UNIVERSITY OF
SOUTHERN CALIFORNIA

A SYSTEM STUDY FOR THE APPLICATION OF MICROCOMPUTERS TO RESEARCH FLIGHT TEST TECHNIQUES

Quarterly Interim Status Report
Period 1 December 1982 through 28 February 1983
Grant NSG-4027

National Aeronautics and Space Administration
Ames Research Center
Hugh L. Dryden Flight Test Center
Edwards, CA 92523
Mr. Larry W. Abbott, Contract Officer's Technical Representative

Prepared By

Richard K. Smyth

Dr. Richard K. Smyth, Principal Investigator

SR-8302-FTT-512

February 1983

(NASA-CR-169869) A SYSTEM STUDY FOR THE APPLICATION OF MICROCOMPUTERS TO RESEARCH FLIGHT TEST TECHNIQUES Quarterly Interim Status Report, 1 Dec. 1982 - 28 Feb. 1983 (University of Southern California) 149 p N83-19487

Unclas
G3/60 08976

Table of Contents

1.0 INTRODUCTION

This interim report covers the activities of this Grant NSG-4027 for the third quarter of the grant year which ends 31 May 1983. This third quarter covers the period from 1 December 1982 through 28 February 1983. The following personnel are assigned to the grant research working 1/4-time under the grant funding:

Dr. Richard K. Smyth	Principal Investigator	Effective 1 June 82
Mr. Phillip Chan	Research Assistant	Effective 1 Jan 83
Mr. Fadi J. Kurdahi	Research Assistant	Effective 1 June 82
Mr. David Ho	Research Assistant	Effective 1 Jan 83
Mr. Carposforo Sosa	Research Assistant	Effective 1 June 82
Mr. Jean-Francois Soulard	Research Assistant	Effective 1 Jan 83

The assignments of the research assistants are as follows:

(1) Design, Code, & Test Air Data System Software and Investigate Hardware; Carposforo Sosa and Fadi Kurdahi

(2) Complete Hardware & Software for Spin Warning System Designed by Team #1 Utilizing SC-01 for Voice Generation; Philip Chan

(3) Complete Software, and Test the On-Board Simulation; Jean-Francios Soulard

(4) Provide Support in Software Coding, Compilation, and Object Code Loading into 68000 Microcomputer for all the Grant Research Teams; David Ho

In addition to the personnel assigned to the grant, other graduate students taking the EE560L microcomputer research course, and students taking directed research EE590L under Dr. R.K. Smyth, have performed research which have contributed to the grant's technical objectives. The final reports of these student team's research reports are attached as appendices to this Quarterly Interim Status Report. The Graduate Students and their contributions to the grant objectives follow:

1. On-Board Simulator — Team #1 (Report Appendix A)
    - Mr. Jeffrey Bluen
    - Mr. Jean-Francios Soulard
    - Mr. Mehdi Namakian
    - Mr. Charles Saleh

2. Spin Warning System — Team #1* (Report Appendix B)
   *used SC-011 voice generation chip
    - Mr. David Barry
    - Mr. David Ho
    - Mr. Renshan Tang
    - Mr. Mohammed Movahed-Ezazi

3. Spin Warning System — Team #2** (Report due May 83)
   **used TI voice generation chip
    - Mr. Steve Meier
    - Mr. Tieh Ku
    - Mr. Tom Wilkenson
    - Mr. Dave Adachi
    - Mr. David Chen

4. On-Board Simulation — Team #2 (Report due May 83)
    - Mr. Jeffrey Bluen (EE590L)
    - Mr. Horng-Ru Hwang (EE590L)

2.0 AIR DATA SYSTEM (ADS)

The ADS equations defined in section 2.4.1 of the Semi-Annual Interim Status Report (30 November 1982) have now been coded, and the cooding for the equations are being tested.

The work on ADS covers two sub-areas: installation of software support tools and implementation of the air data system software. With respect to the first sub-area, during this period we have finished the installation and testing of the software tools resident on the IBM 370/4341 (ECL-VIRGIL). This means that written software for the three areas of research (ADS, SWS, and OBS), can now be tested in the VERSAMODULE-01 system.

Upon completing testing of the M68000 cross-software (Jan 25) we proceeded to continue our work in the air data system. Currently we are in the area of developing software to test each of the 13 implemented ADS equations in order to obtain an estimate of their execution time. These estimates are required to design an appropriate time scheduling.

The progress in the two sub-areas is explained in detail in the following sections.

2.1 Software Support Tools

The system software support tools are applicable to all of the tasks on the grant, although the ADS team is checking out the tools. The following tasks have been completed.

<u>TASK 1</u>: The debugging of the communication software (CMSCPM) that handles file transfers between IBM (VIRGIL) and Computerm (EE560L) has been completed. A short user's guide, describing the use of CMSCPM was written.

<u>TASK 2</u>: Modification of the linker so that it produces a load map of the execution module being generated. A version of this map, displaying the absolute addresses of all the sections comprising the module, is required to

identify the entry point, i.e. the address at which execution starts. This task was achieved by modifying the EXEC (CMS command file) that invokes the linker, inserting new options and the corresponding file definitions.

TASK 3: Test the execution of PASCAL programs in the VERSAMODULE-01. In this respect we encountered two problems:

(a.) Incorrect loading of the execution module from Computerm into VERSAMODULE-01. The second line of the S-format file, containing the execution module, did not get loaded into the monoboard memory. This file, resident in Computerm, is transferred from the IBM using the utility CMSCPM, already mentioned in Task 1. We compared the files in the two systems and we did not detect any modification caused by transmission errors. We also checked the header of the S-format file and replaced it with others, used for S-format files that are known to load normally. The fault persisted. We have avoided this problem by duplicating the second line using a text editor.

(b.) Identification of the section in the execution module that contains the entry point. Motorola supplied us with the correct entry point for a module running under EXORMACS but did not give any information for running a program under VERSAMODULE-01. With the aid of the road map, we surveyed each section until we found the correct entry point.

We have now checked the test file supplied by Motorola as well as some preliminary programs used in ADS. So far, we have not encountered any problems regarding their execution in the VERSAMODULE-01.

2.2 Air Data System Software

The 13 ADS equations (table 2-1) have been coded. The ADS equation timing tests (table 2-2) have been completed and the timing procedure software has been written. The data acquisition routines have been completed and the testing of this software is underway. A potentiometer test fixture is being designed to permit testing of the various ADS algorithms which will emulate the sensor voltage levels with the pots and test for correct computation of the various air data parameters such as Mach number, altitude, etc.

TABLE 2-1

AIR DATA SYSTEM PARAMETER EQUATIONS

PTI (TOTAL PRESSURE):

```
PTI = QCI + PSI
```

MI (INDICATED MACH NUMBER):

```
for PTI/PSI <= 1.893
    MI = SQRT(5.0) * SQRT(((PTI/PSI) ** (2/7)) -1)
for PTI/PSI > 1.893
    x  = 1.839371 * (PSI/PTI)
    MI = SQRT((Ax - Bx - Cx**2 - Dx**3 - Ex**4 - Fx**5 - Gx**6 -
         Gx**9) /x)
```

MINF (FREESTREAM MACH NUMBER)

```
MINF = MI + DM
```

DM is an error correction obtained by interpolating in a look-up table

PSINF (FREESTREAM STATIC PRESSURE)

```
for MINF <=1
    PSINF = PTI / ((1 + 0.2 * MINF**2) ** (7/2))
for MINF >1
    PSINF = (PTI * A * (1-A)** (5/2) / 0.1839371
    where A = 1 / (7 * MINF**2)
```

QBAR (DYNAMIC PRESSURE)

```
QBAR = 0.7 * MINF**2 * PSINF)
```

QCC (CORRECTED AIRSPEED PRESSURE)

```
QCC = PTI - PSINF
```

KEAS (KNOTS EQUALIVANT AIRSPEED)

```
KEAS = MINF * 661.48 * SQRT(PSINF / 2116.22)
```

KCAS (KNOTS CALIBRATED AIRSPEED)

```
KCAS = 1479.1 * SQRT ((1 + (QCC / 2116.22) ** (2/7)) -1)
```

TABLE 2-1

AIR DATA SYSTEM PARAMETER EQUATIONS

(Page 2 of 3)

<u>HP (GEOPOTENTIAL OR PRESSURE ALTITUDE)</u>

```
let R = PSINF / 2116.22
for R > .223361
    HP = 145442 * (R ** .1092632365 - 1)
for .223361 => R > .0540328
    HP = 164219.39  - 20805.7 * Ln(PSI)
for .0540328 => R > .00856663
    HP = 710793.96 * A**2 - 645177.17
    where A = (.0540328 / R) ** .01463563358
for R <= .00856663
    HP = 81660.714 * A**2 - 162928.85
    where A = (.00856663 / R) ** .04097977402
```

<u>AINF (ANGLE OF ATTACK)</u>

let EA = f (MINF)

EA	MINF
.0055	0.0<
.0053	0.2
.0051	0.4
.0044	0.6
.0033	0.8
.0023	0.9
.0	>1.0

```
    AINFF = (1 + EA) * (ALPHA I)
```

<u>BINF (ANGLE OF SIDESLIP)</u>

```
    BINF = (1 + EB) * (BETA I)
    where EP = 0.0
```

ORIGINAL PAGE IS
OF POOR QUALITY

TABLE 2-1

AIR DATA SYSTEM PARAMETER EQUATIONS

GAMMA (FLIGHT PATH ANGLE)

GAMMA = THETA - AINF

HDGAMMA (ALTITUDE RATE)

HDGAMMA = 60 * MINF * CS * Sin(GAMMA)

where CS = Speed of sound per 1962 std. atmoshpere

HDOT (ALTITUDE RATE, TIME DERIVATIVE

let HAV(t) = 1/5 HP(t-i);

HDOT(t) = HAV(t) - HAV(t-1) [time interval = 1]

FQTY (FUEL QUANTITY)

FQTY (0) = 304.85 AND FUSED (0) = 0.0

if FFR(t) => 3.25 and FFR(t) <= 52.0 then FUSED(t) = FFR(t)

if FFR(t) < 3.75 or FFR(t) > 52.0 then FUSED (t) = FUSED (t-

and

FQTY (t) = FQTY (t-1) - FUSED(t)

when TOPOFF = 1 FQTY(t) is reset to 304.85

once LAUNCH = 1 never reset FQTY(t)

TABLE 2-2

TIMING TESTS FOR ADS COMPUTATIONS

(See Table 2-1 for ADS Parameter Equations)

PARAMETER	PARAMETER NAME	Execution time in milliseconds TEST 1	TEST 2	TEST 3	AVERAGE
PSINF *	Free Stream Static Pressure	85.70	81.33	87.48	84.84
QBAR	Dynamic Pressure	5.115	5.115	5.115	5.115
QCC	Corrected Airspeed Pressure	2.13	2.10	2.07	2.10
KEAS	Knots Equivalent Airspeed (true)	17.58	17.23	17.17	17.33
KCAS	Knots Calibrated Airspeed (indicated)	88.03	90.33	86.71	88.36
HP **	Geopotential or Pressure Altitude	55.95	170.38	87.12	104.48
AINF	Angle of Attack	14.145	14.145	14.145	14.145
GAMMA	Flight Path Angle	2.130	2.10	2.055	2.10
HDGAMMA	Altitude Rate (Computed for GAMMA)	39.43	40.48	40.48	40.13
FQTY	Fuel Quantity	3.165	3.150	3.150	3.16
	TOTAL	313.38	426.36	345.50	361.75
	MAX ITERATION RATE	3.19/sec	2.35/sec	2.89/sec	2.76/sec

SAFE INTERATION RATE 2/sec

* Two equations, selected by freestream Mach No. value

** Four equations, selected by freestream Mach No. value

ORIGINAL PAGE IS
OF POOR QUALITY

3.0 ON-BOARD SIMULATOR (OBS)

The student team cited in the introduction and lead by Bluer & Soulard produced a final report on the OBS which is included as Appendix A of this progress report. Messrs. Bluen and Soulard are continuing work on the OBS. Mr. Hwang is providing coding for the aircraft lateral equations of motion (3 degree-of-freedom) using the T-38 aircraft parameters for the equations. He is using the data from a NASA research report written by Mr. Teper of Systems Technology, Inc. (STI) for the equations and parameters. His program will be menu-driven and will permit the selection of various flight conditions to be used. Messrs. Bluen & Soulard will integrate Mr. Hwang's equations and software modules into the overall OBS software.

The modification and improvements to OBS team # 1's project are described below. The system concepts are covered in figures 3-1 through 3-5.

3.1 Implementation of Separate Motions

In the earlier form of this project the line-of-sight and the range between the two planes were generated by programmed functions. This choice implied that the host and the target were not actually moving independently. The new implementation generates separate geometry parameters for both airplanes. These parameters are updated during every time step of the simulation loop. The program calculates the Cartesian coordinates of the two planes (XH and YH for the host and XT and YT for the target) using the velocities (VELOCITY and TVELOCITY respectively) and the turning angles (PSI and PSIT respectively) given by the simulation loop. The variations are first calculated (DELTA-XH, DELTA-YH, DELTA-XT, DELTA-YT) and then added to the old values of the coordinates (XOH, YOH, XOT, YOT). At the beginning of the loop the newly computed values are assigned to the old value variables and the process is started again for the new time step.

In addition, both aircraft use separate simulation loops to generate the turning angle commands.

3.2 New Tracking Procedure

The "prediction" algorithm presented in the former report proved to be inappropriate for the use of the OBS. It has been replaced by a conventional guidance law, called the proportional guidance system, in which the parameters are calculated using actual geometry of the scene (as opposed to "anticipated" position of the target as used in the "prediction" algorithm). The parameters used by the proportional guidance are:

- The line of sight rate, SIGMA
- The guidance factor, LAMBDA
- The guidance gain, GUIDGN

These parameters are calculated with the updated geometry given by the program, that is, passed to the guidance law that generates an optimal command for the host. The pursuit loop can be closed automatically (as is done now) or by an actual pilot using a display screen.

3.3 Providing Target Maneuvers

At the beginning of the simulation, the pilot or the operator is asked the initial geometry of the scene. He must give the coordinates of the target relative to his starting position. The host starting position is taken as the origin of the grid. The target maneuver is detemined by its initial turning angle command, or alternately, can be programmed as a sequence of commands.

3.4 Providing Different Types of Aircraft

Two possibilities are considered:

- Providing fixed pre-defined types, chosen in a menu, or
- Providing ad libitum types under reasonable limits

Choosing pre-defined types could ease the initialization procedure for the user but, on the other hand, ad libitum types allow an infinite range of

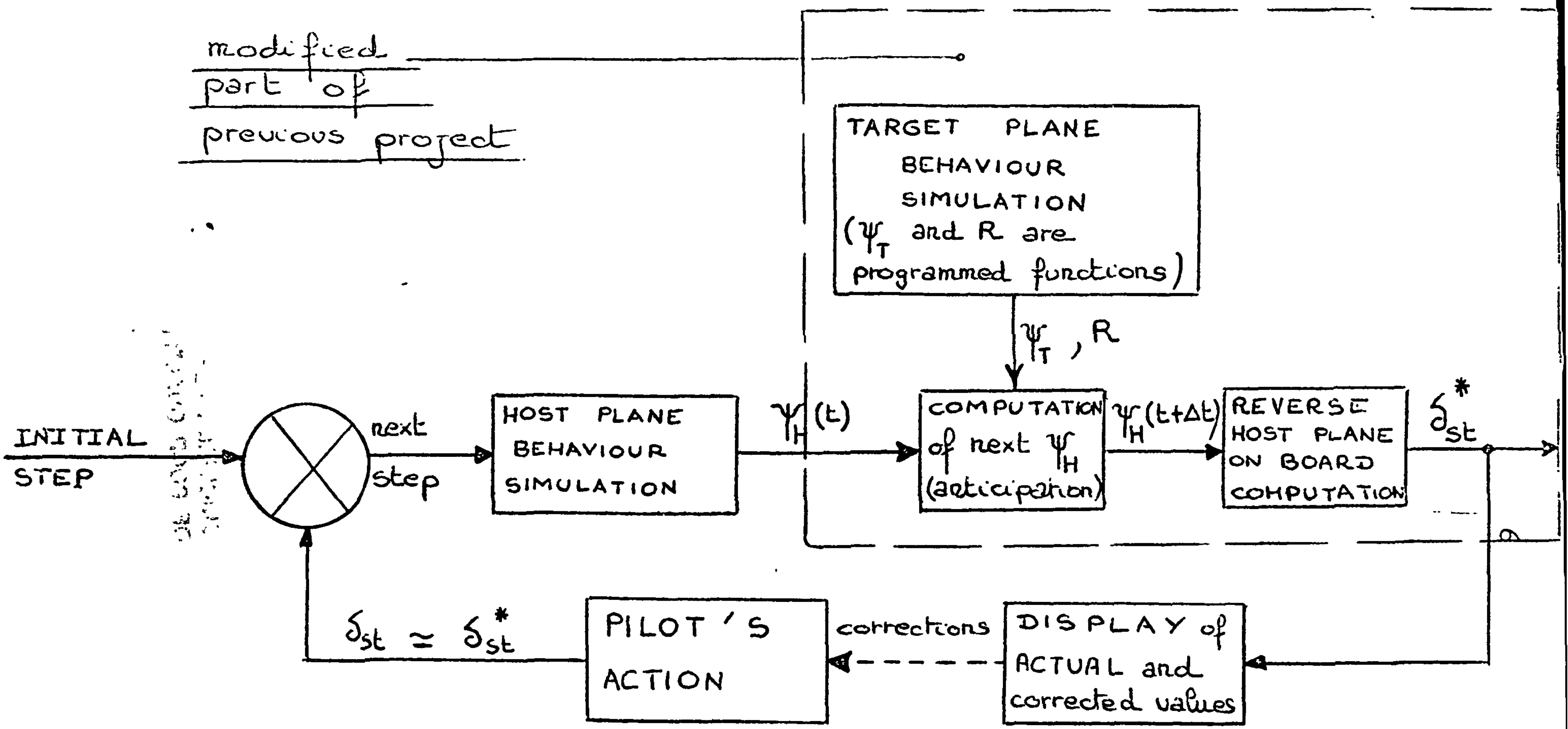


FIGURE 3-1

PREVIOUS PROJECT

- OVERALL FLOW CONTROL DIAGRAM -

- with modified part appearing -

(δ_{st}^* = anticipated δ_{stick} in order to shoot the target next Δt).

