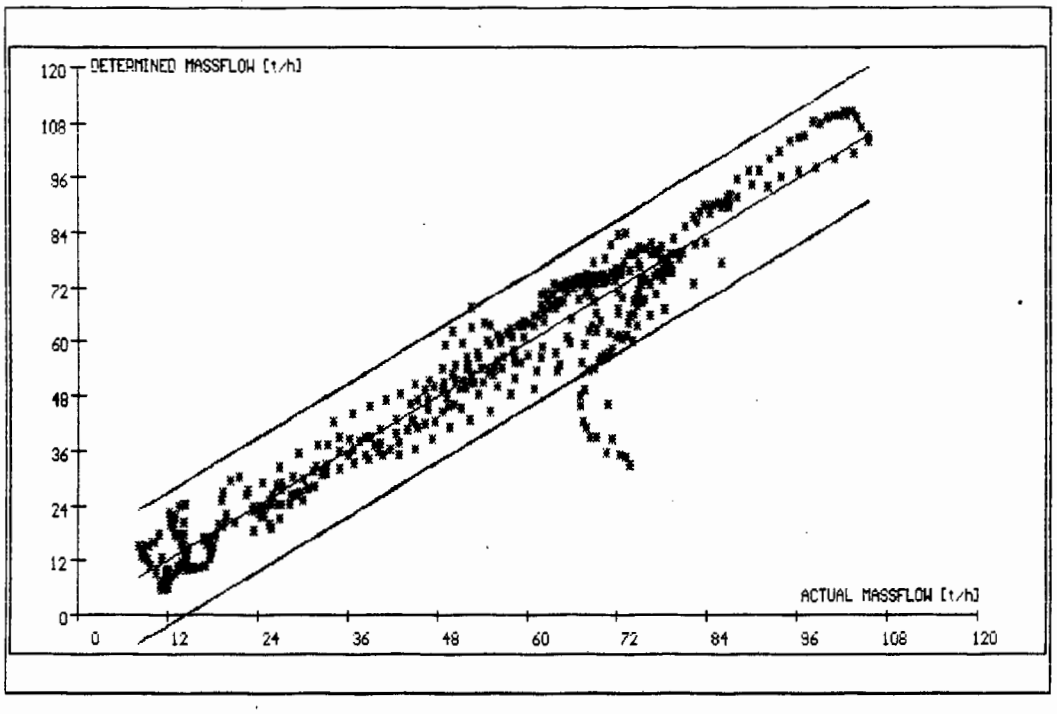


Figure K.8: Correlation curve for τ_f = 600 s

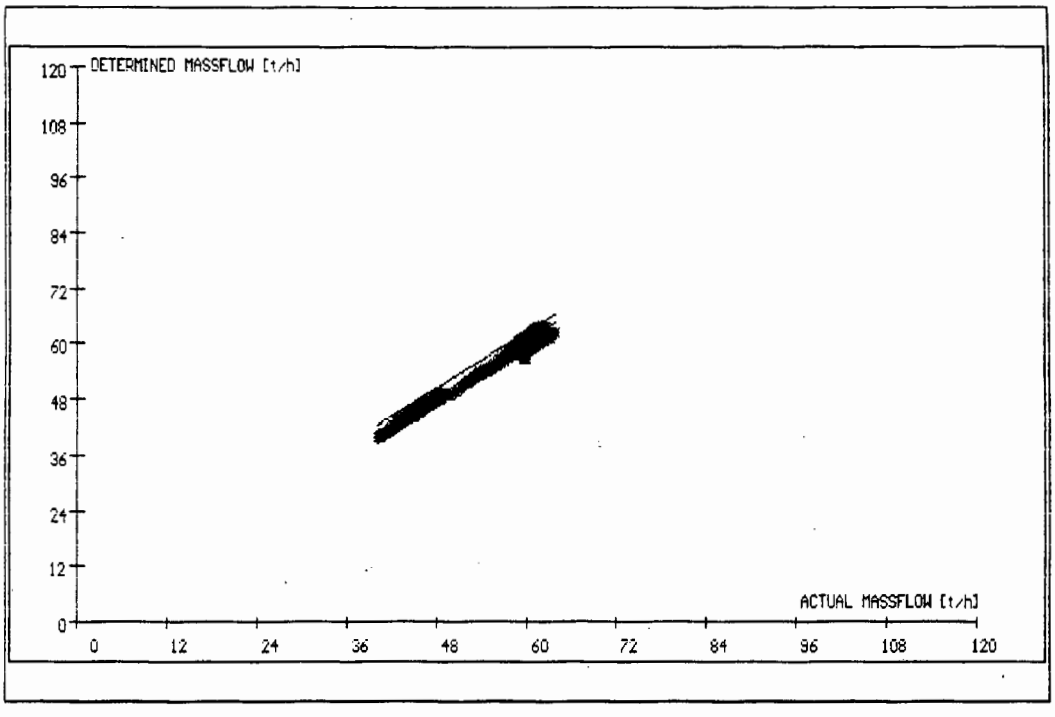


Figure K.9: Correlation curve for τ_f = 6000 s

K.3 TIME PLOTS AND CORRELATION GRAPHS FOR DIFFERENT SPEED SETPOINT

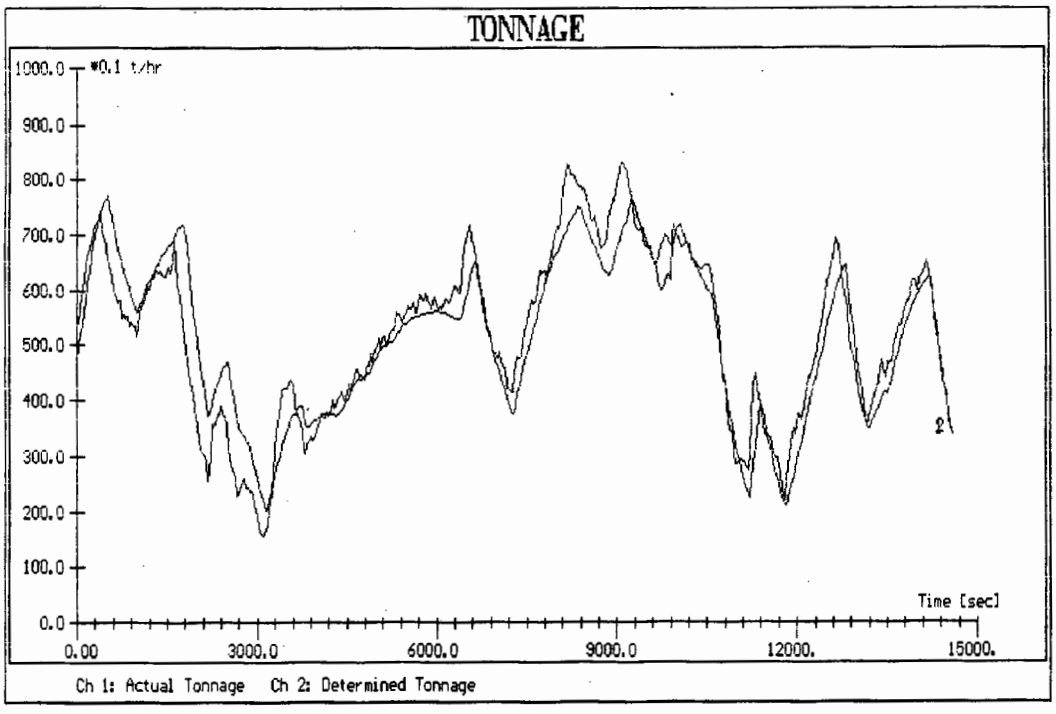


Figure K.10: Time plot for test done at v_0 = 100%

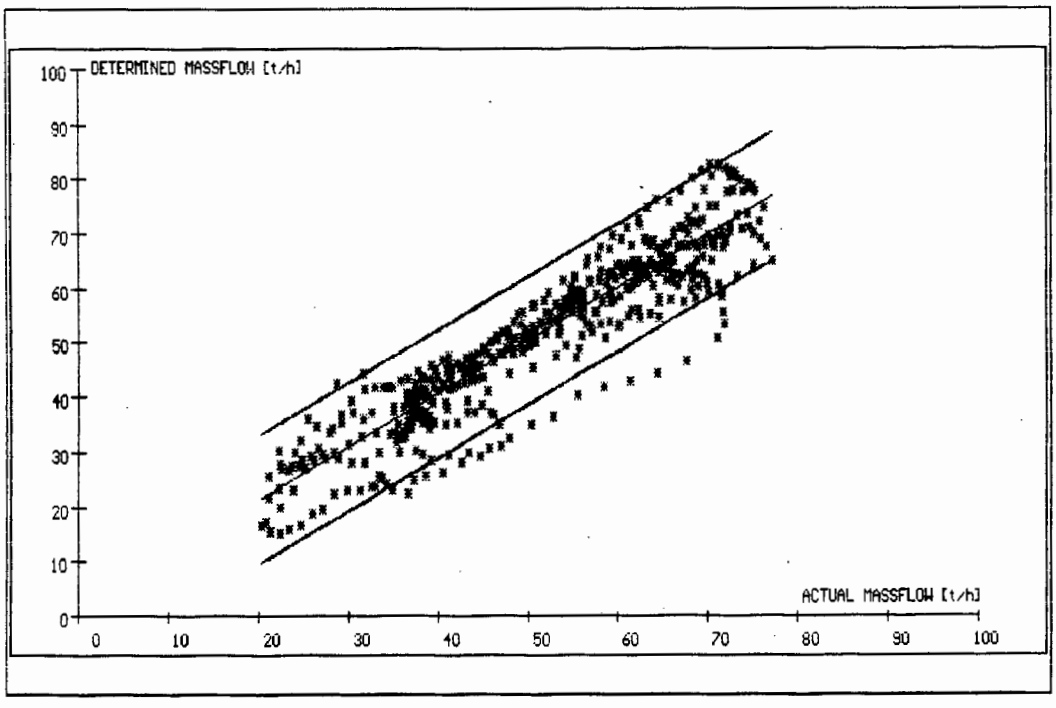


Figure K.11: Correlation graph for test done at v_0 = 100%