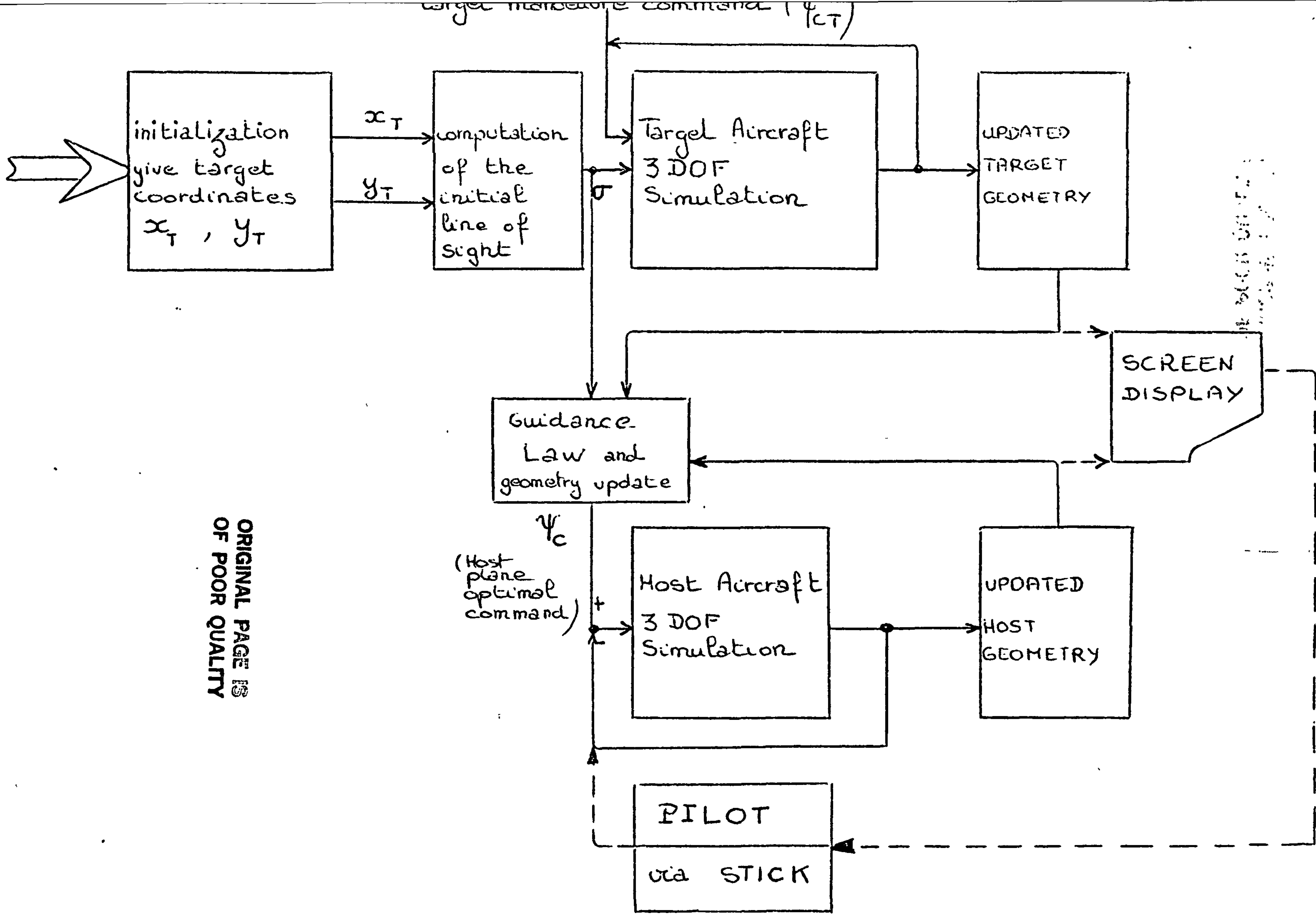


FIGURE 3-2

ON BOARD SIMULATOR FOR AIR TO AIR INTERCEPTION

- GENERAL FLOW ORGANISATION -

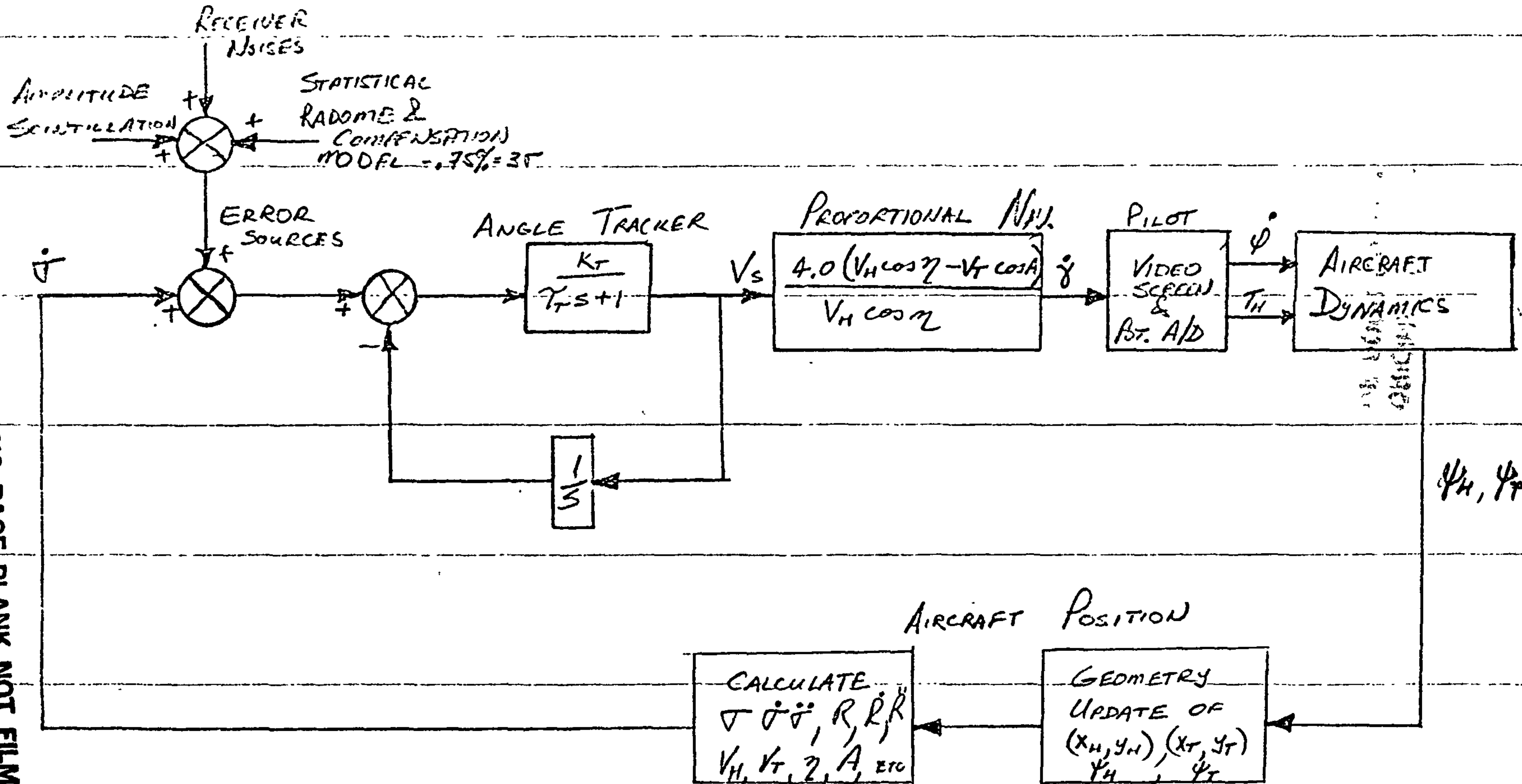


FIGURE 3-4 ANALYTICAL BLOCK DIAGRAM FOR OBS

H = HOST AIRCRAFT

T = TARGET AIRCRAFT

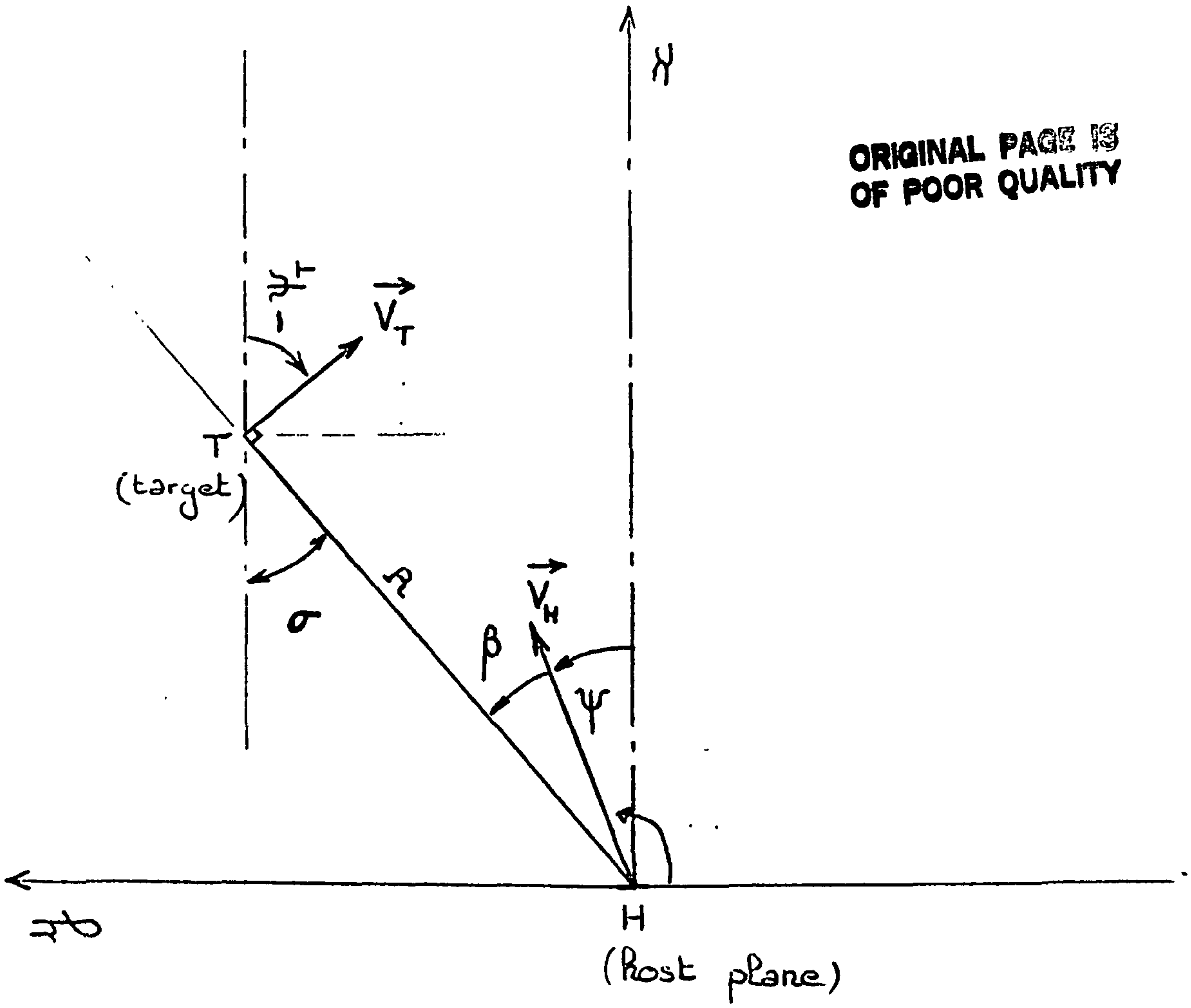
K_T = ANGLE TRACKER GAIN

τ_T = ANGLE TRACKER TIME CONSTANT

PRECEDING PAGE BLANK NOT FILMED

ORIGINAL PAGE IS OF POOR QUALITY

FIGURE 3-5 Flight Path Geometry



notation correspondance	Basic relations
V_H = VELOCITY V_T = TVELOCITY ψ = PSI ψ_T = TPSI R = RANGE σ = NEWSIGMA	geometry $R\,\dot{\sigma} = V \sin(\sigma - \psi) - V_T$ proportional guidance $\dot{\psi} = \lambda\,\dot{\sigma}$

aircraft but require more work from the user. An operation manual may be suitable but a decision has not yet been made.

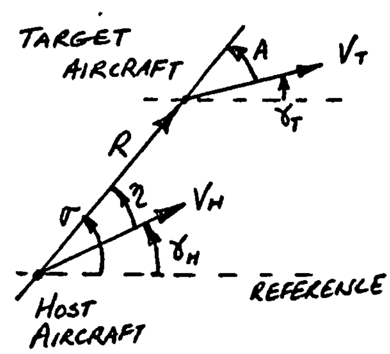
3.5 Proportional Navigtion

Proportional navigation tries to keep the proper lead angle from host to target aircraft always forcing the host line-of-sight rate, $\dot{\sigma}$, to zero. It determines the host aircraft velocity vector turning rate which is proportional to the line-of-sight rate. The host aircraft velocity heading, γ_H, changes until $R\dot{\sigma}$ becomes zero. The most important component of this computation is the accurate measurement of the spatial line-of-sight rate.

The aircraft is assumed to have a large accurate antenna used to give line-of-sight information and is assumed to be space stabilized using free gyros. We will further assume that the antenna is slaved to keep its axis perfectly aligned with the gyro axis, so we will not have to model gimbal and torquer dynamics. Initially, no noise on the steering signal is assumed.

The proportional relationship between $\dot{\sigma}_H$ and $\dot{\gamma}_H$ is constantly readjusted by the closing rate as determined from the changing geometry. The relative velocity is divided by the host aircraft velocity and multiplied by the assumed Λ of 4.0.

λ steering gain
V_R closing rate along R
V_{HR} host aircraft closeing rate along R
σ line-of-sight
η look angle
γ velocity vector direction
$\dot{\gamma}$ turning rate



$$\dot{\gamma}_H = \lambda\dot{\sigma} = \Lambda\lambda\dot{\sigma} = \Lambda\frac{V_R}{V_{HR}}\dot{\sigma}$$

$$= \Lambda\left[\frac{V_H\cos\eta - V_T\cos A}{V_H\cos\eta}\right]$$

4.0 SPIN WARNING SYSTEM

Mr. Philip Chan fixed Team #1's breadboard voice generator using the SC-01 chip. During early February he had the breadboard speaking words and phrases using phonemes programmed into the system using a CRT keyboard. He is programming the works and phrases contained in the SWS matrix shown in table 4-1. The phoneme method of generating words used by the SC-01 chip appear to be a flexible and feasible method.

TABLE 4-1 STALL WARNING & COMMAND TO RECOVERY

Rate DC ±10	Discrete 1	Discrete 2	Discrete 3	Absence of Discrete 1,2,3
	ASYMMETRIC THRUST			
YAW RATE	AB BLOWOUT	AB STALL	MIL STALL	SYMMETRIC THRUST (NO BLOWOUT OR STALL)
Less than 40^{0}/sec	"Unload" If IAS $<$ 150 KTS "Left or Right (Lighted AB) Engine Mil"	"Unload" "Left or Right (Stalled) Eng Idle" If IAS $<$ 175 KTS "Left or Right (Unstalled) Eng Mil"	"Unload" If IAS $<$ 150 & Altitude $<$ 15K "Left or Right Eng Idle" (Unstalled Eng)	No Warning
Less than 50^{0}/sec but greater than 40^{0}/sec	"Unload" "Left or Right Engine Idle"	"Unload" "Both Engines Idle"	"Unload" "Both Engines Idle"	"Unload"
Greater than 50^{0}/sec	"Left or Right (Lighted AB) Engine Mil" "Stick Full Left or Right" "Stick Full Fwd"	"Both Engines Idle" "Stick Full Left or Right" "Stick Full Fwd"	"Both Engines Idle" "Stick Full Left or Right" "Stick Full Fwd"	"Stick Full Left or Right" Push Stick into turn

IAS DC ±10 0-10

Stick Pos Pitch ±IOU
Roll ±IOU

AOA = Angle of Attack
A_Y = Lateral Acceleration
ALT = ±10V to 50K
UNLOAD = TN g's $\rightarrow$ 1g
IAS = Indicated Airspeed (in knots)

After any warning from table above:
If Yaw Rate $<$40°/sec for two second "Center Controls".
If Yaw Rate $<$30°/sec for two seconds & Airspeed $>$120 Kts "Recovery Complete"
AOA $<$ -10° and Ay $>$ .1g "Stick Half-Aft and Hold": then;
If AOA $>$ 0 for 5 sec "Recovery Complete".

Either engine above 1215° (EGT turbine limit temp) for 3 sec "Left" or "Right Engine Off"

5.0 ACTIVITIES PLANNED FOR NEXT (LAST) QUARTER

The Air Data System software should be completed and tested using simulated sensor signals through the analog-to-digital converter card. The ADS software should be available for NASA Dryden to use on their flight simulator by the end of May 1983.

The on-board simulator will integrate the three degree-of-freedom (3 DOF) equations with the intercept algorithms. Emphasis will be placed on generalizing the OBS to tasks other that the air-to-air intercept mission currently programmed.

The Spin Warning System effort being done by the team #2 will provide a final research report to be incorporated in the Grant Final Report produced in May 1983. Mr. Chan will continue working on the Team #1 and will implement the PASCAL-based software on the 68000 microcomputer and integrate the software with the SC-01 voice generator.

APPENDIX A

ON-BOARD SIMULATOR

TEAM #1 FINAL REPORT

ORIGINAL PAGE IS
OF POOR QUALITY

UNIVERSITY OF SOUTHERN CALIFORNIA

EE 560 1

ON BOARD SIMULATOR

FOR

AIR TO AIR INTERCEPTOR .

PROJECT BY :

Jeffrey Bluen ,
Mehdi Namakian ,
Charles Saleh ,
Jean-Francois Soulard .

TABLE OF CONTENTS

I N T R O D U C T I O N

The project presented here is a synthesis of two subjects suitable for EE 560L, the graduate research course for Advanced Microcomputer Applications at the University of Southern California. The two original parts from which this work derives are:

--- On Board Flight Simulator (Project #13.0)

--- Air-to-Air Intercept Mode (Project #18.0)

The On Board Simulator part contains a three degree-of-freedom Aircraft Behavior Simulation, providing parameters used by the Interception procedure. These parameters could also be used for verifying closed loop performance before flight.

The Air-to-Air Intercept Mode is a software package integrated in the simulation process that generates a Target motion and performs a tracking procedure that predicts the most likely next target position, for a defined time step. This procedure also updates relative position parameters and gives adequate fire commands. The simulation of the input data, provided by an angle tracker, is done by pre-chosen programmed functions. This allows a wide range of target behavior as well as the full control of it when testing the procedures.

ON BOARD SIMULATOR

Preventing mishaps and saving tremendous amounts of money and time are the motivators for the construction of this 16-bit microcomputer simulation. The immediate benefits are derived from the fact that the On Board Simulator (OBS) is intended to use the existing hardware - aircraft flight computer. It can be easily reprogrammed to simulate different aircraft, and it also uses an inexpensive processor. It can provide parameters to check out the aircraft flight computer as an added side benefit.

The OBS in effect can use part of the host Aircraft's electronics to close the loop for the simulator thus spreading the computing load. A simulator of this type previously required the computing ability of much larger units and so large simulations were built justifying the greater computer expense. With the advent of the small, efficient, and computationally powerful 16-bit microprocessor, a new approach to simulation is possible. It can now execute the requisite equations in a real time sense enabling flight simulation in this hybrid configuration.

The On Board Simulator will be connected to the actual aircraft via the onboard computer and pilot commands. It will receive aircraft flight commands from the pilot, will simulate the aircraft dynamics and return to the aircraft flight computer updated parameters of the vehicle geometry.

Various aerodynamic and geometry equations can be programmed depending upon the type and model of aircraft being modeled.

The proposal to EE 560L included a 3 DOF of aircraft dynamics which would receive information from the pilot joystick as he appraises the engagement geometry seen on his video display. His input was a command to the host aircraft roll rate.

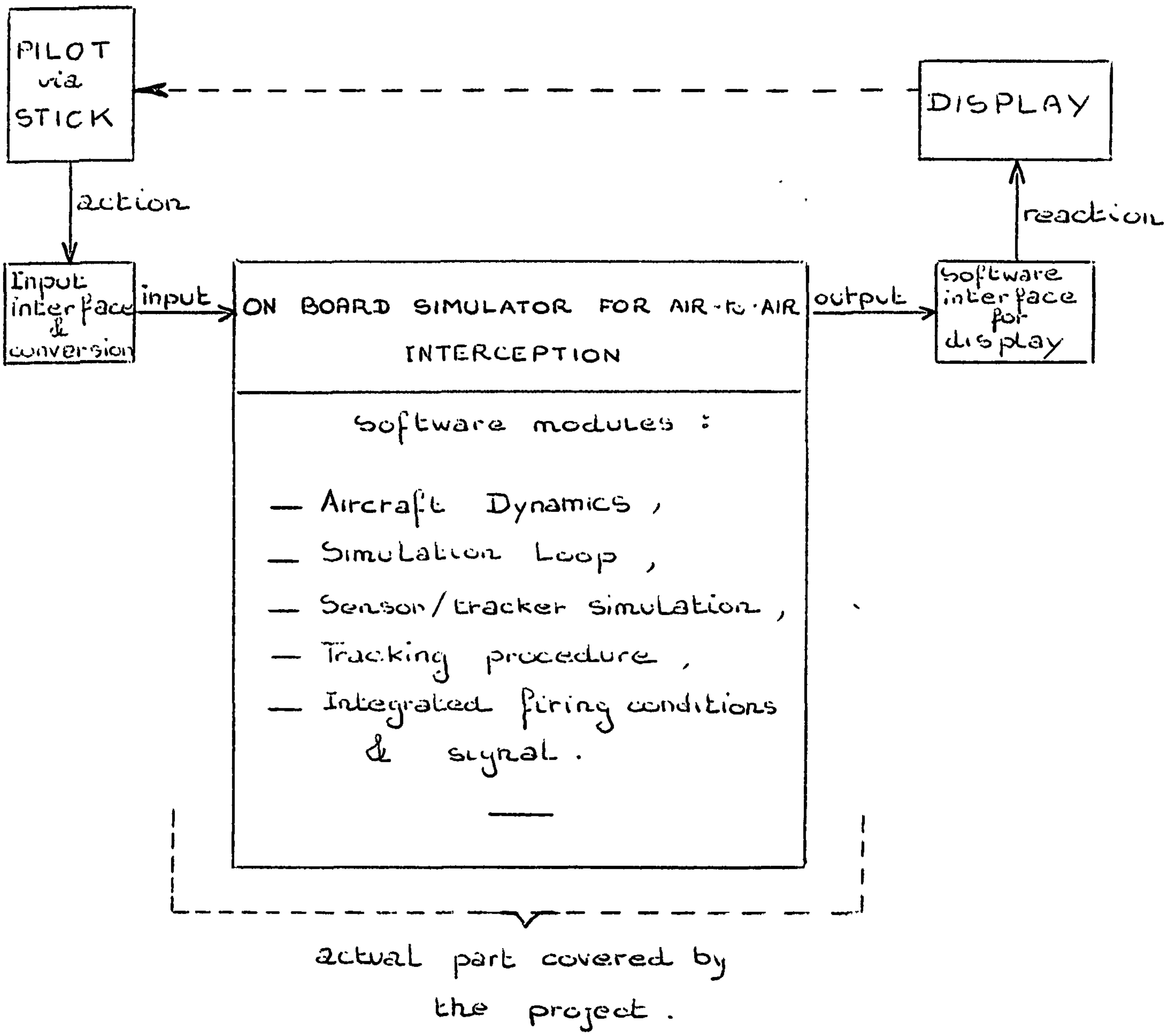
The approach to building the On Board Simulator began with the high level language, PASCAL, three degree-of-freedom simulation of the dynamic parameters. The working model was designed and tested and results are included. The remaining step is to download to object code in the Motorola 68000 microcomputer. This has not yet successfully been done because of unavailability of the required software and hardware tools.

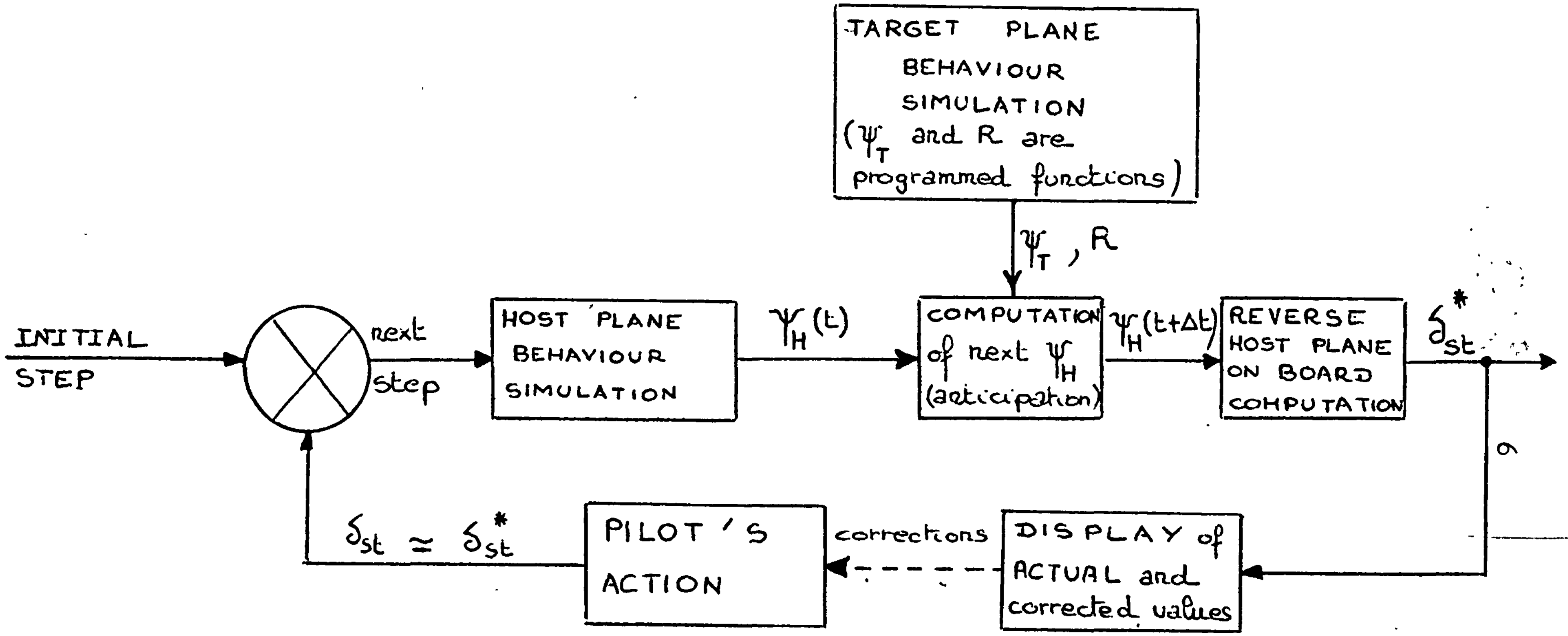
A preliminary program designed to test the CRT driver and A/D conversion is also included. It was not fully implemented. (See section on preliminary program.)

AIR-TO-AIR INTERCEPTION

The air-to-air interception procedure was implemented using different modules: a tracking process using both the parameters passed by the simulation and a target motion generator, that could be reprogrammed at one's will. According to the data provided by both the generator and the simulation, the next target position of the target is estimated and the optimal next position of the host aircraft is calculated.

On Board Simulator Architecture





- OVERALL FLOW CONTROL DIAGRAM -

(δ_{st}^* = anticipated δ_{stick} in order
to shoot the target next Δt).

ORIGINAL PAGE IS
OF POOR QUALITY

A I R C R A F T   D Y N A M I C S

The focus of this project is the simulation of both the Host (H) and the Target (T) planes motions. The Host plane's pilot gives orders to his machine and we had to recreate the response of the instruments and the structure to these commands. That is why we need some fundamental Aircraft Dynamics results.

In this introducing short study, we only try to show very basic data about Aircraft maneuvers and the parameters on which the pilot can act. The equations and their computing are treated in the project.

I. THE PILOT'S COMMANDS:

To control the plane's motion, the pilot has three main mechanisms:

--- The THROTTLE, commanding the thrust, that is the power given by the engine. For a definite aerodynamic configuration of the plane, at a constant altitude $Z=Z_0$, the throttle commands the speed of the plane.

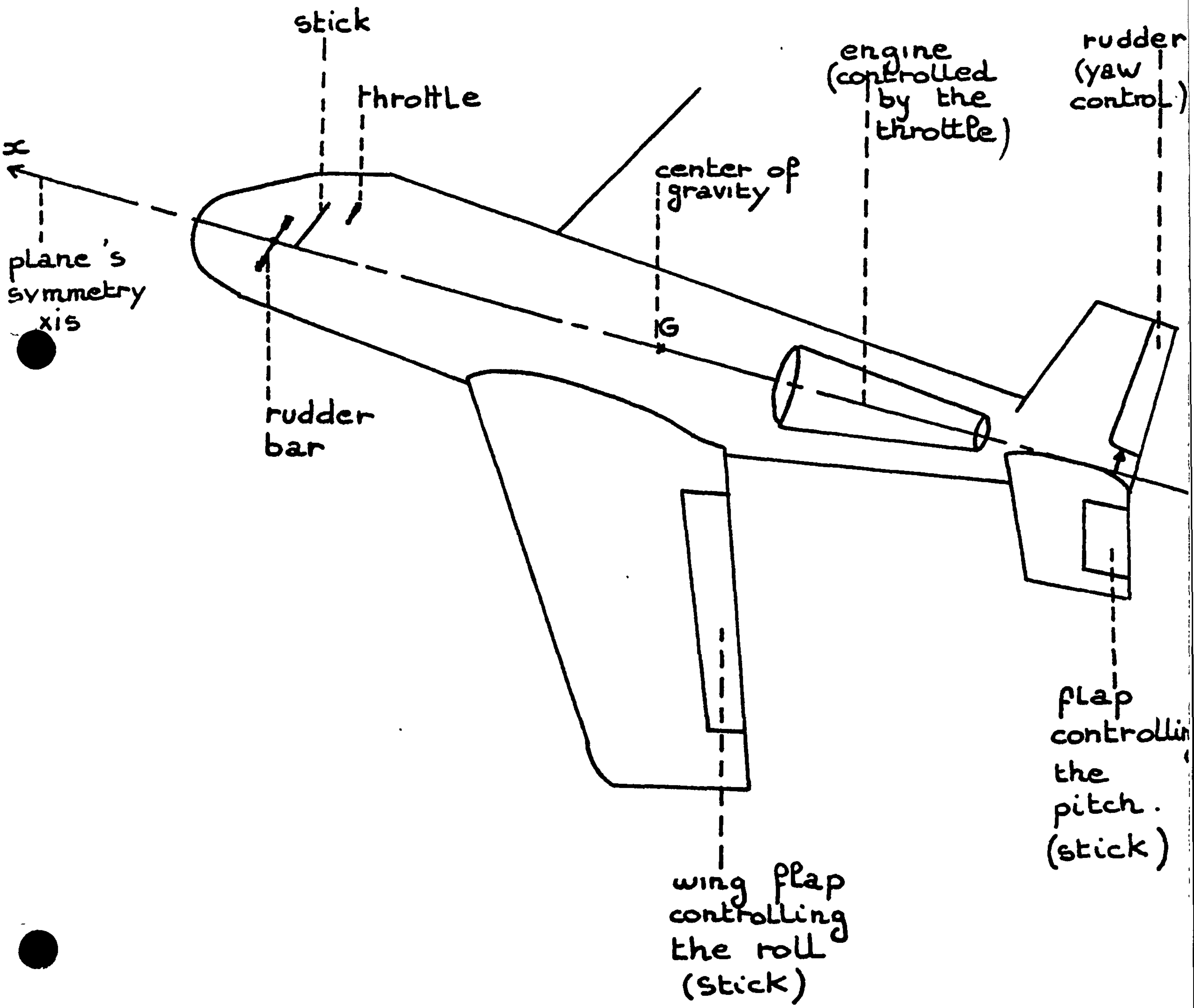
--- The stick, commanding the pitch, attack and roll angles by its action on the wing flaps.

--- The rudder bar, commanding the rudder angle that directly acts on the yaw angle.

These three mechanisms interact very closely, in such a way that it is hard to point out the effect of one of them without looking over the two others. (see Figure 1) figure I :

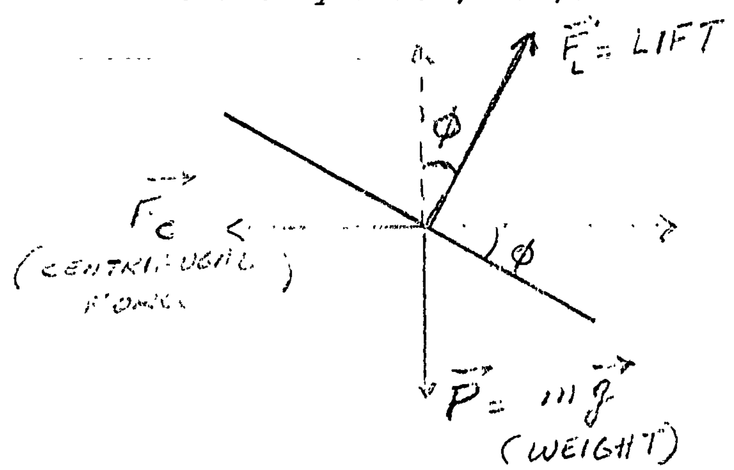
CONTROL UNITS OF A SIMPLE AIRPLANE .

ORIGINAL PAGE IS
OF POOR QUALITY



BASIC EQUATIONS FOR AIRCRAFT DYNAMICS - CONSTANT ALTITUDE:

Our method of obtaining coordination is based on the fact that for a certain bank angle and true air speed, there is only one value of yaw rate ($\dot{\psi}$) for which coordination can be achieved (refer to objective of the project, Aircraft Dynamics, 2.2).



$\dot{C}$(center of the Circle)
$\|\vec{OC}\| = R$

V_T = tangential aircraft velocity

$$F_L \sin\phi = m \frac{V_T{}^2}{R} = m\,\dot{\psi}^2 R = m\,V_T\,\dot{\psi}$$

$$F_L \cos\phi = m\,g \quad \Rightarrow \quad tg\,\phi = \frac{V_T}{g}\dot{\psi} \quad \Rightarrow$$

$$\boxed{\dot{\psi} = \frac{g}{V_T}\, tg\phi}$$

Approximation used:

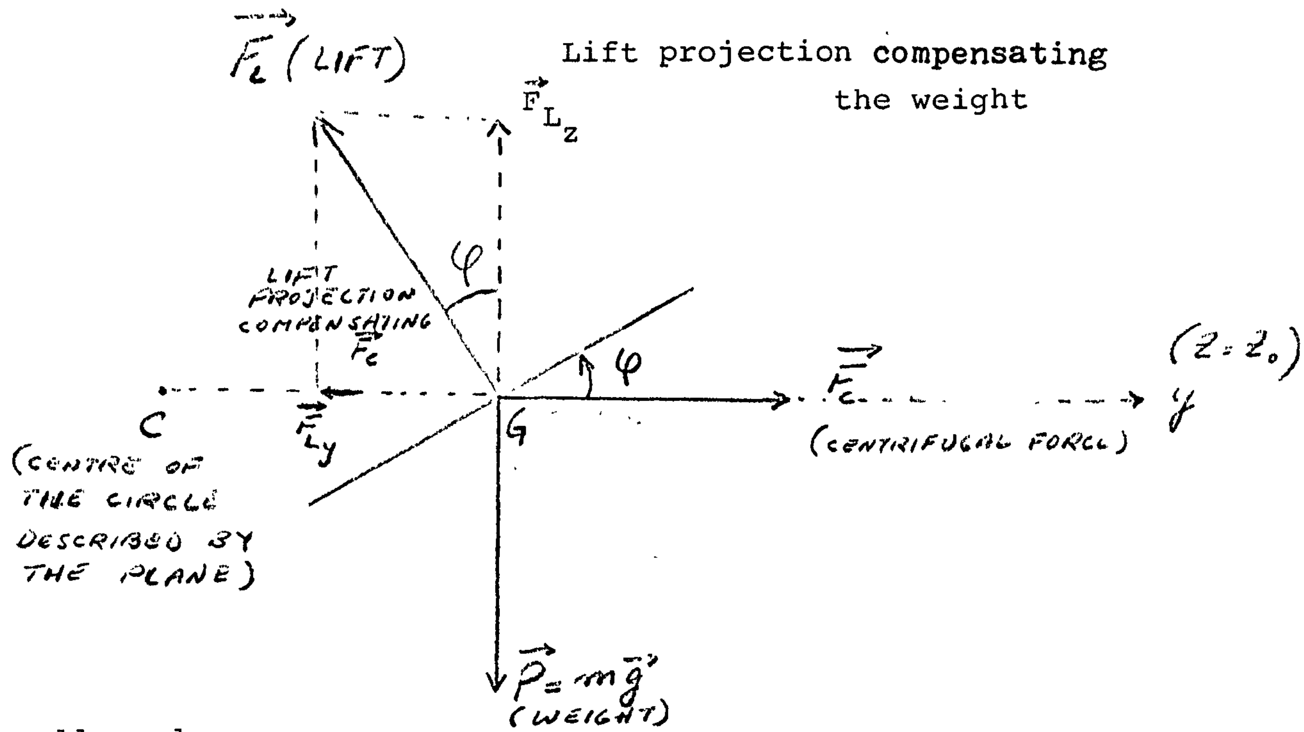
for $\phi < 20^0$ $\quad tg\,\phi \simeq \phi \rightarrow$

$\dot{\psi} \cong \frac{g}{V_T}\,\phi$

STUDY OF TURNING:

ORIGINAL PAGE IS OF POOR QUALITY

When a pilot wants his plane to turn, at a constant altitude, he must roll his aircraft. In order to cancel the effect of gravity the centrifugal force should be as shown below:



φ = roll angle

$F_c = m\, w^2\, r$ with

- m = mass of the aircraft
- w = rotation speed
- r = ray of the circle (or part of circle described by the plane)

BASIC EQUATIONS:

$F_{Ly} = F_c \Rightarrow \underline{F_L \sin\varphi = m\, w^2\, r}$

$\underline{F_L = \frac{1}{2}\rho\, S\, V^2 C_z}$, with:

- ρ = volumic mass of air at $Z=Z_o$
- S = lifting area of the plane
- V = speed of the plane
- Cz= aerodynamic coefficient of lift

$F_{Lz} = mg \Rightarrow \underline{F_L \cos\varphi = mg}$

$\Rightarrow F_L = m\,\frac{g}{\cos\varphi} \Rightarrow n_z = \frac{1}{\cos\varphi}$

A PRIORI LIMITATIONS TO THE SIMULATION MODEL:

Within the aerodynamic equations we can see that for a given roll angle, the turning rate is entirely defined. That is why our model will only consider the roll rate as a variable and our pilot will only use the stick to control his airplane. Note that this is true only because we assume that the host aircraft and target aircraft are co-altitude.

We also have to be aware of the priori limitations existing on a turning manuever. The human body cannot stand more than a limited acceleration, measured in number of G's ($1G = 9.81\ m/s^2$). This number is represented in the turning equations by n_z sometimes called the charge factor. In a turning configuration we showed that n_z was equal to $1/\cos\psi$; the following table gives the values of turning angles for different charge number, measured in units of G.

Value of n_z (G's)	Value of the turning angle *
8	84 degrees
7	83 degrees
6	80.5 degrees
5	79.5 degrees
4	77 degrees
3	75.5 degrees

* Note that these angles are lowered by Aerodynamic considerations not taken into account here.

Summary of the OBS parametres.

type	designation	symbol used	other usual symbol
Input	Lateral stick position	δ_{st}	
Data	True airplane velocity	V	TAS
Output	Roll rate Roll angle Turning rate Turning angle	$\dot{\phi}$ ϕ $\dot{\psi}$ ψ	P R

Important Assumptions

$Z = Z_o$: we stay at a constant altitude

α = constant $\Rightarrow \dfrac{\partial C_z}{\partial \alpha} = 0 \Rightarrow C_z$ = constant

$\beta = 0$: we turn without sliding.

BUILDING THE CONTROL LOOP:

The pilot acts on the stick by giving it an angular displacement δ_{ST}, that drives the wings ailerons to an angular displacement δ_a that creates a banking moment. This moment is proportional to the rolling acceleration $\ddot{\phi}$, and the diagram leading to $\dot{\phi}$ is the following:

$$\boxed{\text{PILOT}} \xrightarrow{\delta_{ST}} \boxed{K_1} \xrightarrow{\delta_a} \boxed{K_2} \xrightarrow{\ddot{\phi}} \boxed{\frac{1}{S}} \xrightarrow{\dot{\phi}}$$

When we have $\dot{\phi}$, we can use the formula: $\dot{\psi} = \frac{g}{V_T} \operatorname{tg} \phi$, to calculate $\dot{\psi}$. For this, we need first to integrate $\dot{\phi}$ then we assume that $\operatorname{tg}\phi \approx \phi$, which is true for the small angles, and our formula becomes: $\dot{\psi} = \frac{g}{V_T} \phi$. The diagram between $\dot{\phi}$ and $\dot{\psi}$ is:

$$\xrightarrow{\dot{\phi}} \boxed{\frac{1}{S}} \xrightarrow{\phi} \boxed{\frac{g}{V_T}} \xrightarrow{\dot{\psi}} \boxed{\frac{1}{S}} \xrightarrow{\psi}$$

to feedback loop

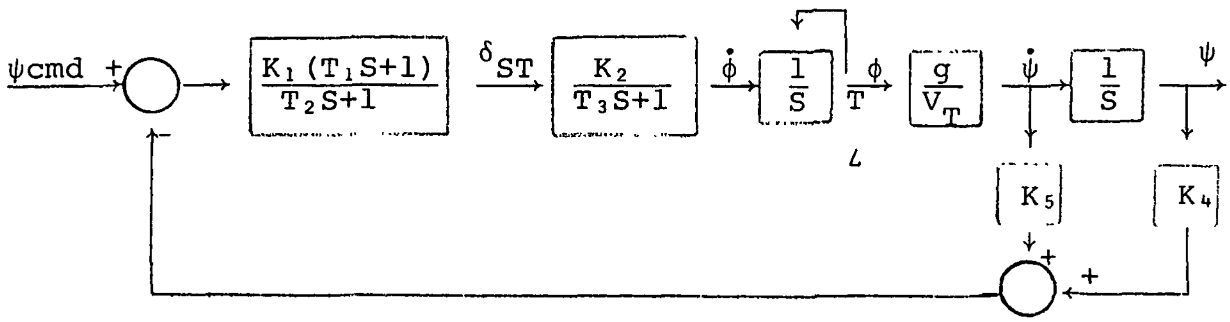
These diagrams do not take into account the delays that can occur in the commands. We tried to figure it by using the following control diagram:

$$\text{PILOT} \xrightarrow{\delta_{ST}} \boxed{\frac{K_2}{T_3S + 1}} \xrightarrow{\dot{\phi}}$$

with the pilot being modelized as:

$$\xrightarrow{E} \boxed{\frac{K_1\ (T_1S + 1)}{T_2S + 1}} \longrightarrow \delta_{ST}$$

FINAL 3-DOF SIMULATION CONFIGURATION:



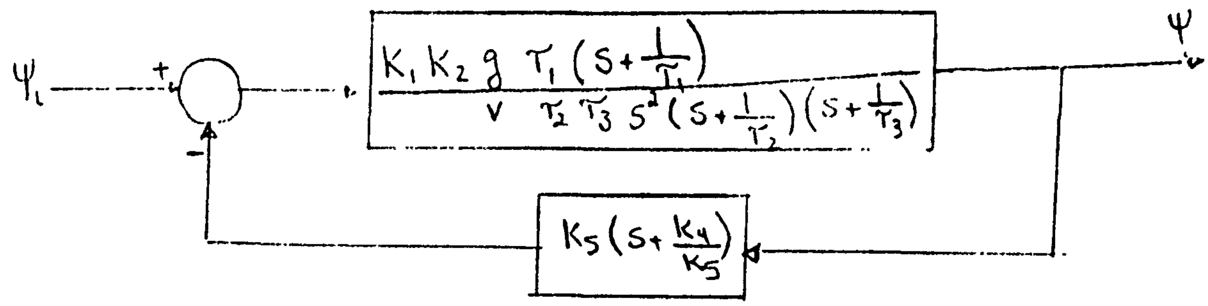
This is the final control loop description for the three degree-of-freedom simulation in PASCAL. It has an added rate damping path with gain K_5 and is shown to be stable in the following root locus analysis. The first order lag acting on the stick command models the time delay in the dynamic reaction of the aircraft to the commanded rolling moment.

The lag was previously placed with the aerodynamics but concern has arisen over the aircraft roll angle during its maneuvers. For accurate modeling and evaluation of possible limiting it was determined that the lag should delay the roll rate and not the turning rate.

In evaluating the roll limiter it is noted that without any limiter on the roll angle it grows to 4.2 radians at time equals 1.5 seconds. This is clearly unrealizeable so a shunt limiter was placed on roll angle limit. Attached plots of heading, roll angle, and roll rate for both unlimited and limited cases clearly depict this situation and solution.

Root Locus Hand Calculation

Starting with the system:



In order to determine a suitable gain $(K_1 \cdot K_2)$ a root locus analysis was performed on the open loop transfer function KGH:

$$KGH = \frac{K(S + 2.5)(S + .5)}{S^2 \ (S + 4)(S + 2)}$$

Where

$$\tau_1 = .4$$
$$\tau_2 = .25$$
$$\tau_3 = .5$$
$$K_4/K_5 = .5$$
$$K_4 = 1$$
$$V = 200 \text{ m/s}$$
$$g = 10 \text{ m/s}^2$$

Two methods were used to determine the optimal gain, the first is a hand calculation and the second corraborating analysis was performed using an interactive computer aided design tool called TOTAL which derives from the USAF Eglin base.

$$KGH = \frac{K \ (S + 2.5)(S + .5)}{S^2 \ (S + 4)(S + 2)}$$

1. Find the root locus asymptates.

$$\overline{OA} = \frac{4 + 2 - (2.5 + .5)}{2} = \frac{6 - 3}{2} = \frac{3}{2}$$

2. Find the root locus "real" value as it crosses the line +j.

The phase angle criterion yields: for (-1+j)

$$= [90° + \tan^{-1}(\frac{.5}{1})] + [\tan^{-1}\frac{1}{1.5}] - 2\,[135°] - [45°] - [\tan^{-1}\frac{1}{3}]$$

$$= \quad +116.6 \quad + \quad 33.7 \quad - \quad 270 \quad - \quad 45 \quad - \quad 18.4$$

$$= \quad -183.1°$$

3. Use the magnitude condition to find K at (1±j)

$$1 = |KGH| \quad \Rightarrow \quad K = \left.\frac{1}{|GH|}\right|_{s=-1+j}$$

$$K = \frac{(\sqrt{1+1}\,)^2 \sqrt{3^2+1^2}\,\sqrt{2}}{\sqrt{1+(1.5)^2}\,\sqrt{1+.5^2}}$$

$$= \frac{2\sqrt{2}\,\sqrt{10}}{\sqrt{3.25}\,\sqrt{1.25}}$$

$$= 4.44$$

4. To find the roots that correspond to K = 4.44

Choose a value on the real axis and compute its K value.