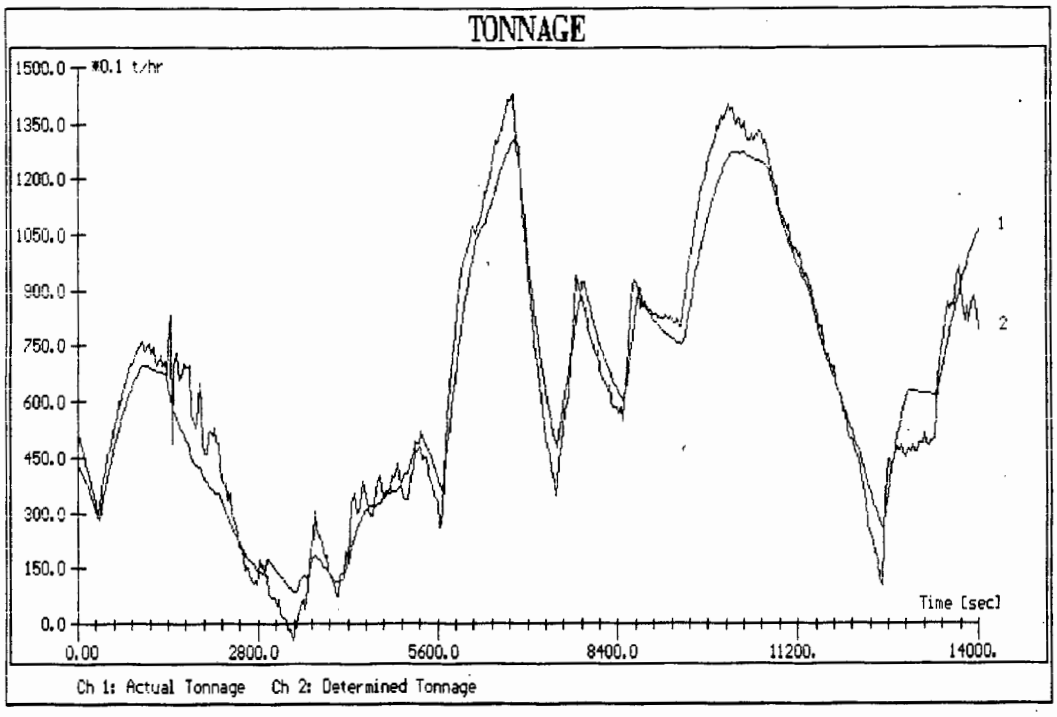


Figure K.12: Time plot for test done at v_0 = 75%

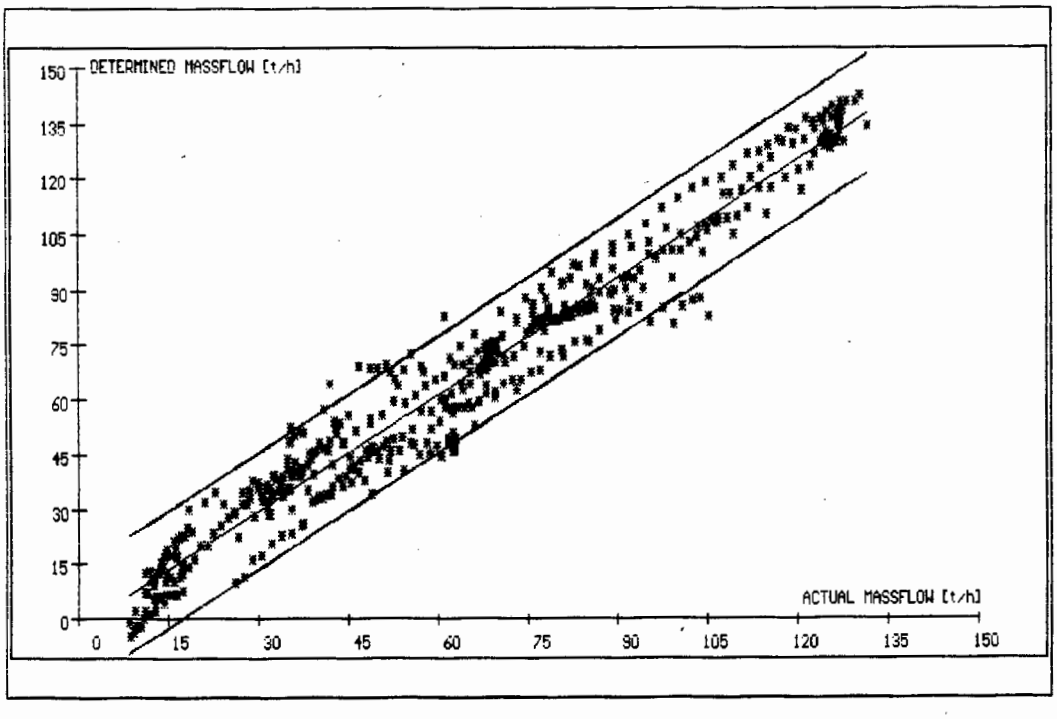


Figure K.13: Correlation graph for test done at v_0 = 75%

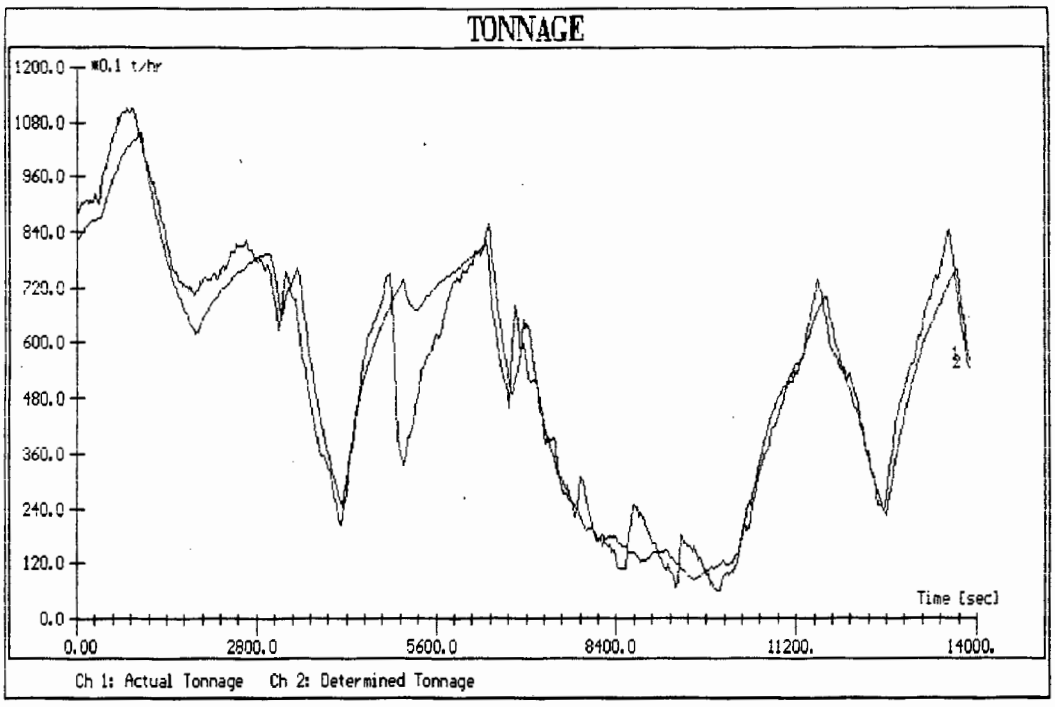


Figure K.14: Time plot for test done at v_0 = 50%

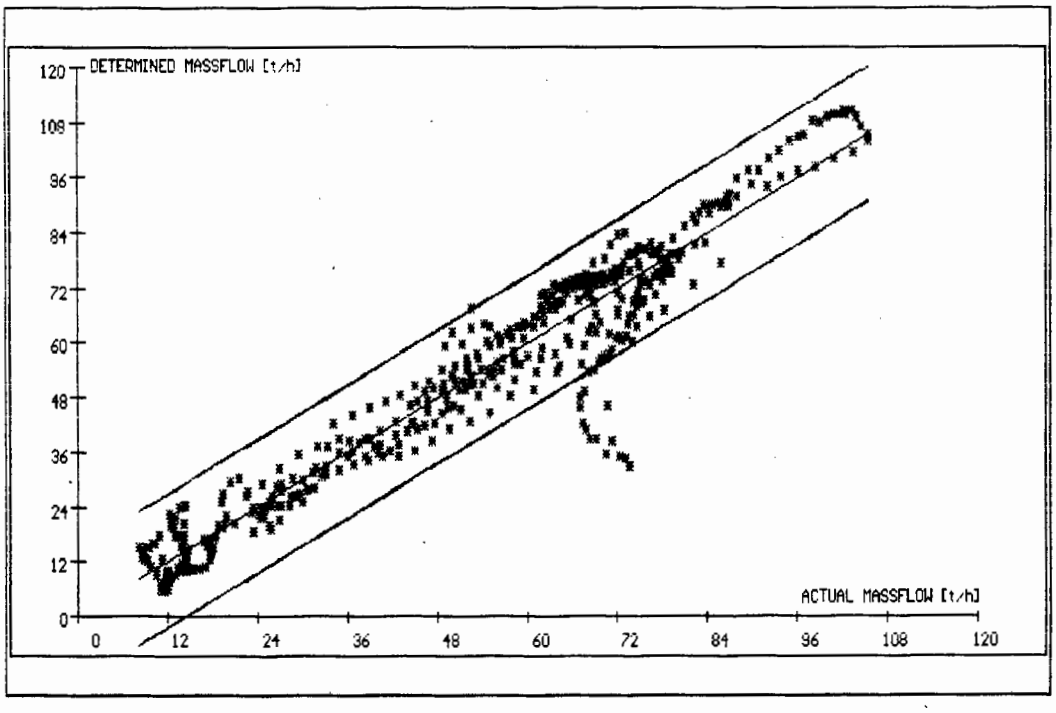


Figure K.15: Correlation graph for test done at v_0 = 50%