Say S = -3.5

$$K = \frac{(-3.5)^2\,(-3.5+4)(-3.5+2)}{(-3.5+2.5)(-3.5+.5)} = \frac{12.25\;(.5)(+1.5)}{(1)\quad(3)} = 3.06$$

This is too small an S value

Say S = -3.0

$$K = \frac{(-3.0)^2\ (-3.0 + 4.0)(-3.0 + 2.0)}{(-3.0 + 2.5)\ (-3.0 + .5)} = \frac{9 \cdot 1 \cdot 1}{(.5)(2.5)} = 7.2$$

This is too large an S value

Say S = -3.35

$$K = \frac{(3.35)^2\ (+3.35 - 4.0)(3.35 - 2.0)}{(3.35 - 2.5)\ (3.35 - .5)} = \frac{11.22\ (.65)(1.35)}{(.85)\ (2.85)} = 4.06$$

$$\boxed{S = -3.3}$$

$$K = \frac{(3.3)^2\ (-3.3 + 4.0)\ (3.3 - 2.0)}{(3.3 - 2.5)\ (3.3 - .5)} = \frac{10.89\ (.7)\ (1.3)}{(.8)\ (2.8)} = 4.42$$

Trying a similar point for the other pole zero pair which moves an equal distance:

S = -1.26

$$K = \frac{(1.26)^2\ (-1.26 + 4.0)\ (-1.26 + 2.0)}{(-1.26 + 2.5)\ (1.26 - .5)} = \frac{1.59\ (2.74)\ (.74)}{(1.24)\ (.76)} = 3.42$$

S = -1.0

$$K = \frac{1\quad 3\quad 1}{(1.5)\ (.5)} = 4.0$$

$$\boxed{S = -.9}$$

$$\frac{.9^2\ (3.1)\ (1.1)}{(1.6)\ (.4)} = 4.3$$

S = -.78

$$\frac{.78^2\ (-.78 - 4.0)\ (-.78 + 2.0)}{(-.78 + 2.5)\ (-.78 + .5)} = \frac{.608\ (3.22)\ (1.22)}{(1.72)\ (.28)} = 4.96$$

Since $K_G = \frac{K_1 K_2 g}{V\ \tau_3 \tau_2}$

and $K_H = K_5$

If poles are at -3.3, -.9, (1±j) K = 4.44

What is the gain for design?

$$K\ \ 4.44 = \frac{\tau_1}{\tau_2 \tau_3}\ \frac{K_5 K_1 K_2\ g}{V} \qquad \tau_1 = .4$$

$$= \frac{.4}{.25\ \ .5}\ \frac{\cancel{2}\ K_1 K_2}{\cancel{20}\ 10}$$

$$= .04 \cdot 8\ K_1 K_2 = .32\ K_1 K_2$$

$$\frac{4.44}{.32} = K_1 K_2$$

$$\underline{\underline{13.875 = K_1 K_2}}$$

Checking by plugging in values:

$$G = \frac{\overset{\overset{K}{\wedge}}{K_1 K_2 g}}{V \quad S^2} \quad \frac{(\tau_1 \; S + 1)}{(\tau_2 \; S + 1) \; (\tau_3 \quad S + 1)} = \frac{K \; \tau_1 (S + \frac{1}{\tau_1})}{\tau_3 \tau_2 (S + \frac{1}{\tau_2}) \; (S + \frac{1}{\tau_3})}$$

$$H = K_5 \; (S + \frac{K_4}{K_5})$$

$$\frac{4(s)}{4_c(s)} = \frac{G}{1 + GH} = \frac{K \; \tau_1 (S + \frac{1}{\tau_1})}{\tau_3 \tau_2 S^2 (S + \frac{1}{\tau_1}) (S + \frac{1}{\tau_2}) + K \; \tau_1 (S + \frac{1}{\tau_1}) \; K_5 \; (S + \frac{K_4}{K_5})}$$

$$= \frac{K \quad \tau_1 \quad (S + \frac{1}{\tau_1})}{\tau_3 \tau_2 \; S^2 (S^2 + \frac{1}{\tau_1} + \frac{1}{\tau_2}) \; S + \frac{1}{\tau_1 \tau_2}) + \underbrace{K \; K_5 \; \tau_1}_{X} (S^2 \; + (\frac{1}{\tau_1} + \frac{K_4}{K_5}) \; S + \frac{K_4}{\tau_1 K^5})}$$

$$= \tau_3 \tau_2 \; S^4 + \tau_3 \tau_2 (\frac{1}{\tau_1} + \frac{1}{\tau_2}) \; S^3 + \frac{\tau_3}{\tau_1} S^2 + \underbrace{K \; K_5 \; \tau_1}_{X} \; S^2 + X \; (\frac{1}{\tau_1} \; + \frac{K_4}{K_5}) \; X + \frac{X \; K_4}{\tau_1 K_5}$$

$$\Rightarrow K = \frac{13.9 \; X \; 10}{200} = .695$$

$$X = K \bullet K_5 \; \tau_1 = .695 \times (2) \times .4 = .556$$

$$= \frac{\frac{K \; \tau_1}{\tau_3 \; \tau_2} \; (S + \frac{1}{\tau_1})}{S^4 + (\frac{1}{\tau_1} + \frac{1}{\tau_2}) \; S^3 + (\frac{\tau_3}{\tau_1} + K \; K_5 \; \tau_1) \; S^2 \; + K \; K_5 \; \tau_1 \; (\frac{1}{\tau_1} + \frac{K_4}{\tau_5}) \; S + K \; K_5 \; \tau_1 (\frac{K_4}{\tau_1 K_5})}$$

$$= \frac{\frac{(.695)(.4)}{(.5) \; (.25)} \quad S + \frac{1}{.4}}{S^4 + (\frac{1}{.5} + \frac{1}{.25}) \; S^3 + \frac{\frac{.5}{.4} + .556}{(.5)(.25)} \; S^2 + \frac{.556(\frac{1}{.4} + .5)}{(.5) \; (.25)} \; S + \frac{.556 \; (\frac{.5}{.4})}{(.5) \; (.25)}}$$

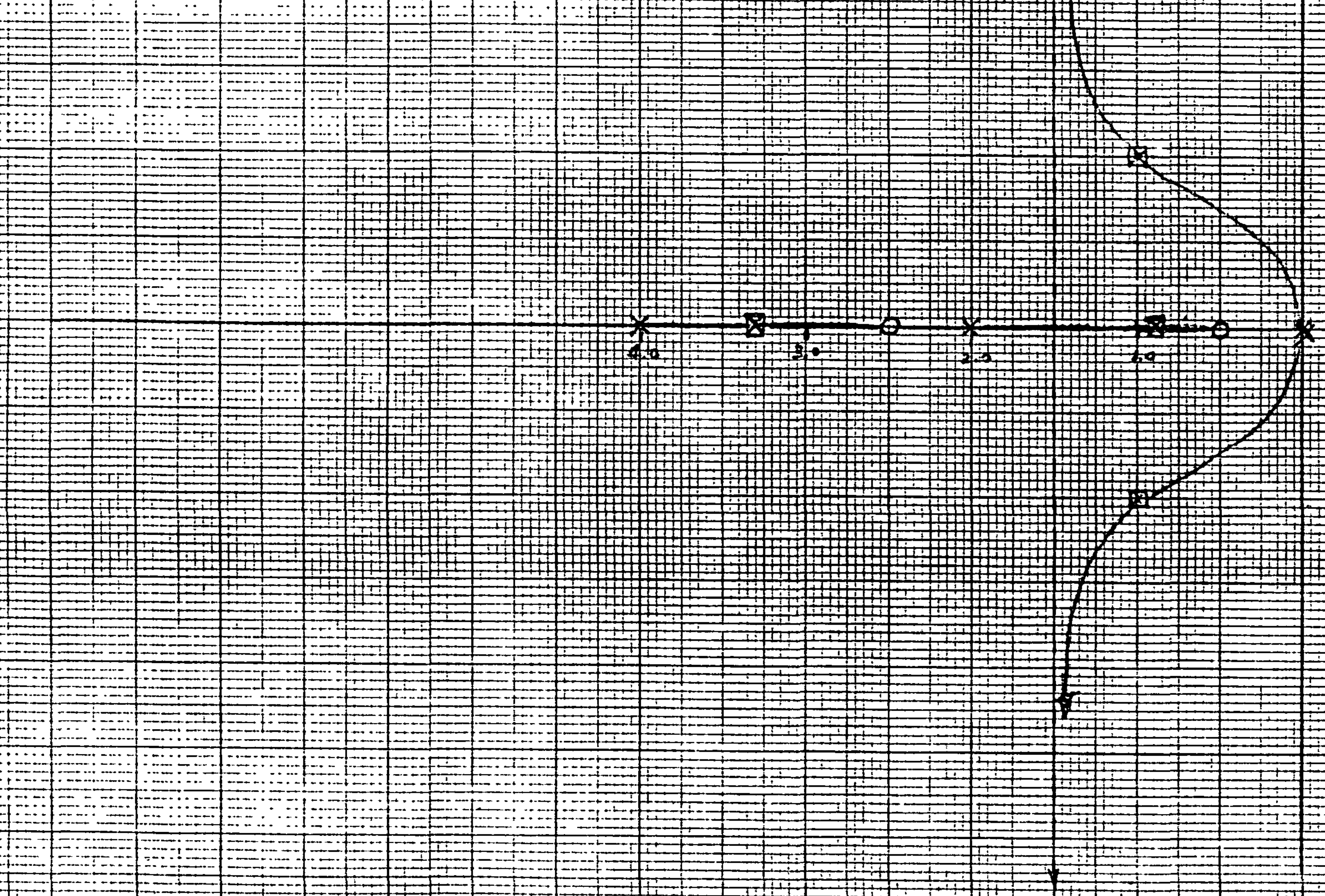
$$= \frac{2.22 \; (S + \frac{1}{.4})}{S^4 + 6 \; S^3 + 14.4 \; S^2 + 13.3 \; S + 5.56}$$

This is similar to the computer's TOTAL answers.

$$KGH = \frac{K(S+2.5)(S+.5)}{S^2(S+4)(S+2)}$$

$\tau_1 = .4$
$\tau_2 = .25$
$\tau_3 = .5$
$K_4/K_5 = .5$

Root Locus



10/28/82 J.B.

ORIGINAL PAGE IS
OF POOR QUALITY

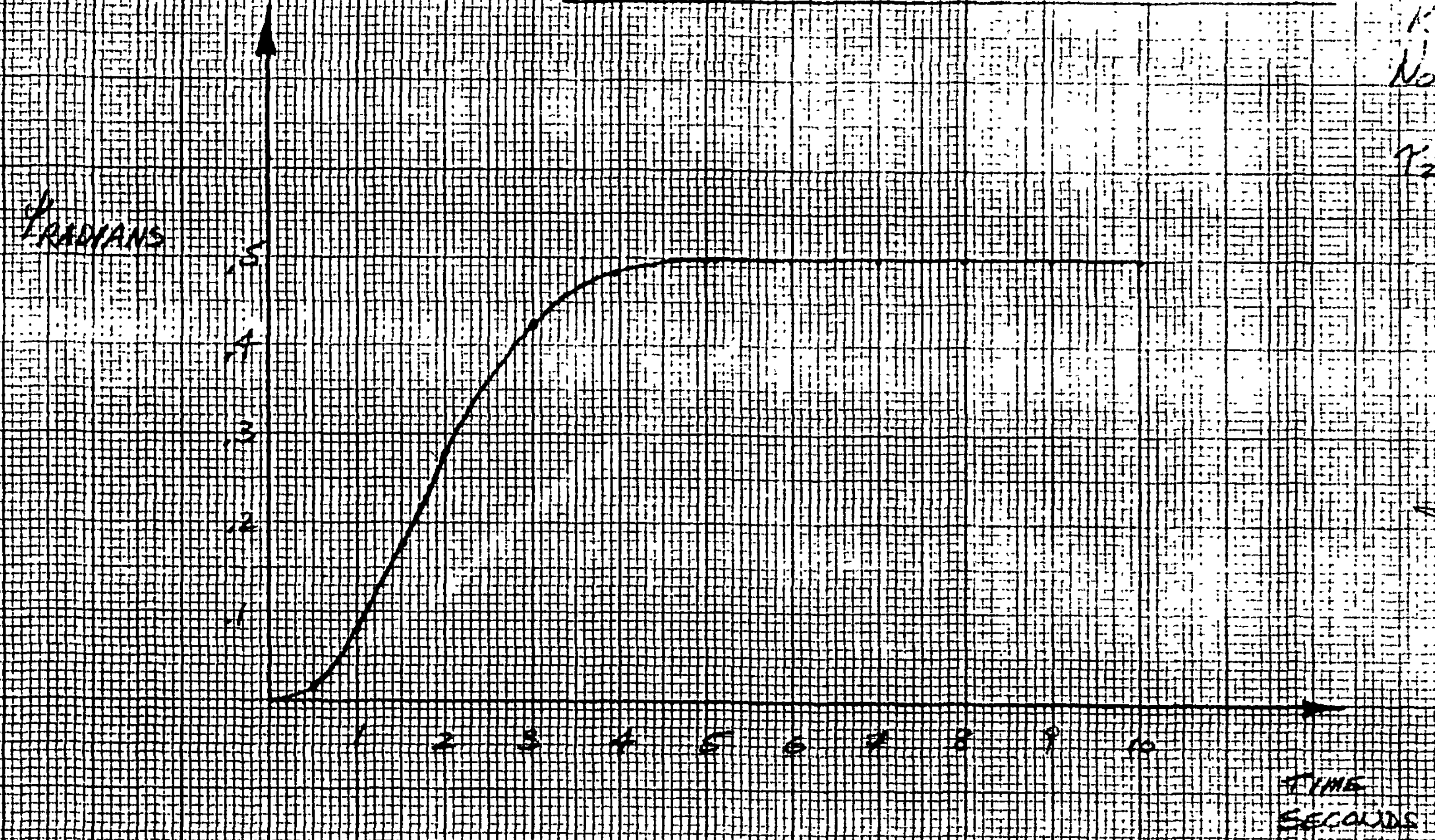
3 DOF SIMULATION

TIME RESPONSE TO A .5 STICK INPUT

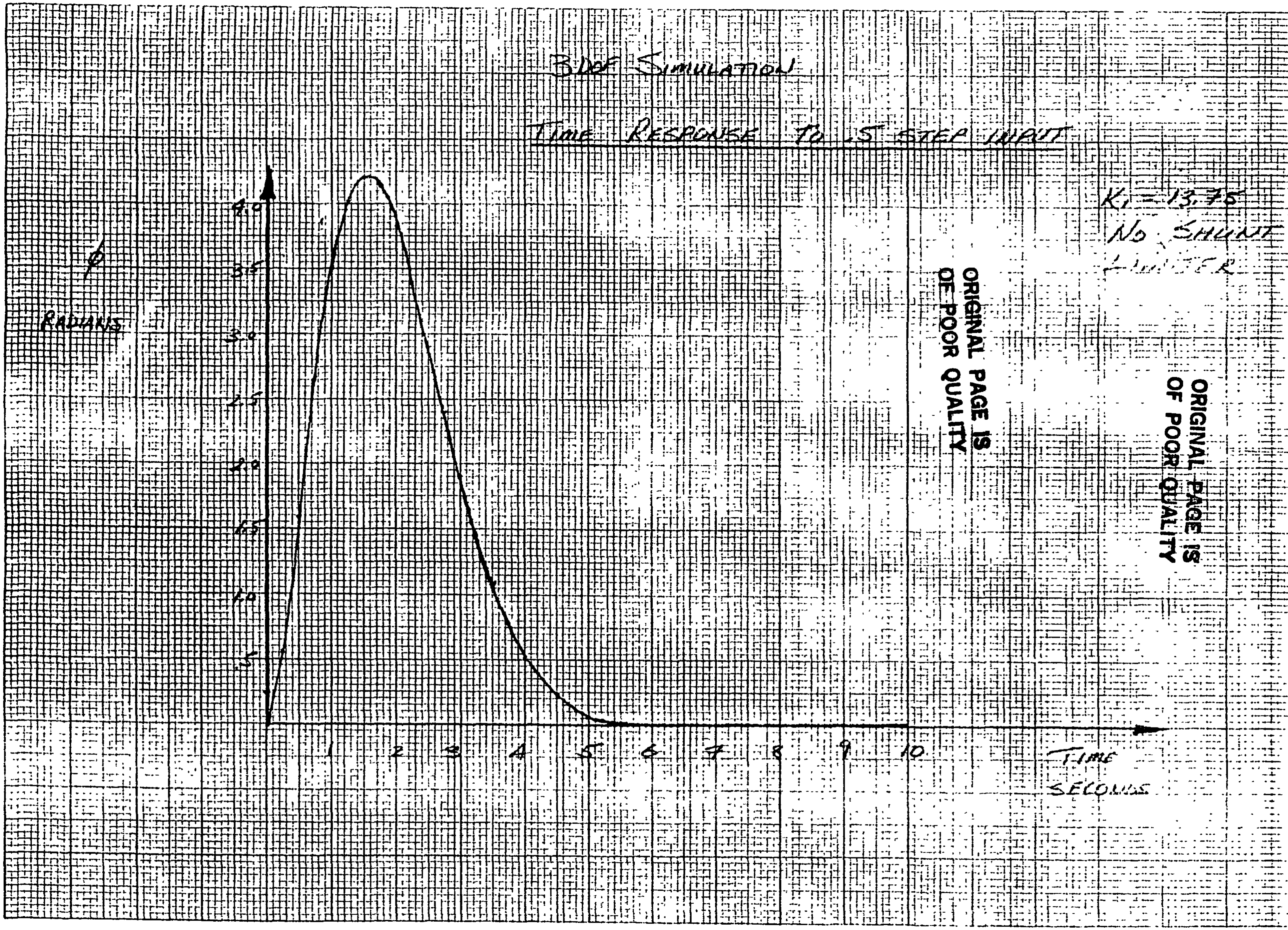
K = 13.75

NO SHUNT LIMITER

Y_2 ON $\int \delta_{ST}$



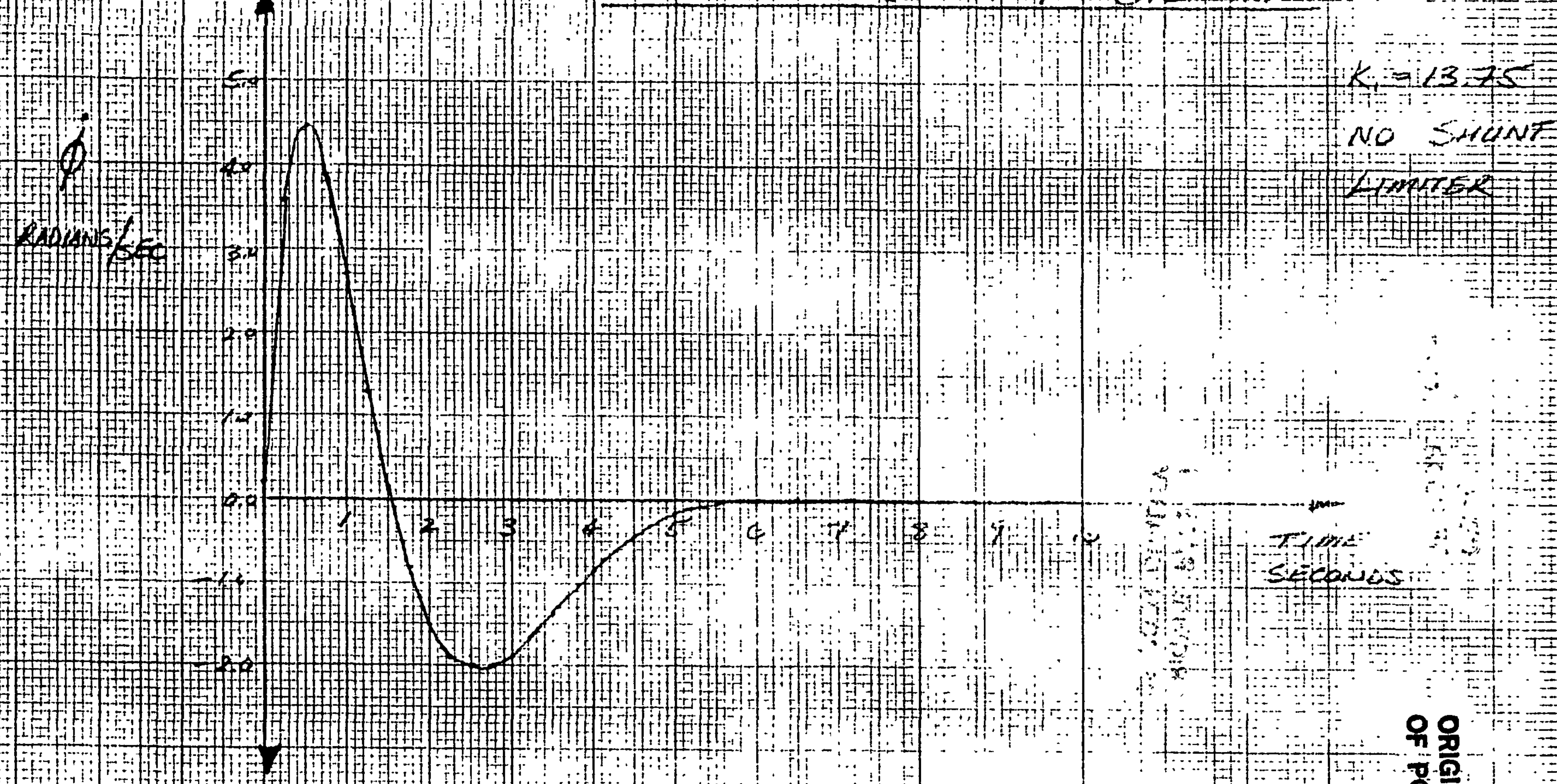
ORIGINAL PAGE IS
OF POOR QUALITY



ORIGINAL PAGE IS
OF POOR QUALITY

3DOF SIMULATION

TIME RESPONSE TO .5 STEP INPUT



K•E 10 X 10 TO ½ INCH 7 X 10 INCHES
KEUFFEL & ESSER CO MADE IN U.S.A. 46 1323

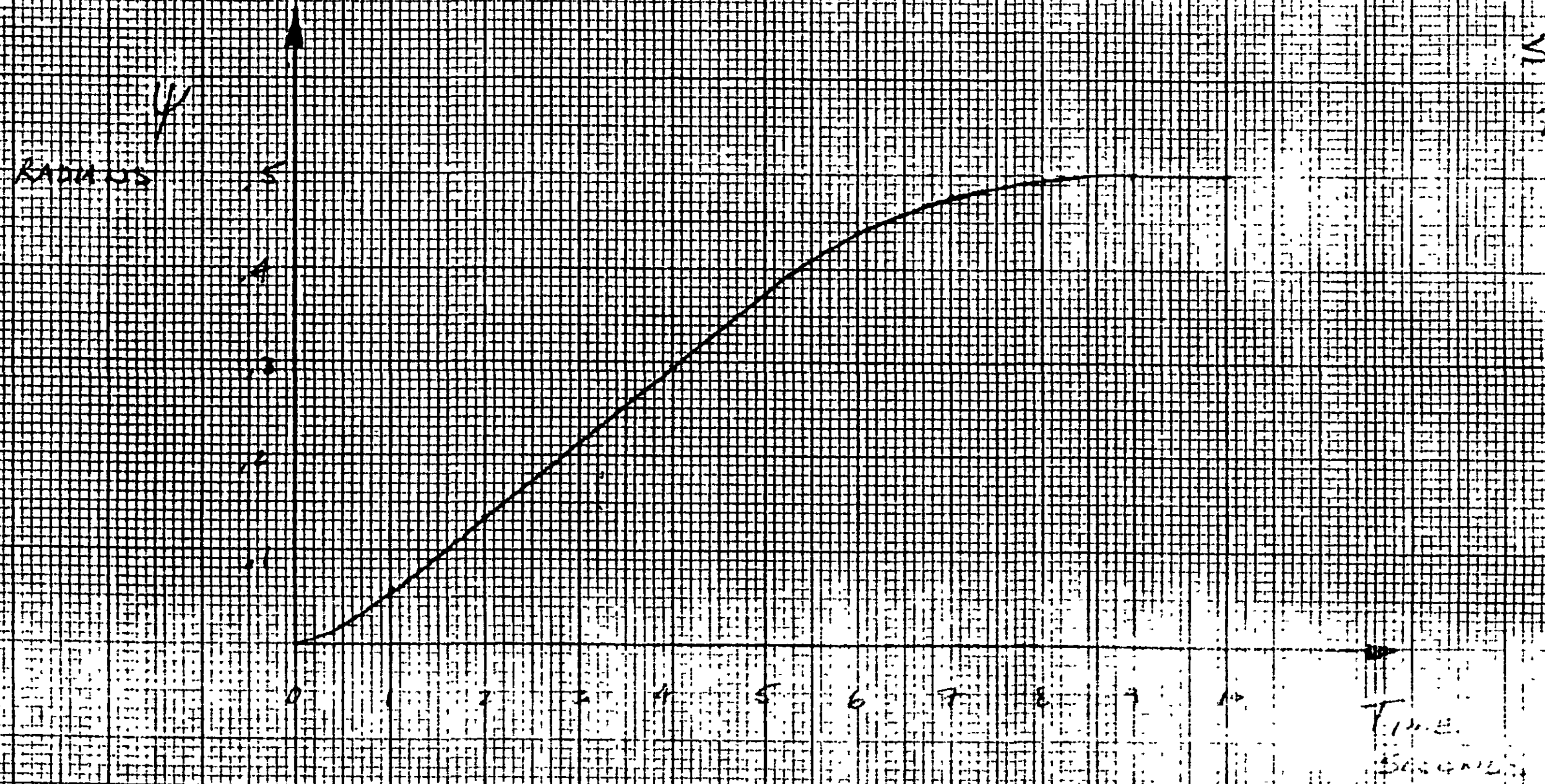
3DOF SIMULATION

TIME RESPONSE TO A .5 STEP INPUT

$K_1 = 13.75$

SHUNT LIMITER

T_3 ON δ_{ST}



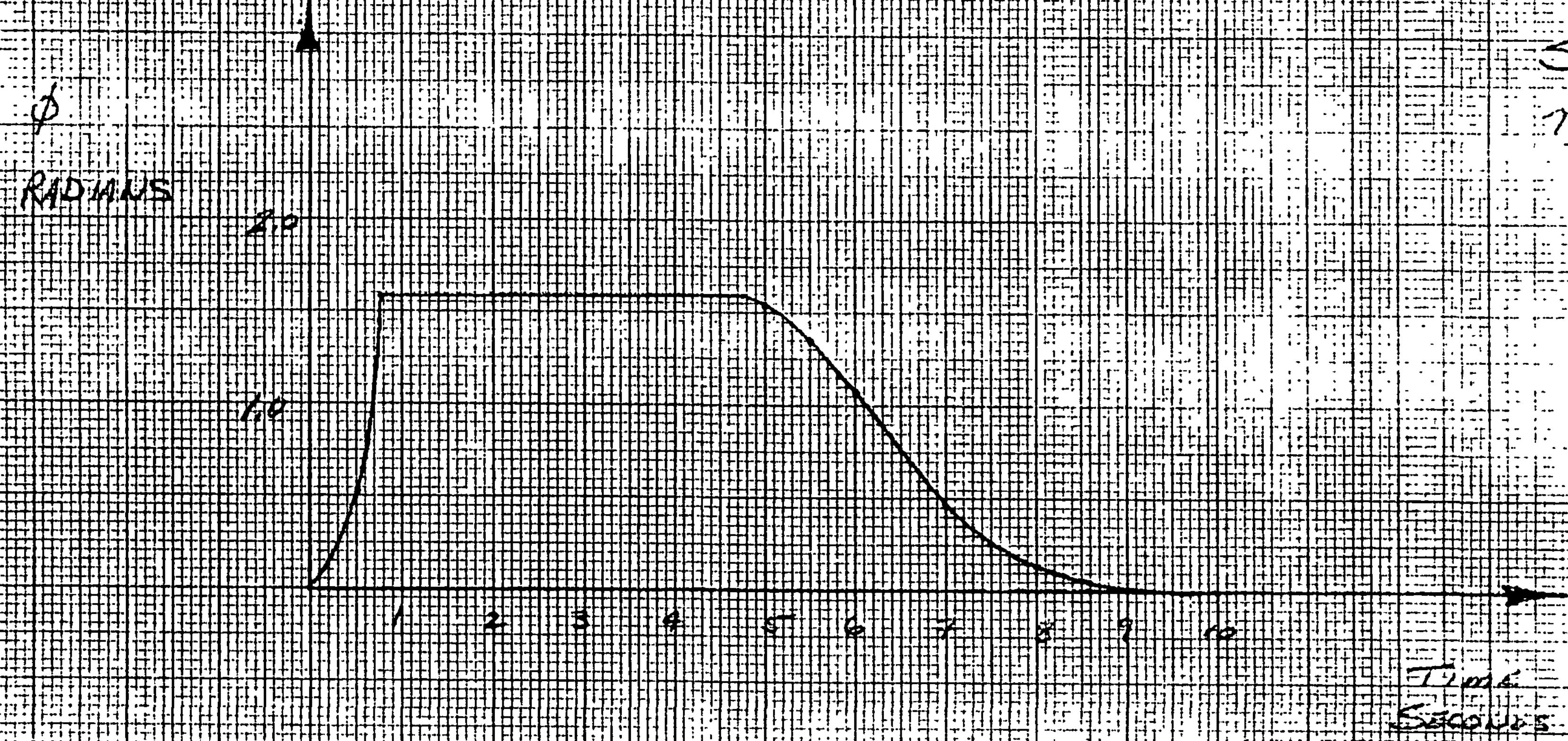
3DOF SIMULATION

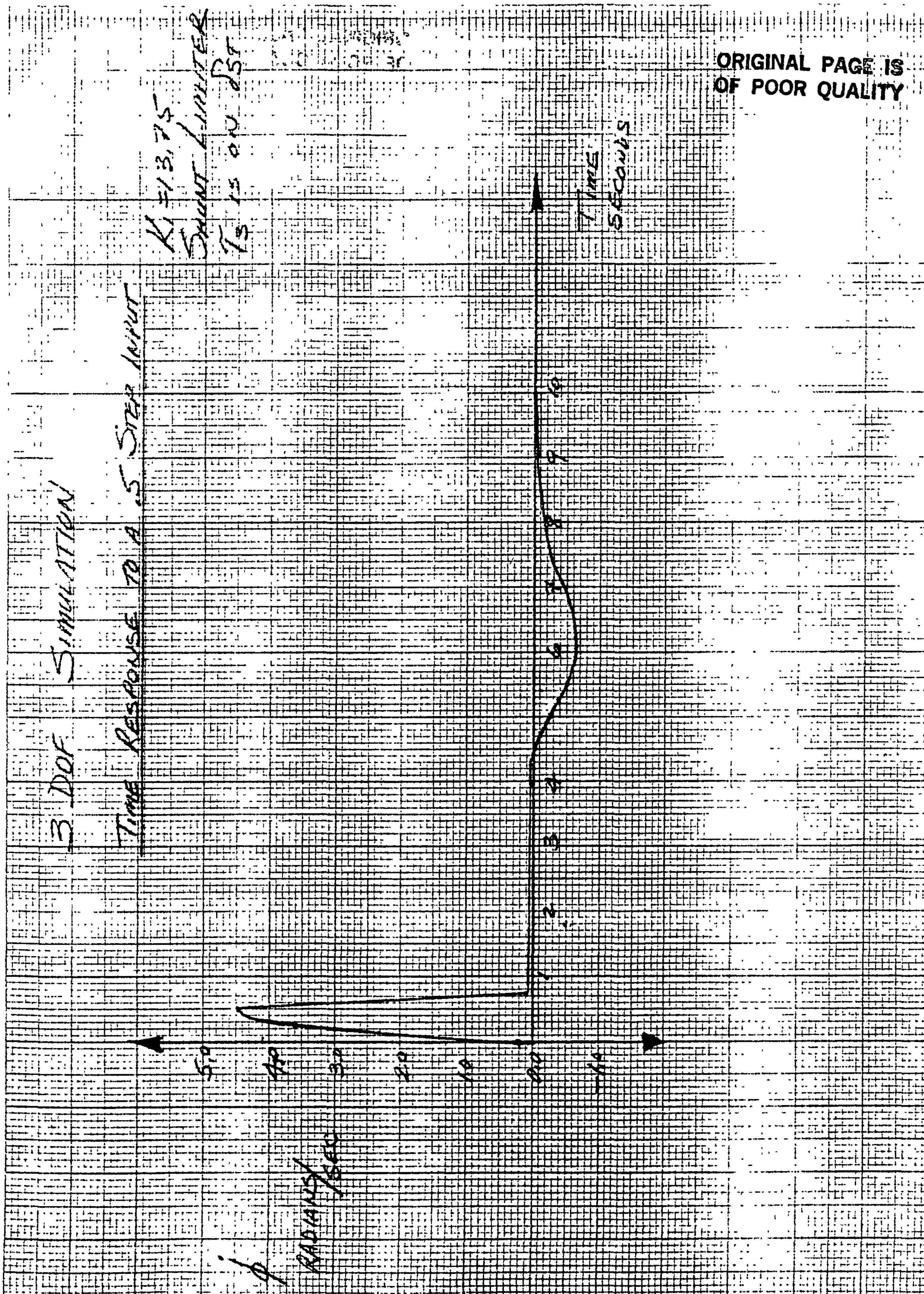
TIME RESPONSE TO A .5 STEP INPUT

$K_1 = 13.75$

SHUNT LIMITER

τ_3 ON δ_{ST}




3 DOF SIMULATION
TIME RESPONSE TO A .5° STEP INPUT
K1 = 13.75
SHUNT LIMITER
IS ON δST
TIME
SECONDS
φ̇
RADIANS/SEC
5.0
4.0
3.0
2.0
1.0
0.0
-1.0
1
2
3
4
6
7
8
9
10

ORIGINAL PAGE IS
OF POOR QUALITY

ANALYSIS

The user package called TOTAL is an easy way to analyze control loops. One enters in G(s), then H(s) and from there performs requested options interactively.

```
NET 172040
PLEASE SIGN ON  KWE,L0507LA
 02/10/20. 08.54.35.
WESTERN CYBERNET CENTER  SN105 CY 176   NOS 1.3H/477.769
PASSWORD
[redacted]
TERMINAL:     122, TTY
PLEASE CHANGE YOUR CDC SUPPLIED PASSWORD.
RECOVER /SYSTEM: BATCH
$RFL,0.
/

 NOTICE  PLEASE SEE   EXPLAIN,NEWS

ATTACH,TOTAL/UN=01703LA
/TOTAL
 WELCOME TO TOTAL -- VERSION 3.0
 (C) 1980  --  STANLEY J. LARIMER

 ********  COLIN USERS  ********
 QUESTIONS OR PROBLEMS, CONTACT:
 CAPT DENNIS DIDALEUSKY
 AD/SDES-A (2-5678/2-5669)
```

The general options categories available are:

```
    S AN INTERACTIVE COMPUTER-AIDED DESIGN PROGRAM
    GITAL & CONTINUOUS CONTROL SYSTEM ANALYSIS.
    NTAINS 160 OPTIONS DIVIDED INTO GROUPS OF 10
    RDING TO GENERAL APPLICATION.

  IONS ENDING IN 0 LIST THE NEXT 10 OPTIONS,
  R EXAMPLE, OPTION 30 LISTS OPTIONS 30 THRU 39.

THE FOLLOWING ARE THE MAIN OPTION GROUPS:
    0-9:  TRANSFER FUNCTION INPUT OPTIONS
  10-19:  MATRIX INPUT OPTIONS
  20-29:  BLOCK DIAGRAM MANIPULATION OPTIONS
  30-39:  TIME RESPONSE OPTIONS
  40-49:  ROOT LOCUS OPTIONS
  50-59:  FREQUENCY RESPONSE OPTIONS
  60-69:  POLYNOMIAL OPERATIONS
  70-79:  MATRIX OPERATIONS
  80-89:  DIGITIZATION OPTIONS
  90-99:  OPTIONS OF PARTICULAR INTEREST
100-109:  MORE TRANSFER FUNCTION INPUT OPTIONS
110-119:  MORE MATRIX OPTIONS
120-129:  MORE BLOCK DIAGRAM MANIPULATIONS
130-139:  STATE TRANSITION SIMULATION OPTIONS
140-149:  DOUBLE-PRECISION DISCRETE TRANSFORM OPTIONS
150-159:  MULTI-RATE FREQUENCY RESPONSE OPTIONS
```

ORIGINAL PAGE IS
OF POOR QUALITY

The input open loop transfer function is as follows:

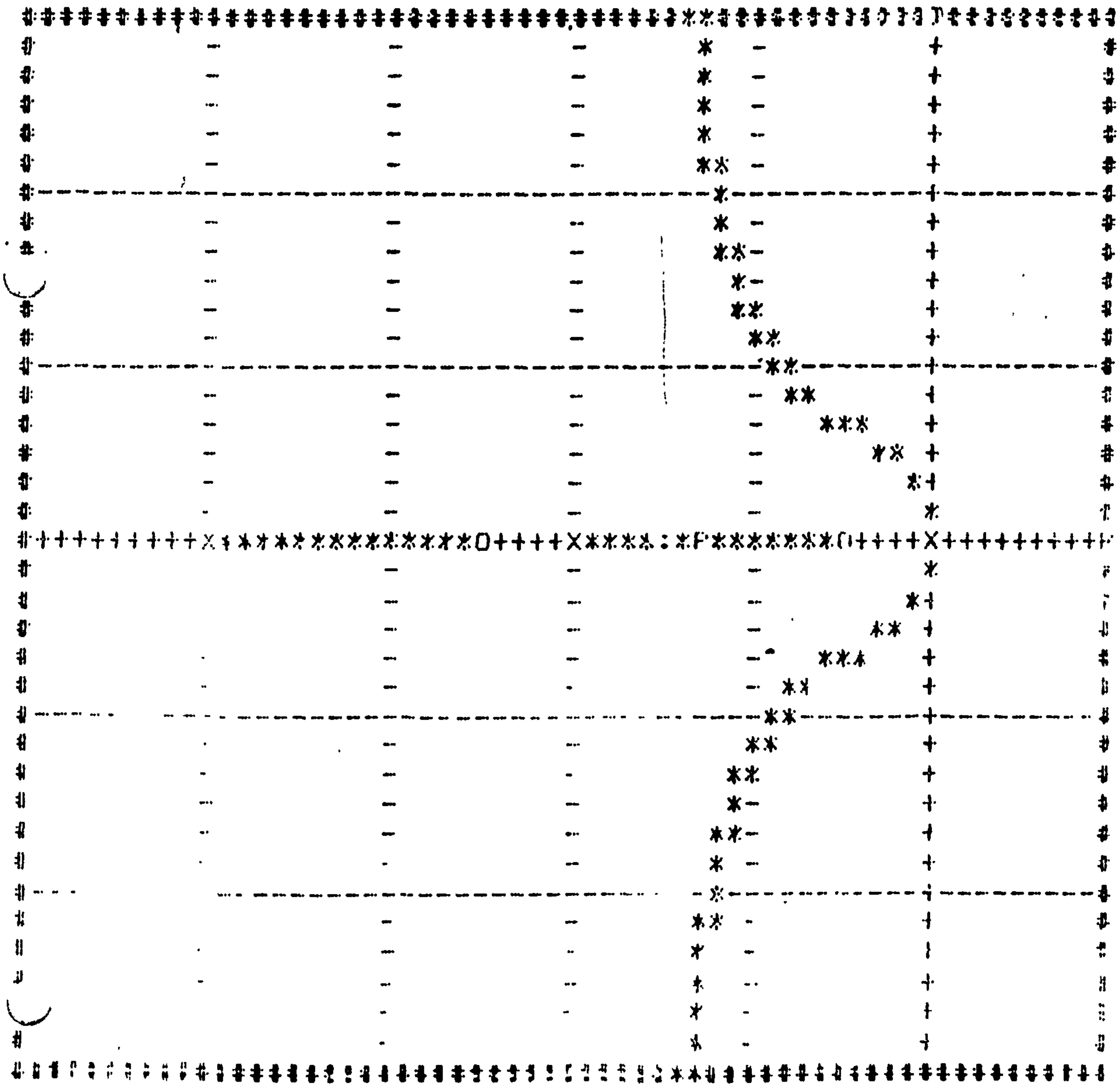
```
                    OLF= GAIN*( OLNK/OLDK )=    2.000
                         GAIN=    1.000

                           OLTF(S) NUMERATOR
 I            OLNPOLY(I)                              OLZERO(I)
 1      (     2.000       )S** 2        (  -.5000       ) + J(    0.          )
 2      (     6.000       )S** 1        (  -2.500       ) + J(    0.          )
 3      (     2.500       )                             OLNK=      2.000

                           OLTF(S) DENOMINATOR
 I            OLDPOLY(I)                              OLPOLE(I)
 1      (     1.000       )S** 4        (  0.           ) + J(    0.          )
 2      (     6.000       )S** 3        (  0.           ) + J(    0.          )
 3      (     8.000       )S** 2        (  -2.000       ) + J(    0.          )
 4      (   0.            )S** 1        (  -4.000       ) + J(    0.          )
 5      (   0.            )                             OLDK=      1.000
```

and the root locus options are:

```
 OPTION ?
? 40

 (40-49  ROOT LOCUS OPTIONS
          (SET TSAMP=0 FOR S-PLANE, TSAMP=SAMPLE TIME FOR Z-PLANE)
 * 40     LIST OPTIONS
 * 41     GENERAL ROOT LOCUS
 * 42     ROOT LOCUS WITH A GAIN OF INTEREST
 * 43     ROOT LOCUS WITH ZETA (DAMPING RATIO) OF INTEREST
 * 44     LIST N POINTS ON A BRANCH OF INTEREST
 * 45     LIST ALL POINTS ON A BRANCH OF INTEREST
 * 46     LIST LOCUS ROOTS AT A GAIN OF INTEREST
 * 47     LIST LOCUS ROOTS AT A ZETA OF INTEREST
 * 48     PLOT ROOT LOCUS AT USER'S TERMINAL
 * 49     LIST CURRENT VALUES OF ALL ROOT LOCUS VARIABLES

 *     TYPE: HELP,49 FOR DEFINITION OF ROOT LOCUS VARIABLES

 *     A CALCOMP PLOT FOR OPTIONS 41,42,43,& 48 MAY BE
       OBTAINED BY TYPING:  PLOT,41  OR  PLOT,42  ETC.

 OPTION ?
? 41
```

OPEN-LOOP (OLTF) ROOT LOCUS USING OPTION 4


```
  4 POLES AT
   X =   0.                        Y =  0.
   X =   0.                        Y =  0.
   X =  -2.0000                    Y =  0.
   X =  -4.0000                    Y =  0.

  2 ZEROS AT
   X =  -.50000                    Y =  0.
   X =  -2.5000                    Y =  0.

  GAIN CONSTANT (OLNK/OLDK)=     2.0000000

 REGION OF CALCULATION-REAL:  CC=  -5.00     TO AA=   1.00
                        IMAJ:  DD=  -3.00     TO BB=   3.00
```



```
 GRID SCALE:  X-AXIS:  1 INCH=      1.0000
              Y-AXIS:  1 INCH=      1.0000
```

The four branches plotted in the root locus are tabulated enabling choosing a desired gain or damping ratio.