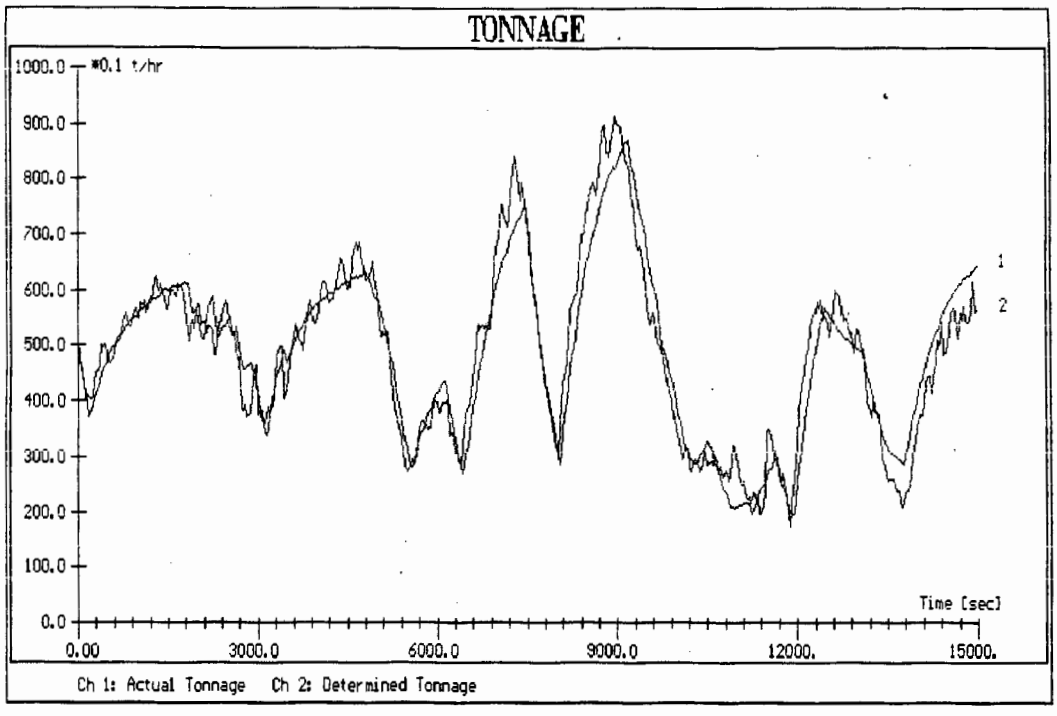


Figure K.16: Time plot for test done at v_0 = 25%

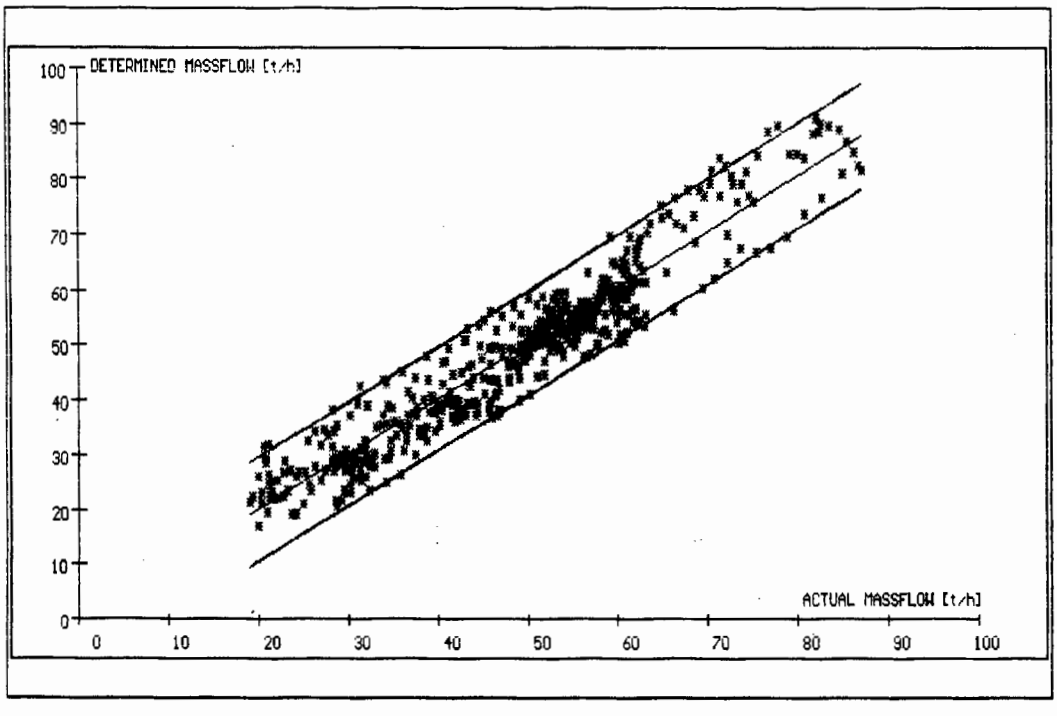


Figure K.17: Correlation graph for test done at v_0 = 25%

K.4 CROSS VALIDATION RESULTS

K.4.1 Initial Speed Setpoint = 75%

Y - Intercept = 2484.70

Slope = 3.78

Time Interval [sec]	Total Tons [t]	Error [%]
0<=t<=2000	29.09	0.463
2000<=t<=4000	10.31	-50.31
4000<=t<=6000	21.15	-17.25
6000<=t<=8000	51.83	-5.93
8000<=t<=10000	45.51	-2.34
10000<=t<=12000	57.01	-1.24
12000<=t<=14000	34.23	22.14
0<=t<=14000	249.14	-8.62

Table K.1: Table of cross validation results for v_0 = 75%

K.4.1 Initial Speed Setpoint = 50%

Y - Intercept = 2162.50

Slope = 7.01

Time Interval [sec]	Total Tons [t]	Error [%]
0<=t<=2000	47.59	-0.22
2000<=t<=4000	37.19	-11.28
4000<=t<=6000	33.92	-24.44
6000<=t<=8000	28.85	-12.77
8000<=t<=10000	7.42	-54.34
10000<=t<=12000	24.17	-20.66
12000<=t<=13890	26.43	-6.79
0<=t<=13890	205.57	-13.18

Table K.2: Table of cross validation results for v_0 = 50%

K.4.1 Initial Speed Setpoint = 25%

Y - Intercept = 1610.00

Slope = 9.13

Time Interval [sec]	Total Tons [t]	Error [%]
0<=t<=2000	28.94	0.88
2000<=t<=4000	28.27	0.97
4000<=t<=6000	27.27	8.09
6000<=t<=8000	27.55	2.33
8000<=t<=10000	21.42	-1.12
0<=t<=10000	133.44	2.35