REGION OF CALCULATION REAL: CC = 0.00 TO AA = 1.00
IMAJ: DD = 3.00 TO BB = 3.00

```
BRANCH STARTING AT (0.) + J(0.)
TYPE L TO LIST, S TO SKIP, OR $ TO ABORT >
L
```

ORIGINAL PAGE IS OF POOR QUALITY

BRANCH NUMBER 1

CALCULATION STEP SIZE = .1000
PRINTING STEP SIZE = .1500

LOCUS REAL	LOCUS IMAG	DIST TO ORIGIN	GAIN	ZETA	CD
0.	0.	.10000000E-02	0.	1.00000	1
[illegible]0047E-01	-.19664590	.19933076	[illegible]314	.16350	0
.1[illegible]328	-.37332331	.39372087	[illegible]3487	.31770	0
[illegible]082	-.51723892	.58042675	[illegible]1806	.45377	0
.4[illegible]7784	-.62847910	.76115130	1.[illegible]	.56412	[illegible]
.60[illegible]9384	-.72527393	.94409433	1.63619	.64018	0
[illegible]30167	.8434[illegible]15	1.1300369	1.90077	.67197	[illegible]
[illegible]04393	-1.0000[illegible]77	1.3380861	2.14717	.66443	[illegible]
[illegible]3568	-1.17[illegible]49	1.5324159	2.42431	.63967	0
1.[illegible]24092	-1.36[illegible]700	1.7231060	2.73555	.61076	0
1.1127007	1.5550569	1.9121467	3.08171	.58191	0
1.1641246	-1.7403210	2.1004321	3.46004	.55423	0
1.2081168	-1.9434155	2.2883203	3.87215	.52775	0
1.2456279	-2.1390604	2.4760026	4.31587	.50300	0
1.2774876	-2.3373017	2.6636356	4.79297	.47960	0
1.3044925	-2.5354671	2.8513671	5.30470	.45750	0
1.32730[illegible]	-2.7341506	3.0393290	5.85090	.43674	0
-1.346824[illegible]	-2.9332019	3.2276321	6.43387	.41728	0
-1.3554344	-3.0328305	3.3217367	6.73920	.40803	0

```
            BOUNDARY
BRANCH STARTING AT (0.) + J(0.)
TYPE L TO LIST, S TO SKIP, OR $ TO ABORT >
? S
BRANCH STARTING AT (-2.) + J(0.)
TYPE L TO LIST, S TO SKIP, OR $ TO ABORT >
? L
```

BRANCH NUMBER 3

CALCULATION STEP SIZE = .1000
PRINTING STEP SIZE = .1500

LOCUS REAL	LOCUS IMAG	DIST TO ORIGIN	GAIN	ZETA	CD
-2.0000000	0.	2.0000000	0.	1.00000	1
1.8000000	0.	1.8000000	.703297	1.00000	0
-1.6000000	0.	1.6000000	1.24121	1.00000	0
1.4000000	0.	1.4000000	1.54121	1.00000	0
-1.2000000	0.	1.2000000	1.77231	1.00000	0
-1.0000000	0.	1.0000000	2.00000	1.00000	0
-.80000000	0.	.80000000	2.30741	1.00000	0
-.60000000	0.	.60000000	4.50947	1.00000	0
-.50000000	0.	.50000000	0.	1.00000	[illegible]

```
BRANCH STARTING AT (-4.) + J(0.)
TYPE L TO LIST   TO SKIP, OR $ TO ABORT >
```

ORIGINAL PAGE IS
OF POOR QUALITY

BRANCH NUMBER 4

CALCULATION STEP SIZE = .1000
PRINTING STEP SIZE = .1500

LOCUS REAL	LOCUS IMAG.	DIST TO ORIGIN	GAIN	ZETA	CR
4.0000000	0.	4.0000000	0.	1.00000	1
3.8000000	0.	3.8000000	.605874	1.00000	0
3.6000000	0.	3.6000000	1.21619	1.00000	0
3.4000000	0.	3.4000000	1.86023	1.00000	
3.2000000	0.	3.2000000	2.60063	1.00000	
3.0000000	0.	3.0000000	3.60000	1.00000	
2.8000000	0.	2.8000000	5.45391	1.00000	
2.6000000	0.	2.6000000	13.5200	1.00000	0
2.5000000	0.	2.5000000	0.	1.00000	

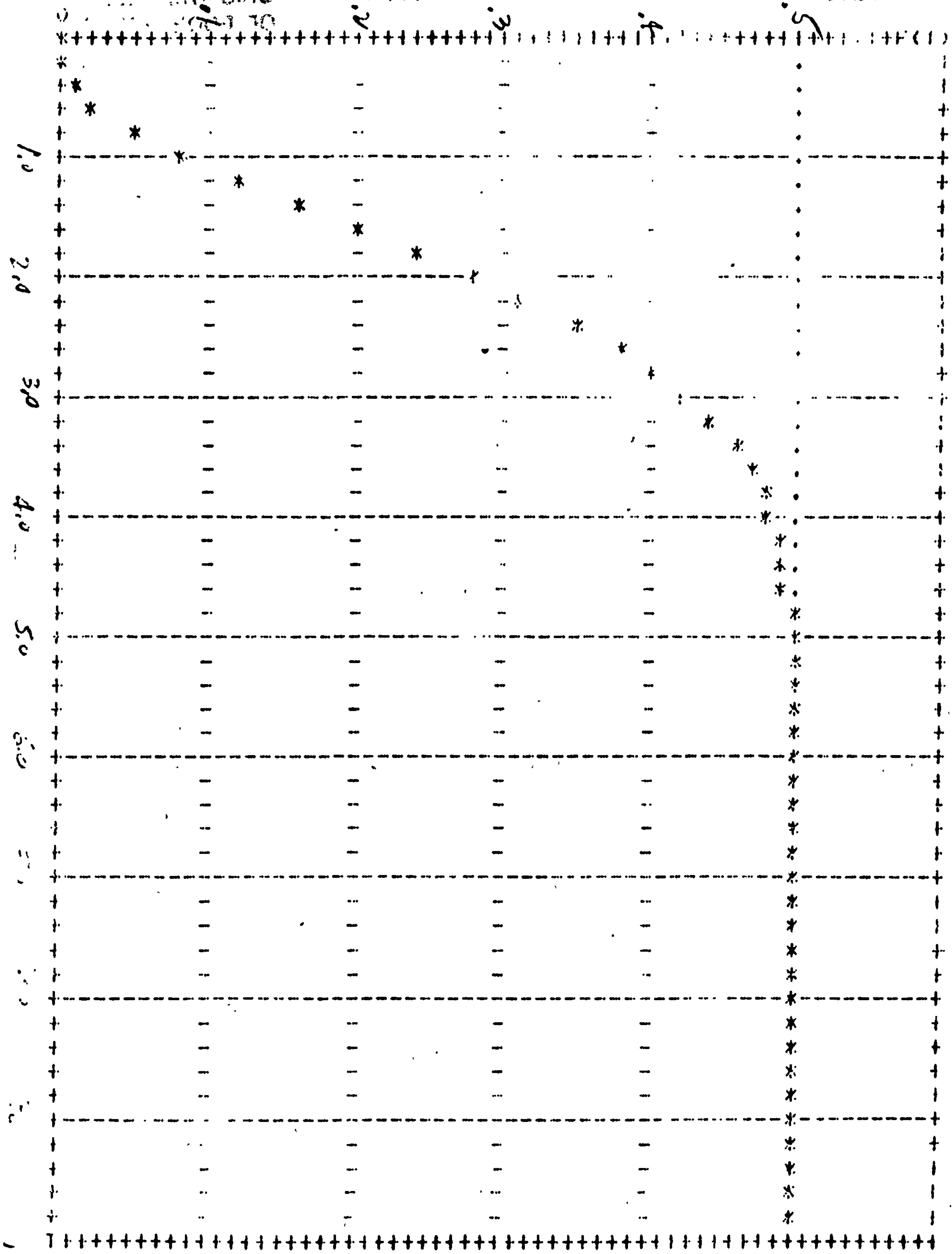
Choosing the gain to be 2.2 yields a closed loop transfer function as follows: Note that this matches the hand calculation.

```
                CLOSED-LOOP TRANSFER FUNCTION

                 CLK= ( CLNK/CLDK )=     2.200

                    CLTF(S) NUMERATOR
I        CLNPOLY(I)                          CLZERO(I)
1    (   2.200      )S** 1      (  -2.500     ) + J(  0.          )
2    (   5.500      )                      CLNK=   2.200

                   CLTF(S) DENOMINATOR
I        CLDPOLY(I)                          CLPOLE(I)
1    (   1.000      )S** 4      (  -.9089     ) + J(   1.034      )
2    (   6.000      )S** 3      (  -.9089     ) + J(   1.034      )
3    (   12.40      )S** 2      (  -.8790     ) + J(  0.          )
4    (   13.20      )S** 1      (  -3.303     ) + J(  0.          )
5    (   5.500      )                      CLDK=   1.000
```

The unit step response follows as does the frequency response.

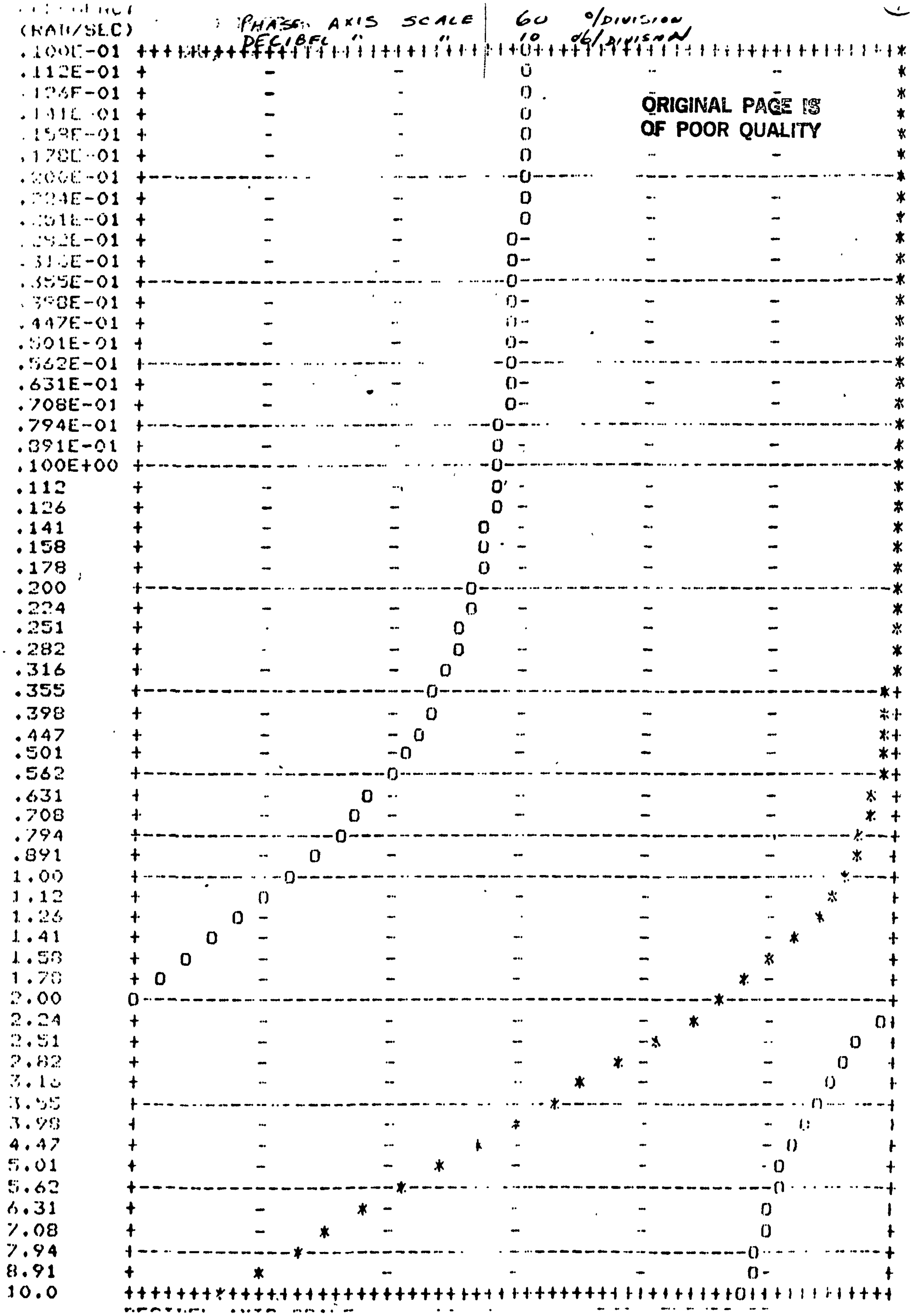
ORIGINAL PAGE IS
OF POOR QUALITY

ENTER INITIAL TIME, FINAL TIME
? 0,10
REGION OF CALCULATION: T= 0. TO T= 10.00
F(T)= 0. TO F(T)= 1.200



GRID SCALE: T-AXIS: 1.000 SECONDS/DIVISION
F(T) AXIS: .2000 UNITS/DIVISION

OPTION >
? 50


(RAD/SEC)
PHASE AXIS SCALE 60 °/DIVISION
DECIBEL " " 10 db/DIVISION
.100E-01
.112E-01
.178E-01
.224E-01
.355E-01
.447E-01
.501E-01
.562E-01
.631E-01
.708E-01
.794E-01
.891E-01
.100E+00
.112
.126
.141
.158
.178
.200
.224
.251
.282
.316
.355
.398
.447
.501
.562
.631
.708
.794
.891
1.00
1.12
1.26
1.41
2.00
2.24
2.51
2.82
3.98
4.47
5.01
5.62
6.31
7.08
7.94
8.91
10.0

```
PROGRAM SIMSING(INPUT,OUTPUT);
  [illegible]
CONST
   PSIC =0.5;
   CONST1 =1.0;
   CONST2 = 13.75;
   GRAV = 10.0;
   VELOCITY = 200.0;
   CONST4 = 1.0;
   CONST5 = 2.00;
   TAU1 = 0.40;
   TAU2 = 0.25;
   TAU3 = 0.50;
   TFINAL = 10.0;
   N = 5;
   PHILIM = 1.57;    (* ENTER IN RADIANS *)

TYPE
   WORD = PACKED ARRAY [1..10] OF CHAR;
   RANGE = ARRAY [1..4] OF REAL;
   ABSCISSA = ARRAY [1..1001] OF REAL;
   ORDINATE = ARRAY [1..1001,1..2] OF REAL;
VAR
   EPS,LEADX,LEADXD,DELT,DELTD,PHI,PHID,PHIDD,PSI,PSID,T: REAL;
   O : ARRAY [1..10] OF REAL;
   I : ARRAY [1..10,1..2] OF REAL;
   RANGE1 : RANGE;
   PLOTPSI : ORDINATE ;
   TIME : ABSCISSA;
   IY,P,M : INTEGER;
   IOPT,IER : INTEGER;
   PRINTI : INTEGER;
   X,J,K : INTEGER;
   XCH,ICHAR : CHAR;
   PLOTIME :INTEGER;
   NLETTERS: INTEGER;
   TITLE,XWORD,YWORD : WORD;
   DELTAT : REAL;
   INC : INTEGER;




BEGIN (* MAIN PROGRAM *)
   TITLE := 'PSI OF A/C';
   XWORD := 'TIME      ';
   YWORD := 'PSI       ';
   T := 0.0;
   LEADXD := 0.0;
   DELTD := 0.0;
   PHID := 0.0;
   PHIDD := 0.0;
   PLOTIME := 1;
   PSID  :=0.0;
   PRINTI := 0;
   LEADX := 0.0;
   DELT  := 0.0;
   PHI := 0.0;
   NLETTERS := 10;
   XCH := 'X';
   IOPT := 0;
   DELTAT := 0.01;
   IY :=101;
   P := [illegible]
```

```
M := 1 ;
FOR K:=1 TO 4 DO
   RANGE[K] := 0.0;
PSI := 0.0;
WHILE T <= TFINAL DO
BEGIN
   (* SET THE DERIVATIVE EQUATIONS *)
   EPS := PSIC - CONST4*PSI-CONST5*PSID;
   LEADXD := (EPS*CONST1- LEADX)/TAU2;
   DELT := LEADX +LEADXD*TAU1;
   PHIDD := (DELT*CONST2-PHID)/TAU3;
   PSID := (GRAV/VELOCITY) *PHI;

   (* SHUNT LIMITS THE AIRCRAFT ROLL ANGLE *)
   IF (PHI*PHID > 0.0) AND (ABS(PHI) >PHILIM) THEN PHID :=0.0
;


   (* ASSIGN *)
   I[1,1] := LEADX ;
   I[1,2] := LEADXD ;
   I[2,1] := DELT ;
   I[2,2] := DELTD ;
   I[3,1] := PHID;
   I[3,2] := PHIDD;
   I[4,1] := PHI ;
   I[4,2] := PHID;
   I[5,1] := PSI ;
   PLOTPSI[PLOTIME,1] := PSI;
   I[5,2] := PSID ;

   (* INTEGRATION *)
   FOR J := 1 TO N DO
     O[J]:= I[J,1] + I[J,2]*DELTAT;

   (* REASSIGNMENT *)
   LEADX := O[1];
   PHID:= O[3];
   PHI := O[4];
   PSI := O[5];

   (* PRINT OUT THE PARAMETERS EVERY (DELTAT*25) INTEGRATIONS
*)
   PRINTI := PRINTI + 1;
   IF PRINTI = 1 THEN
   BEGIN
      WRITELN;
      FOR X := 1 TO N DO
             WRITELN(T,I[X,1],I[X,2],O[X]);
      WRITELN;
      PRINTI:= PRINTI - 25;
   END;
   TIME[PLOTIME] := T;
   PLOTIME := PLOTIME + 1 ;
   T := T + DELTAT;
END;
ICHAR := 'X';
INC := 1;
WRITELN('SUCCESS');
END.
*
```

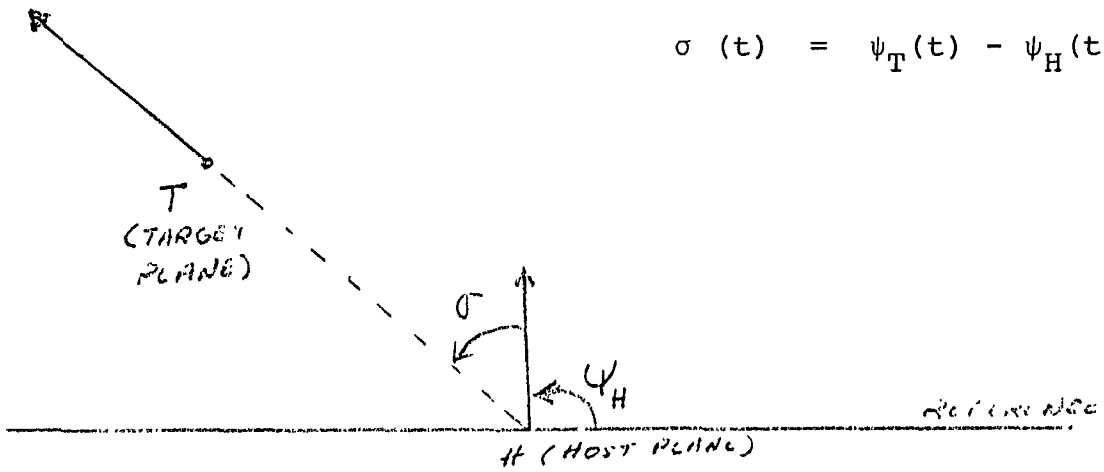
ORIGINAL PAGE IS OF POOR QUALITY

TRACKING EQUATIONS

1. POSITION PARAMETERS:

$$\| \overrightarrow{HT} \| = R$$

$$\sigma(t) = \psi_T(t) - \psi_H(t)$$



Let V_m be the speed of the projectile fired on the pursuing host aircraft. The time taken to reach the target plane is $T = V_m/R$. During this amount of time the target plane moves of an angle δ equal to $\dot{\psi}_T . T$. That means that the projectile must be fired before $\psi_T = \psi_H$, exactly when $\sigma = \delta$. If we assume that a tracking system capable of giving the angular position of the target plane relatively to the host plane is available, the angular speed $\dot{\psi}_T$ can be computed as:

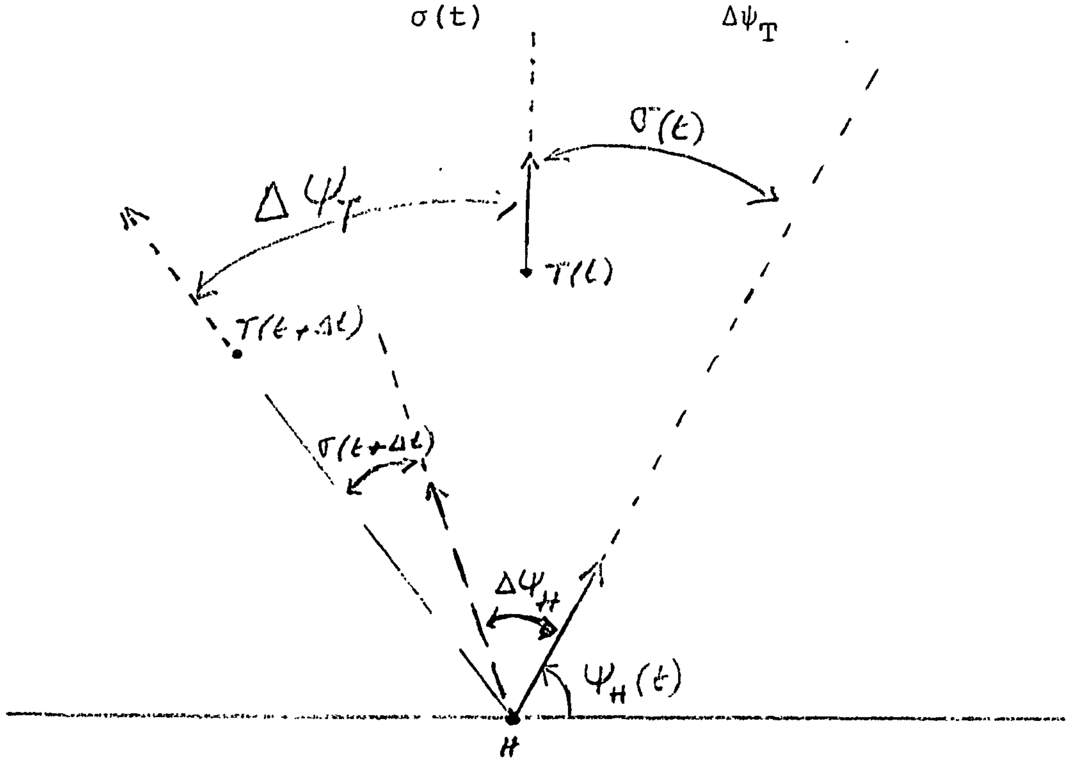
$$\dot{\psi}_T(t) = \frac{\psi_T(t+\Delta t) - \psi_T(t)}{\Delta t} \quad \text{for every step of time } \Delta t.$$

2. BASIC RELATIONS:

In order to shoot the target as soon as possible, that is the next step of time, the next value of ψ_H, must be:

$$\psi_H(t + \Delta t) - \psi_H(t) = \dot{\psi}_H(t) \times \Delta t$$

$$\Delta\psi_H + \delta = \underbrace{\psi_T(t) - \psi_H(t)}_{\sigma(t)} + \underbrace{\dot{\psi}_T(t) \times \Delta t}_{\Delta\psi_T}$$



$$\psi_H(t + \Delta t) = \psi_T(t + \Delta t) - \delta = \psi_T(t) + \dot{\psi}_T(t) \times \Delta t - \delta$$

$$\dot{\psi}_H(t + \Delta t) = \frac{\psi_H(t + \Delta t) - \psi_H(t)}{\Delta t}$$

We now need to generate a sequence of target positions, that is create the functions R and ψ_T. Two ways were discussed:

--- generating ψ_T using the host plane parameters: the behavior of the target is conditioned by the behavior of the chaser. This is an approach very close to reality.

--- programming functions ψ_T and R in an appropriate way. We assume that the planes are similar in performance, particularly in speed, so that the range between them can only differ of R of a small value ΔR. ψ_T must be a continuous function from $TR^{+} \longrightarrow (0, 2\Pi)$. The end of fight is determined by T_{max}, that in real case can be interpreted as a fuel shortage.

The last option was chosen for our project, because it allows us to implement easily (change of function) a different target behavior.

CALCULATION ALGORITHM

BEGIN..

INIT $\psi_T(t)$, $R(t)$, $\dot{\psi}_T(t)$;

$\delta(t) = \dot{\psi}_T(t) \times (V_m/R(t)$;

GET INPUT $\psi_H(t)$;

/*Calculation of next ψ: */

Do until TMAX:

$\psi_H(t + \Delta t) = \psi_T(t) + \dot{\psi}_T(t) \times \Delta t - \delta(t)$; (store ψ_H;)

(if $\psi_H(t + \Delta t) \quad \psi_H$max then $\psi_H = \psi_{Hmax}$; $\overset{\downarrow *}{\delta}_{SE}$

GET $\delta_T(t + \Delta t)$, $R(t + \Delta t)$;

$$\dot{\psi}_T(t + \Delta t) = \frac{\psi_T(t + \Delta t) = \psi_T(t)}{\Delta t};$$

(if $\dot{\psi}_T(t = \Delta t) \quad \dot{\psi}_T$max then $\dot{\psi}_T = \dot{\psi}_T$max;)

$\delta(t + \Delta t) = \dot{\psi}(t + \Delta t) \times (V_m/R\ (t + \Delta t)$;

NEXT:

$\psi_T(t) \leftarrow \psi_T(t + \Delta t)$

$\dot{\psi}_T(t) \leftarrow \dot{\psi}_T(t + \Delta t)$

$\delta(t) \leftarrow \delta(t + \Delta t)$

END;

EXAMPLES OF TARGET MOTION GENERATION

$$\psi_T(t) = \quad 2\,\pi \cos 2\,\pi\, t/t_{max}$$

$$2\,\pi \sin 2\,\pi\, t/t_{max}$$

$$2\,\pi \left(\frac{t}{t_{max}}\right)^2$$

..........(must be continued)

$$R(t) = R(\psi(t)) = \quad a\, e^{-\psi_T(t)} \qquad \text{(ellipse)}$$

$$a\,(1 + \cos\psi) \qquad \text{(cardioid)}$$

$$a = \text{constant}$$

. . .

(must be continued)

$\Delta R(t)$ = periodic or aleatory function giving small values (max $\simeq$ 10% . R).

(Must not be continued as it is a small value.)

LIMITATIONS

In order to stay close to reality, we introduced the following limitations: The host plane turning rate cannot be greater than (/20) radians/s, that is 9 degrees per second; the initial value of the target turning rate is set to 0.01 radians/s, that is 0.57 degrees per second.

CORRESPONDENCE ALGORITHM PROGRAM:

Algorithm	Program
$\psi_T(t)$	OPSIT
R	RANGE
$\dot{\psi}_T(t + \Delta t)$	NPSIT
δ	DELTA
Δt	DELTAT
$\dot{\psi}_H(t + \Delta t)$	HPSID
$\psi_H(t)$	PSI
$\psi_H(t + \Delta t)$	EXPPSI

In the computation of the tracking equations and anticipation parameters, converging results have been obtained with basic versions.

The "fire" order will be displayed each time $|\psi_H - \psi_T| <= \delta$.

Different test cases and air fight scenarios have been programmed and the results are satisfactory (fire orders displayed within 10 seconds--see listings attached).

EXPLANATION OF THE ENCOUNTER

On the graph ,the positions of the host and the target airplanes are plotted . Each number on the points represents a time step . The + marks are the series of predicted tracking angles placed with the base of the next attaching the arrow of the previous tracking vector . This allows us to see that the series of optimal angles are drawing a curve looking like the target trajectory . This is a verification of the tracking angle generation procedure . The points 6,7,8,9 of the host plane trajectory show the optimal angle relative to the actual position of the plane . This is the real case.

For each position of the aircraft the procedure gives the optimal angle the pilot has to command in order to shoot the target as soon as possible . As we can see on the graph , this command makes the plane point directly to the target .

The firing condition is determined by a "window" of + or - delta , delta being the lead angle depending on the range between the planes and the missile velocity .

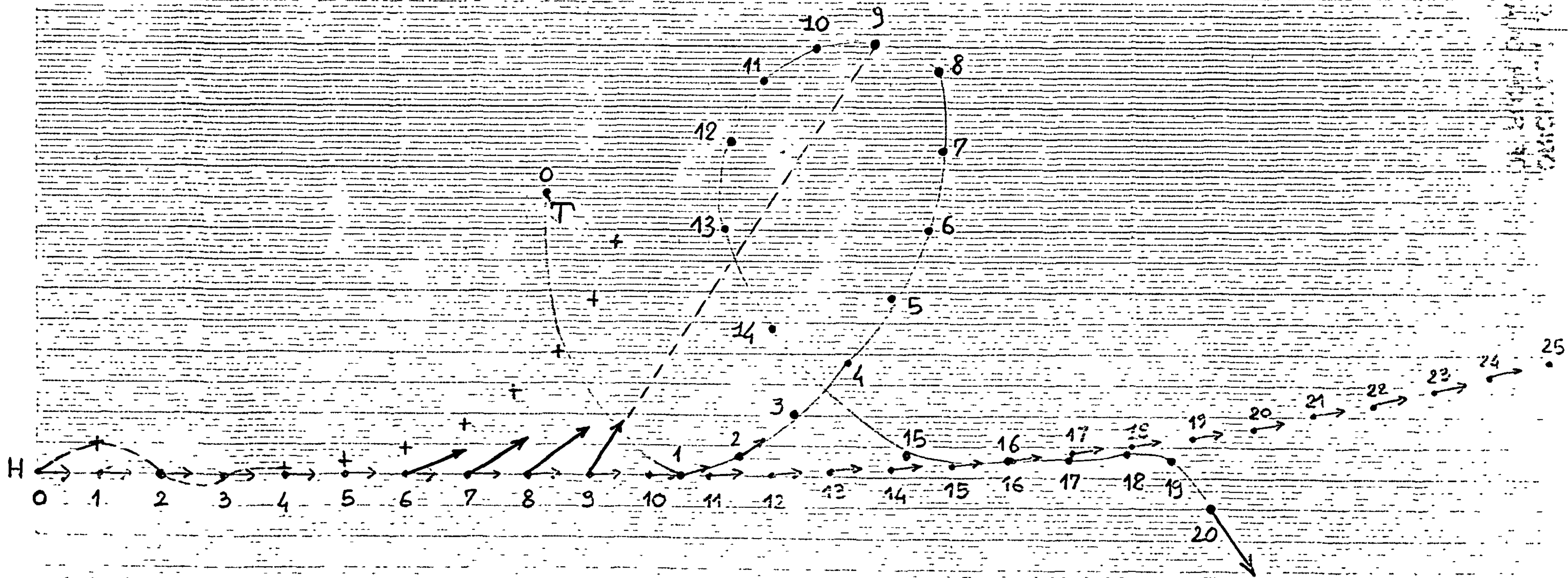
PHOT35 . PAS SCENARIO

$$\psi_T = 2\pi - 2\pi \cos 2\pi \frac{t}{t_{max}}$$

$$R = 1000 . (1 + \cos \psi_T)$$

(+ tracking angles accumulated)

non corrected host (response to step input)



ORIGINAL PAGE IS
OF POOR QUALITY

```
*Start* Job PHOT35 Req #3576 for NETLPT  Date 18-Dec-82 16:26:56 Monitor: USC
*Start* Job PHOT35 Req #3576 for NETLPT  Date 18-Dec-82 16:26:56 Monitor: USC
*Start* Job PHOT35 Req #3576 for NETLPT  Date 18-Dec-82 16:26:56 Monitor: USC

BBBB   L       U   U  EEEEE  N   N
B   B  L       U   U  E      N   N
B   B  L       U   U  E      NN  N
BBBB   L       U   U  EEEE   N N N
B   B  L       U   U  E      N  NN
B   B  L       U   U  E      N   N
BBBB   LLLLL   UUUUU  EEEEE  N   N

*Start* Job PHOT35 Req #3576 for NETLPT  Date 18-Dec-82 16:27:02 Monitor: USC
*Start* Job PHOT35 Req #3576 for NETLPT  Date 18-Dec-82 16:27:02 Monitor: USC
*Start* Job PHOT35 Req #3576 for NETLPT  Date 18-Dec-82 16:27:02 Monitor: USC

PPPP   H   H   OOO   TTTTT   333   55555        PPPP    AAA    SSSS
P   P  H   H  O   O    T    3   3  5            P   P  A   A  S
P   P  H   H  O   O    T        3  555          P   P  A   A  S
PPPP   HHHHH  O   O    T       3      5         PPPP   A   A   SSS
P      H   H  O   O    T        3      5        P      AAAAA      S
P      H   H  O   O    T    3   3  5   5   ..   P      A   A      S
P      H   H   OOO     T     333    555    ..   P      A   A  SSSS

*Start* Job PHOT35 Req #3576 for NETLPT  Date 18-Dec-82 16:27:02 Monitor: USC
*Start* Job PHOT35 Req #3576 for NETLPT  Date 18-Dec-82 16:27:02 Monitor: USC
*Start* Job PHOT35 Req #3576 for NETLPT  Date 18-Dec-82 16:27:02 Monitor: USC

 SSSS                                   GGGG  U   U  M   M  BBBB   Y   Y
S         i     t                      G      U   U  MM MM  B   B   Y Y
S              ttt     eee      ##     G      U   U  M M M  B   B    Y
 SSS      i     t     e   e     ##     G      U   U  M   M  BBBB     Y
    S     i     t     eeeee            G  GGG U   U  M   M  B   B    Y
    S     i     t     e         ##     G   G  U   U  M   M  B   B    Y
SSSS      i      tt    eeee     ##      GGG   UUUUU  M   M  BBBB     Y

File: PS:<SPOOL>PHOT35.PAS.1/USER:BLUEN/COPIES:1/LIMIT:59/FORM:NORMAL/FILE:ASC
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
0123456789012345678901234567890123456789012345678901234567890123456789012

[PHOTO  Recording initiated  Sat 18-Dec-82 4:0 PM]

[Link from BLUEN, TTY 30]

 TOPS-20 Command processor 4(560)
@TY SIMOL.PAS
PROGRAM SIM3DOF(INPUT,OUTPUT);


CONST
    PSIC =0.5;
    CONST1 =1.0;
    CONST2 = 13.75;
    GRAV = 10.0;
    VELOCITY = 200.0;
    CONST4 = 1.0;
    CONST5 = 2.00;
    TAU1 = 0.40;
    TAU2 = 0.25;
    TAU3 = 0.50;
    TFINAL = 10.0;
    N = 5;
    PHILIM = 1.57;    (* ENTER IN RADIANS *)
    PI = 3.141592 ;
    MISSLV = 1000.0;

VAR

    EPS,LEADX,LEADXD,DELT,DELTD,PHI,PHID,PHIDD,PSI,PSID,T: REAL;
    C : ARRAY [1..10] OF REAL;
    I : ARRAY [1..10,1..2] OF REAL;
    PRINTI : INTEGER;
    X,J,K : INTEGER;
    XCH,ICHAR : CHAR;
    R,DELTAT : REAL;
    INC : INTEGER;
    CPSIT,NPSIT,RANGE,PSITD,DELTA,EXPPSI,HPSID,EXPPHI,HPHID: REAL;
    HPSIDMAX : REAL ;


BEGIN (* MAIN PROGRAM *)
    T := 0.0;
    LEADXD := 0.0;
    DELTD := 0.0;
    PHID := 0.0;
    PHIDD := 0.0;
    PSID  :=0.0;
    PRINTI := 0;
    LEADX := 0.0;
    DELT  := 0.0;
    PHI := 0.0;
    DELTAT := 0.10 ;
    RANGE:= 2000.0;
    CPSIT:= 0.5;
    PSITD:= 0.01;
    DELTA:= PSITD*MISSLV/RANGE;
    PSI := 0.0;
    HPSIDMAX := PI/20 ;
    (*                       ___________MAIN  LOOP_____________________________*
    WHILE T <= TFINAL DO
    BEGIN
```

47

```
    WRITELN;
    WRITELN;
    WRITELN;
    WRITELN;
    WRITELN (TTY , 'TIME STEP : ' , T,'-------------------------------*');
    WRITELN;
    R:= RANGE*(1+COS(OPSIT))/2;
    WRITELN ('RANGE =     ',R);
    WRITELN;
    EPS := PSIC - CONST4 * PSI - CONST5*PSID;
    LEADXD := (EPS*CONST1- LEADX)/TAU2;
    DELT := LEADX +LEADXD*TAU1;
    PHIDD := (DELT*CONST2-PHID)/TAU3;
    PSID := (GRAV/VELOCITY) *PHI;
    WRITELN (TTY , 'ACTUAL HOST TURNING ANGLE :                   ' , PSI ) ;
    WRITELN;
    (* SHUNT LIMITS THE AIRCRAFT ROLL ANGLE *)
    IF (PHI*PHIC > 0.0) AND (ABS(PHI) >PHILIM) THEN PHIC :=0.0;
  (* CALCULATION OF THE EXPECTED PSI OF THE HOST PLANE *)
  EXPPSI:= OPSIT+PSITD*DELTAT;
  WRITELN (TTY ,'EXPECTED TURNING ANGLE : ' ) ;
  (* SETS EXPPSI MODULO 2*PI *)
   WHILE EXPPSI > 2 * PI DO    EXPPSI := EXPPSI - 2*PI ;
   WHILE EXPPSI < -(2*PI) DO   EXPPSI := EXPPSI + 2*PI ;
  WRITELN (TTY , '----------------------------->    ', EXPPSI) ;
  WRITELN;
   IF   ABS(OPSIT - PSI) <= DELTA   THEN
             WRITELN ( '<<..*** FIRE *** FIRE*** FIRE ***..>>');

  HPSID:= (EXPPSI - PSI ) / DELTAT;
  WRITELN ;
  WHILE ABS(HPSID) > HPSIDMAX DO
        HPSID := (HPSIDMAX * HPSID / ABS(HPSID)) ;
  WRITELN ( ' ESTIMATED HOST TURNING RATE  : ' , HPSID   );
  WRITELN;

  (*-------GENERATION OF THE MOTION OF THE TARGET PLANE------------------*

  NPSIT:= 2*PI - ( 2*PI*COS(2*PI*(T+DELTAT)/TFINAL));
  WRITELN ( ' ANTICIPATED TARGET LINE OF SIGHT ' , NPSIT ) ;
  WRITELN;

  (* -----------------------------------------------------------------*

  EXPPHI:= HPSID*VELOCITY/GRAV;       (* EXPECTED VALUE OF PHI *)
  HPHIC:= (EXPPHI-PHI)/DELTAT;        (* COMMAND TO BE GENERATED *)
  PSITD:= (NPSIT-OPSIT)/DELTAT;       (* TURNING RATE OF THE TARGET *)
  DELTA:= ABS(PSITD*MISSLV/R);        (* NEW ANGLE OF FIRING        *)
  WRITELN ('ANTICIPATED TOLERANCE DELTA ANGLE =   ',DELTA);
  WRITELN ('                      ------ ');
  OPSIT:= NPSIT;                      (* REASSIGNMENT               *)

  (*  ---------------------------------------------------------------*
     (* ASSIGN *)
     I[1,1] := LEADX ;
     I[1,2] := LEADXD ;
     I[2,1] := DELT ;
     I[2,2] := DELTD ;
     I[3,1] := PHID;
     I[3,2] := PHIDD;
     I[4,1] := PHI ;
     I[4,2] := PHIC;
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
        T[5,1] := PSI ;
        T[5,2] := PSID ;

        (* INTEGRATION *)
        FOR J := 1 TO N DO
          C[J]:= I[J,1] + I[J,2]*DELTAT;

        (* REASSIGNMENT *)
        LEADX := C[1];
        PHID:= C[3];
        PHI := C[4];
        PSI := C[5];

        (* PRINT OUT THE PARAMETERS EVERY (DELTAT*25) INTEGRATIONS *)
        PRINTI := PRINTI + 1;
        IF PRINTI = 1 THEN
        BEGIN
          WRITELN;
          FOR X := 1 TO N DO
                WRITELN(T,I[X,1],I[X,2],C[X]);
                WRITELN;
          WRITELN('                        HOST AIRCRAFT') ;
          WRITELN;
                WRITELN('EXPPSI = ',EXPPSI,'      HPSID = ',HPSID);
          WRITELN;
                WRITELN('EXPPHI = ',EXPPHI,'      HPHID = ',HPHID);
                WRITELN;
                WRITELN;
          WRITELN('                        TARGET')  ;
          WRITELN;
                WRITELN('PSITD = ',PSITD);
                WRITELN('NPSIT = ',NPSIT);
          WRITELN;
          WRITELN;
          WRITELN;
          PRINTI:= PRINTI - 25;
        END;
        T := T + DELTAT;
     END;
     WRITELN('SUCCESS');
END.
@
@
@
@
@EX SIMOL.PAS
LINK:   Loading
[LNKXCT SIM3DO execution]
INPUT      :
OUTPUT     :
[INPUT, end with ^Z: ]
^Z


TIME STEP :      0.000000E+00---------------------------*

RANGE =        1.877583E+03

ACTUAL HOST TURNING ANGLE :                 0.000000E+00

EXPECTED TURNING ANGLE :
```