Table K.3: Table of cross validation results for v_0 = 25%

K.5 RESULTS FROM CONTROLLED SPEED TESTS

K.5.1 Time Plots for different Filter Time Constant

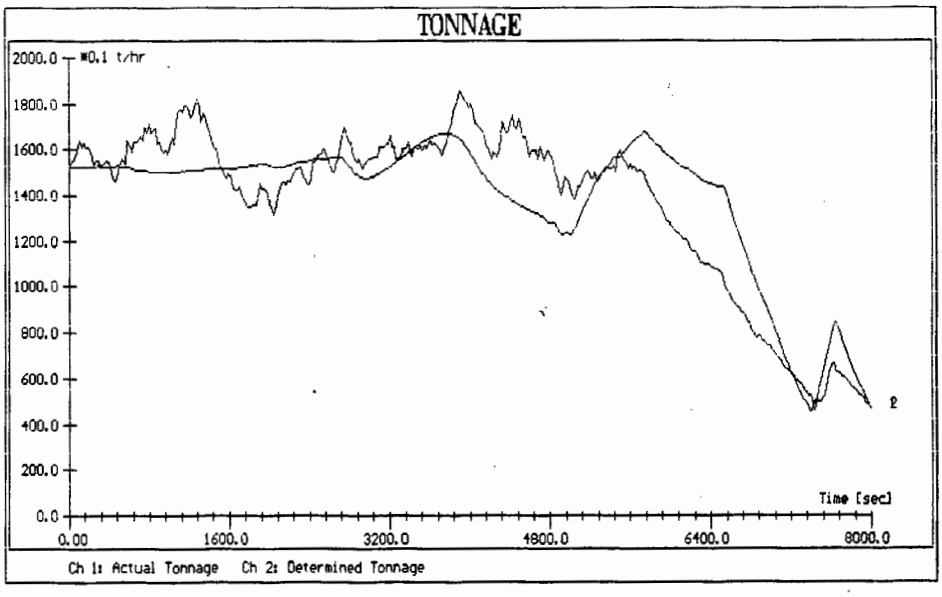


Figure K.18: Time plot for Test 2 when speed is controlled; τ_f = 600 s

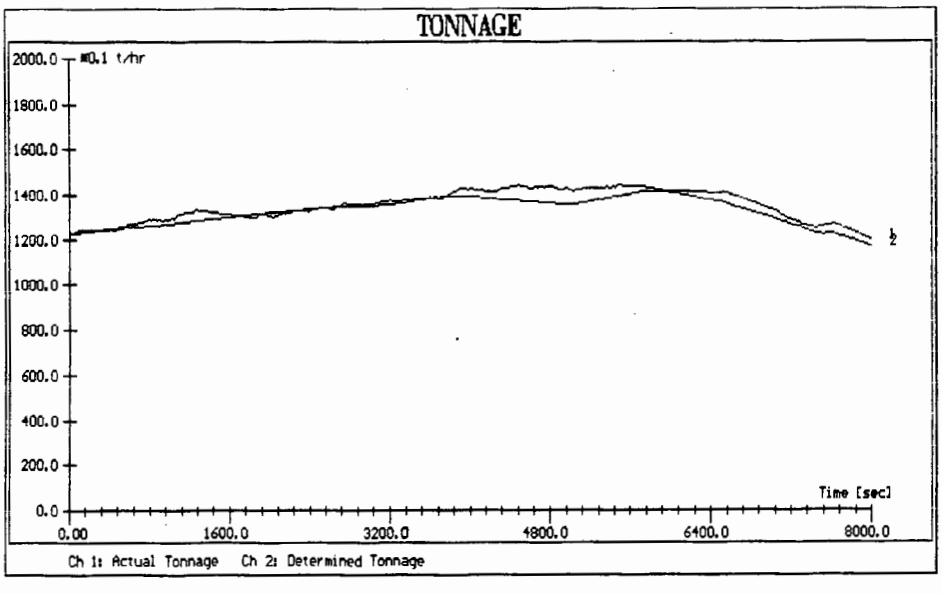


Figure K.19: Time plot for Test 2 when speed is controlled; τ_f = 6000 s

K.5.2 Correlation Graphs for different Filter Time Constant

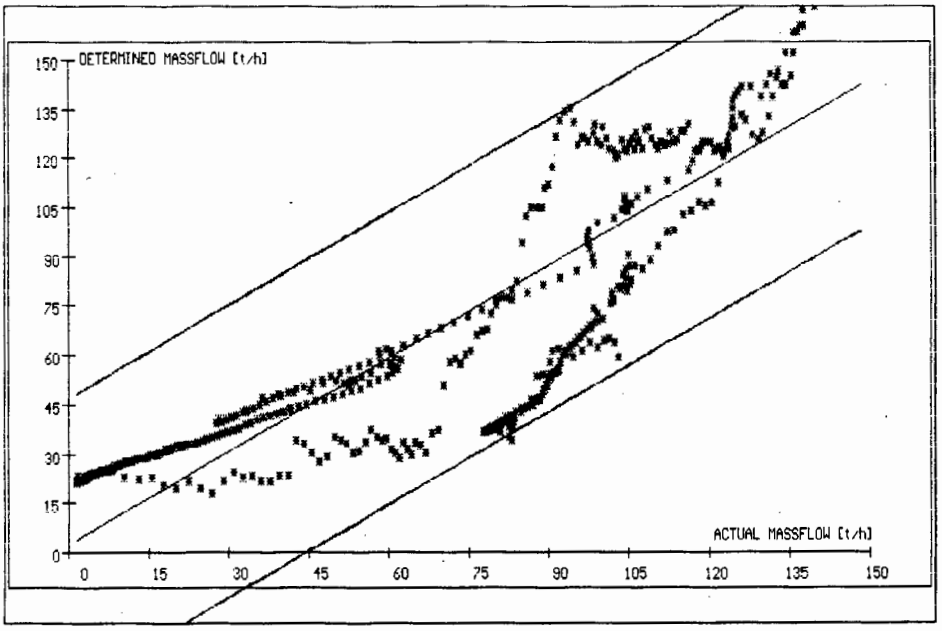


Figure K.20: Calibration graph for Test 1 when speed is controlled; τ_f = 600 s

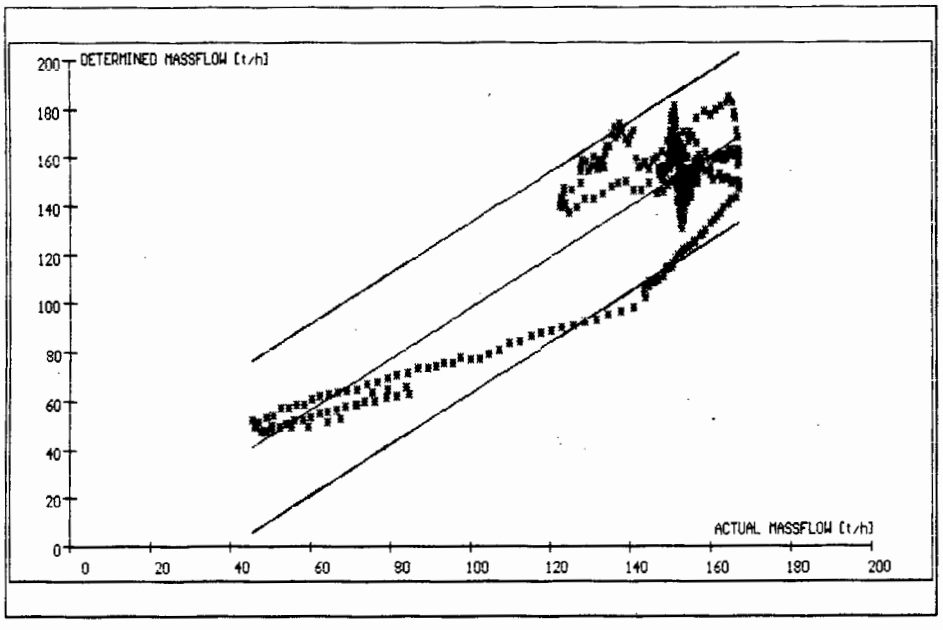


Figure K.21: Calibration graph for Test 2 when speed is controlled; τ_f = 600 s

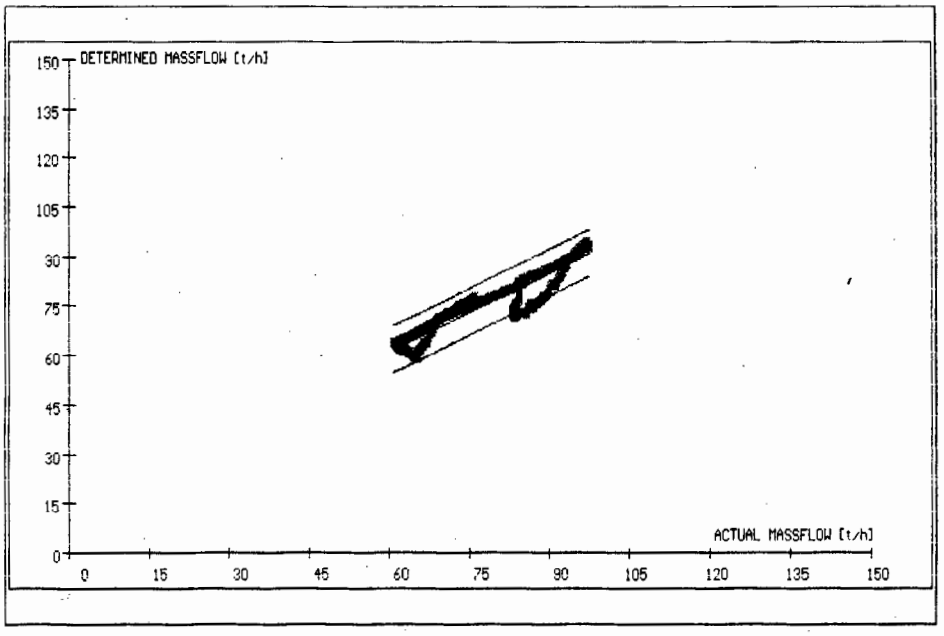


Figure K.22: Calibration graph for Test 1 when speed is controlled; τ_f = 6000 s

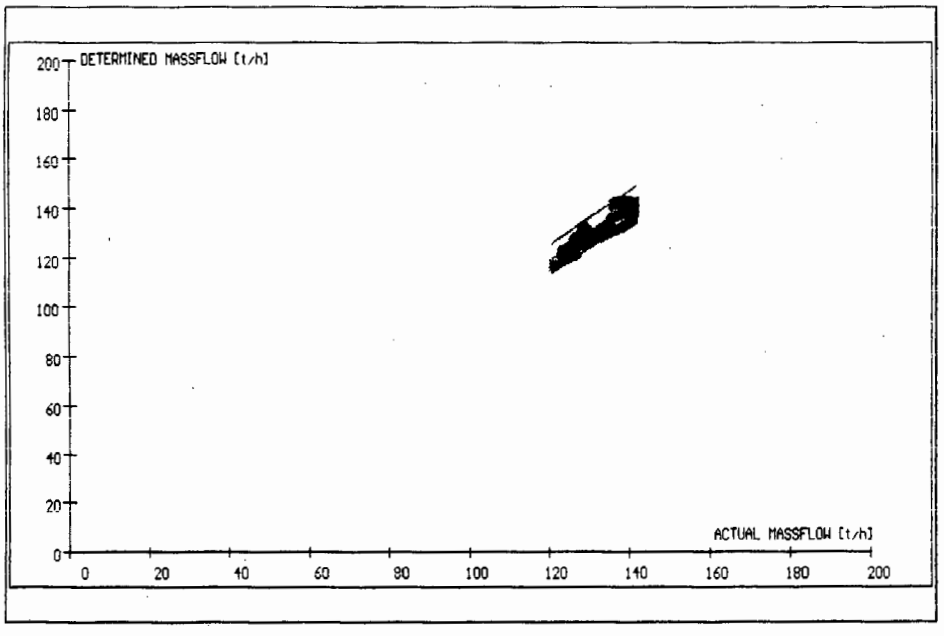


Figure K.23: Calibration graph for Test 2 when speed is controlled; τ_f = 6000 s