49

ORIGINAL PAGE IS
OF POOR QUALITY

```
---- ---------------------------->        5.010000E-01

 ESTIMATED HOST TURNING RATE  :       1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT       1.239848E-02
ANTICIPATED TOLERANCE DELTA ANGLE =        2.596964E+00

   [illegible]E+00    0.000000E+00    2.000000E+00    2.000000E-01
   0.000000E+00    8.000000E-01    0.000000E+00    8.000000E-01
   0.000000E+00    [illegible]E+00    2.200000E+01    2.200000E+00
   0.000000E+00    0.000000E+00    0.000000E+00    0.000000E+00
   0.000000E+00    0.000000E+00    0.000000E+00    0.000000E+00
                     HOST AIRCRAFT
EXPFSI =       5.010000E-01      PFSID =      1.570796E-01
EXPFHI =       3.141592E+00      PFHID =      3.141592E+01

                        TARGET
PSITD =    -4.876015E+00
NFSIT =     1.239848E-02


TIME STEP :      1.000000E-01------------------------------*
RANGE =       1.999923E+03
ACTUAL HOST TURNING ANGLE :                       0.000000E+00
EXPECTED TURNING ANGLE :
------------------------------->       -4.752030E-01
<<..*** FIRE *** FIRE*** FIRE ***..>>
 ESTIMATED HOST TURNING RATE  :     -1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT      4.954481E-02
ANTICIPATED TOLERANCE DELTA ANGLE =       1.857388E-01


TIME STEP :     2.000000E-01-----------------------------*
RANGE =       1.998773E+03
ACTUAL HOST TURNING ANGLE :                       0.000000E+00
EXPECTED TURNING ANGLE :
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
----- -----------------------> 8.669114E 2
<<..*** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE  :    1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT    1.112924E-01
ANTICIPATED TOLERANCE DELTA ANGLE =    3.089273E-01


TIME STEP :     3.000000E-01-------------------------------*
RANGE =       1.993813E+03
ACTUAL HOST TURNING ANGLE :                 1.100000E-03
EXPECTED TURNING ANGLE :
------------------------------->        1.730399E-01
<<..*** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE  :    1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT    1.973977E-01
ANTICIPATED TOLERANCE DELTA ANGLE =    4.318626E-01


TIME STEP :     4.000000E-01-------------------------------*
RANGE =       1.980580E+03
ACTUAL HOST TURNING ANGLE :                 4.015000E-03
EXPECTED TURNING ANGLE :
------------------------------->        2.335031E-01
<<..*** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE  :    1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT    3.075207E-01
ANTICIPATED TOLERANCE DELTA ANGLE =    5.560140E-01


TIME STEP :     5.000000E-01-------------------------------*
RANGE =       1.953087E+03
ACTUAL HOST TURNING ANGLE :                 9.218000E-03
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
EXPECTED TURNING ANGLE :
------------------------------->        4.176439E-01
<<.**** FIRE *** FIRE*** FIRE ****..>>
  ESTIMATED HOST TURNING RATE  :      1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT      4.412270E-01
ANTICIPATED TOLERANCE DELTA ANGLE =        6.845894E-01


TIME STEP :      6.000000E-01-----------------------------*
RANGE =          1.904228E+03
ACTUAL HOST TURNING ANGLE :                        1.697718E-02
EXPECTED TURNING ANGLE :
------------------------------->        5.749323E-01
<<.**** FIRE *** FIRE*** FIRE ****..>>
  ESTIMATED HOST TURNING RATE  :      1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT      5.979890E-01
ANTICIPATED TOLERANCE DELTA ANGLE =        6.232308E-01


TIME STEP :      7.000000E-01-----------------------------*
RANGE =          1.826469E+03
ACTUAL HOST TURNING ANGLE :                        2.739279E-02
EXPECTED TURNING ANGLE :
------------------------------->        7.547509E-01
<<.**** FIRE *** FIRE*** FIRE ****..>>
  ESTIMATED HOST TURNING RATE  :      1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT      7.771875E-01
ANTICIPATED TOLERANCE DELTA ANGLE =        9.811199E-01


TIME STEP :      8.000000E-01-----------------------------*
RANGE =          1.712889E+03
ACTUAL HOST TURNING ANGLE :                        3.780841E-02
```

52

```
EXP...TED TURNING ANGLE :
-------------------------------->          9.563861E-01
<<..*** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE   :     1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT     9.761160E-01
ANTICIPATED TOLERANCE DELTA ANGLE =      1.173039E+00


TIME STEP :      9.000000E-01-----------------------------*
RANGE =        1.558586E+03
ACTUAL HOST TURNING ANGLE :                          4.808035E-02
EXPECTED TURNING ANGLE :
-------------------------------->         1.179044E+00
<<..*** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE   :     1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT     1.199981E+00
ANTICIPATED TOLERANCE DELTA ANGLE =      1.423501E+00


TIME STEP :      1.000000E+00-----------------------------*
RANGE =        1.362376E+03
ACTUAL HOST TURNING ANGLE :                          5.835229E-02
EXPECTED TURNING ANGLE :
-------------------------------->         1.421846E+00
<<..*** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE   :     1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT     1.441907E+00
ANTICIPATED TOLERANCE DELTA ANGLE =      1.775766E+00


TIME STEP :      1.100000E+00-----------------------------*
RANGE =        1.128533E+03
```

53

```
ACTUAL HOST TURNING ANGLE :                           6.8 ?423E-02
EXPECTED TURNING ANGLE :
------------------------------->          1.683833E+00
<<..*** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE  :        1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT       1.702939E+00
ANTICIPATED TOLERANCE DELTA ANGLE =        2.313024E+00

TIME STEP :     1.200000E+00-----------------------------*
RANGE =         8.682413E+02
ACTUAL HOST TURNING ANGLE :                           7.889617E-02
EXPECTED TURNING ANGLE :
------------------------------->          1.963972E+00
<<..*** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE  :        1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT       1.982048E+00
ANTICIPATED TOLERANCE DELTA ANGLE =        3.214640E+00

TIME STEP :     1.300000E+00-----------------------------*
RANGE =         6.002433E+02
ACTUAL HOST TURNING ANGLE :                           8.916811E-02
EXPECTED TURNING ANGLE :
------------------------------->          2.261156E+00
<<..*** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE  :        1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT       2.278131E+00
ANTICIPATED TOLERANCE DELTA ANGLE =        4.932718E+00

TIME STEP :     1.400000E+00-----------------------------*
RANGE =         3.501900E+02
```

54

ORIGINAL PAGE IS OF POOR QUALITY

```
ACTUAL HOST TURNING ANGLE :                         9.5[illegible]C05E-02
EXPECTED TURNING ANGLE :
------------------------------->          2.574214E+00
<<..*** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE  :         1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT         2.590020E+00
ANTICIPATED TOLERANCE DELTA ANGLE =          2.906285E+00


TIME STEP :      1.500000E+00----------------------------*
RANGE =          1.482986E+02
ACTUAL HOST TURNING ANGLE :                         1.097120E-01
EXPECTED TURNING ANGLE :
------------------------------->          2.901909E+00
<<..*** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE  :         1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT         2.916484E+00
ANTICIPATED TOLERANCE DELTA ANGLE =          2.201400E+01


TIME STEP :      1.600000E+00----------------------------*
RANGE =          2.523005E+01
ACTUAL HOST TURNING ANGLE :                         1.195839E-01
EXPECTED TURNING ANGLE :
------------------------------->          3.242949E+00
<<..*** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE  :         1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT         3.[illegible]0236E+00
ANTICIPATED TOLERANCE DELTA ANGLE =          1.346614E+02


TIME STEP :      1.700000E+00----------------------------*
```

```
RANG  =        6.564315E+00
ACTUAL HOST TURNING ANGLE :                              1.302559E-01
EXPECTED TURNING ANGLE :
------------------------------->            3.595987E+00
<<..*** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE  :        1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT       3.607933E+00
ANTICIPATED TOLERANCE DELTA ANGLE =        5.357709E+02


TIME STEP :      1.800000E+00-----------------------------*
RANGE =         1.067802E+02
ACTUAL HOST TURNING ANGLE :                              1.405278E-01
EXPECTED TURNING ANGLE :
------------------------------->            3.959630E+00
<<..*** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE  :        1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT       3.970188E+00
ANTICIPATED TOLERANCE DELTA ANGLE =        3.392533E+01


TIME STEP :      1.900000E+00-----------------------------*
RANGE =         3.240863E+02
ACTUAL HOST TURNING ANGLE :                              1.507998E-01
EXPECTED TURNING ANGLE :
------------------------------->            4.332443E+00
<<..*** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE  :        1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT       4.341572E+00
ANTICIPATED TOLERANCE DELTA ANGLE =        1.145934E+01


TIME STEP :      2.000000E+00-----------------------------*
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
RANGE            6.376226E+02
ACTUAL HOST TURNING ANGLE :                      1.610717E-01
EXPECTED TURNING ANGLE :
------------------------------>          4.712955E+00
<<.**** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE  :      1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT      4.720618E+00
ANTICIPATED TOLERANCE DELTA ANGLE =       5.944677E+00


TIME STEP :     2.100000E+00------------------------------*
RANGE =        1.008229E+03
ACTUAL HOST TURNING ANGLE :                      1.713436E-01
EXPECTED TURNING ANGLE :
------------------------------>          5.099664E+00
<<.**** FIRE *** FIRE*** FIRE ***..>>
  ESTIMATED HOST TURNING RATE  :      1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT      5.105831E+00
ANTICIPATED TOLERANCE DELTA ANGLE =       3.620691E+00


TIME STEP :     2.200000E+00------------------------------*
RANGE =        1.383369E+03
ACTUAL HOST TURNING ANGLE :                      1.816156E-01
EXPECTED TURNING ANGLE :
------------------------------>          5.491044E+00

  ESTIMATED HOST TURNING RATE  :      1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT      5.495690E+00
ANTICIPATED TOLERANCE DELTA ANGLE =       2.618186E+00


TIME STEP :     2.300000E+00------------------------------*
```

```
RANGE =        1.705622E+03
ACTUAL HOST TURNING ANGLE :                     1.918875E-01
EXPECTED TURNING ANGLE :
------------------------------>        5.685549E+00

  ESTIMATED HOST TURNING RATE  :     1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT     5.888657E+00
ANTICIPATED TOLERANCE DELTA ANGLE =      2.303953E+00


TIME STEP :      2.400000E+00------------------------------*
RANGE =        1.923178E+03
ACTUAL HOST TURNING ANGLE :                     2.021095E-01
EXPECTED TURNING ANGLE :
------------------------------>        6.281625E+00

  ESTIMATED HOST TURNING RATE  :     1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT     6.283181E+00
ANTICIPATED TOLERANCE DELTA ANGLE =      2.051418E+00


TIME STEP :      2.500000E+00------------------------------*
RANGE =        2.000000E+03
ACTUAL HOST TURNING ANGLE :                     2.124314E-01
EXPECTED TURNING ANGLE :
------------------------------>        3.945217E-01

  ESTIMATED HOST TURNING RATE  :     1.570796E-01
 ANTICIPATED TARGET LINE OF SIGHT     6.677706E+00
ANTICIPATED TOLERANCE DELTA ANGLE =      1.972621E+00

     2.500000E+00     1.073179E-01    -1.027524E-01     9.754266E-02
     2.500000E+00     6.671694E-02     0.000000E+00     6.671694E-02
     2.500000E+00     0.000000E+00     1.489708E+00     1.489706E-01
     2.500000E+00     2.054388E+00     0.000000E+00     2.054388E+00
     2.500000E+00     2.124314E-01     1.027194E-01     2.227033E-01
```

58

C O N C L U S I O N

We have successfully demonstrated the ability of the On Board Simulator and Tracking Procedure in high level PASCAL code . The implementation details have yet to be completely worked out .

The On Board Simulator was tested using a step input and studying its response . The aircraft turning time constant is 4.4 seconds and damping ratio is .65 , which provided satisfactory results . The final simulator will fly at a velocity of 1000 feet per second and have a maximum bank angle of 82.8 degrees for an 8g turn (without aerodynamic limitation) .

The angle tracking is done relative to an inertial reference angle stored in the host aircraft's reference system . Quick encounters are simulated by inputting a programmed target maneuver into the host aircraft's field of view .

A more accurate tracking method could be developed by following the host and target aircraft separately in an inertal grid and computing relative information from the inertial systems . This would allow longer time engagements .

The On Board Simulator would not be easily upgraded from its present 3 degrees-of-freedom . This would require additional graphics on the screen display and would require much additional work on generating relative dynamics .

APPENDIX B

SPIN WARNING SYSTEM

TEAM #1 FINAL REPORT

Aircraft Spin Warning System Using Voice Generation Techniques

Presented to

Professor Smyth

Department of Electrical Engineering

University of Southern California

Submitted by

David Barry

David Lan Ho

Renshan Tang

Mohammad H. Movahed-Ezazi

Advanced Microcomputer Based Design

EE 560L

December 31, 1982

ORIGINAL PAGE IS
OF POOR QUALITY

Table of Contents

I. System Description

A. Purpose

The objective of the project is to design a microprocessor based Aircraft Spin Warning System which periodically samples the assymetric thrust and yaw rate of an airplane and then issues voice synthesized warnings and/or suggestions to the pilot of how to response to the situation.

The system is to meet the requirements set forth in the June-August 1982 status report of the system study for the Application of Microcomputers to Research Flight Test Techniques, as summarized in table 1 of the paper (included in this report in Figure 1).

Such a system is expected to aid the pilot in recovery from spins and high speed departures which occur during flight tests of aircraft at flight boundary limits.

B. Type of Microcomputer to be Used

Our SWS design is based around the Motorola MC68000 16-bit microprocessor, implemented on a M68KVM01AZ monoboard microcomputer. Since the SWS is a real time application where computing speed is critical, a very fast microprocessor must be selected. The advanced contemporary design, 8-MHZ clock

YAW RATE	ASYMMETRIC THRUST			SYMMETRIC THRUST (NO BLOWOUT OR STALL)
	AB BLOWOUT	AB STALL	MIL STALL	
Less than 40^{o}/sec	"Unload" If IAS $<$ 150 KTS "Left or Right (Lighted AB) Engine Mil"	"Unload" "Left or Right (Stalled) Eng Idle" If IAS $<$ 175 KTS "Left or Right (Undtalled) Eng Mil"	"Unload" If IAS $<$ 150 & Altitude $<$ 15K "Left or Right Eng Idle" (Unstalled Eng)	No Warning
Less than 50^{o}/sec but greater than 40^{o}/sec	"Unload" "Left or Right Engine Idle"	"Unload" "Both Engines Idle"	"Unload" "Both Engines Idle"	"Unload"
Greater than 50^{o}/sec	"Left or Right (Lighted AB) Engine Mil" "Stick Full Left or Right" "Stick Full Fwd"	"Both Engines Idle" "Stick Full Left or Right" "Stick Full Fwd"	"Both Engines Idle" "Stick Full Left or Right" "Stick Full Fwd"	"Stick Full Left or Right"

After any warning from table above:

If Yaw Rate $<$ 40^{o}/sec for two seconds "Center Controls".

If Yaw Rate $<$ 30^{o}/sec for 2 seconds and Airspeed $>$ 120 KTS "Recovery Complete".

AOA $<$ -10^{o} and AY $>$ .1g "Stick Half-Aft and Hold": then;

If AOA $>$ 0 for 5 sec "Recovery Complete".

If Temperature $>$ 1215^{o}C for 3 sec "Left or Right Eng Off".

Table 1. Logic Table

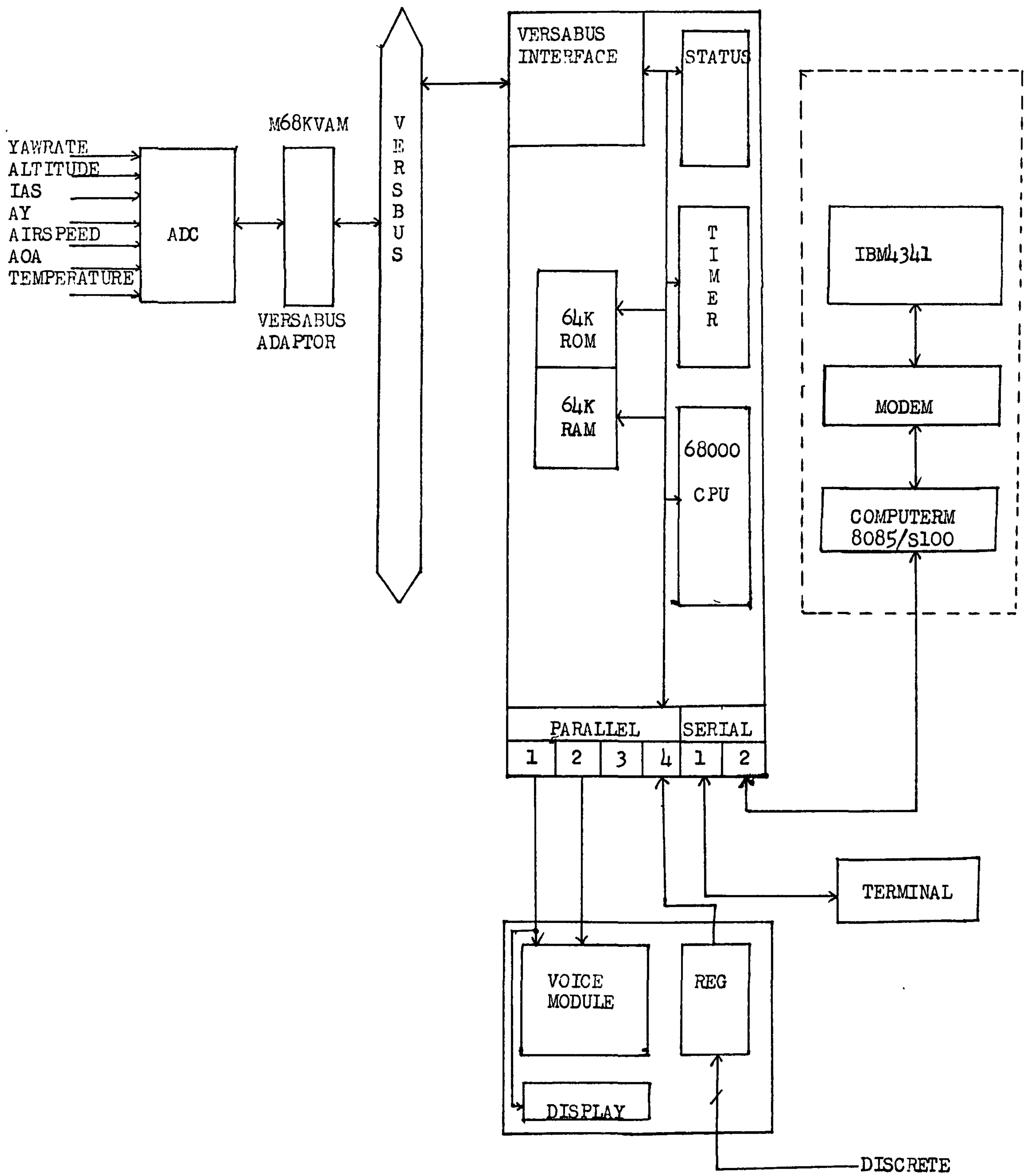


Figure 1. System Block Diagram

rate, and 16-bit word width all contribute to fast processing times and make the MC68000 a good choice for this application.

Convenient I/O and bus structures provide additional advantages that simplify and smooth system design and integration since we have access to peripheral devices with compatable I/O buses.

C. Selection of Speed Synthesizer Device

The SWS project is a developmental project and so requires versatile design features to allow for the many design changes that always occur during the development of a system. This is the main reason behind the selection of the SC01 from the various voice synthesis products on the market. The phoneme based SC01 allows us to speak any word desired, and easily accomodates any changes in the future. LPC based speech methods provide better voice quality, but have restricted and unversatile vocabularies and are much more expensive. The SC01, on the other hand, has been on the market for a couple of years and has well documented dictionaries and application articles making it a clear choice for a short term, low budget project.

D. Voice Synthesis Design Description

The block diagram of the voice synthesis design is shown in Figure 1 . The one byte of discrete data, seven channels of ADC data, and PTM timer are controled by the input interface. The two byte output data (phoneme and pitch) is transferred to SC01 voice synthesizer via the output interface as well as to the hexadecimal LED Display for test of the data.

The software moniter aircraft parameters from the ADC and discrete input register, analyzes these parameters and then, if necessary, sends voice synthesizer to speak out the required message.

The functional details of the modules and their design are described in the next two chapter.

II. Software Development

A. General Description

As indicated in Table 1, there are a total of 14 sentences to be spoken in various cases. These sentences are 'UNLOAD', 'LEFT-ENG-MIL', 'RIGHT-ENG-MIL', 'LEFT-ENG-IDLE', 'RIGHT-ENG-IDLE', 'BOTH-ENG-IDLE', 'STICK-FULL-LEFT', 'STICK-FULL-RIGHT', 'STICK-FULL-FWD', 'CENTER-CONTROLS', 'RECOVERY-COMPLETE', 'STICK-HALF-AFT-AND-HOLD', 'LEFT-ENG-OFF', 'RIGHT-ENG-OFF'. In some conditions, depending on input values received from ADC channels, up to 3 sentences need to be spoken. Each sentence may contains 8 to 26 phonemes. For each phonemes, the SC01 takes an average of 100 ms to complete its speech synthesis process. Therefore, in some cases, the time consumed to speak three sentences may take more than 2 seconds. Two seconds is a large amount time for computer busy waiting for phoneme output. Note that another requirement in Table 1 is to keep sensing each ADC channel to determine if the situation has changed or not. The sensing rate should be much less than 2 seconds. It is obvious that there are two requirements to be performed: (1) the software has to keep sensing ADC channels while the SC01 is speaking some sentences. (2) a real time clock is necessary to count the actual time that the temperature exceeded the temperature tolerance limit or for how longer the YAW RATE has recovered to its normal condition, etc.

Therefore, a program 'VOICE' and a subprogram 'SMALLMOUTH' are implemented to solve the above requirements. The program 'VOICE' is used to dump sentences in an output buffer, the subprogram SMALLMOUTH will send all phoneme and pitchcodes from the output buffer to the SC01 via 2 8-bit ports for voice synthesis. These two programs interleave the CPU by timer and parallel port interrupt. The data structure for the sentence table and buffers are shown in Figure . Each buffer is a record which contains a bufferfull flag, along with 50 phoneme codes and 50 pitch codes. The program VOICE contains a procedure SET-SENTENCE-TABLE which stores the phoneme and pitch symbols of all sentences. Then VOICE sets up the timer clock, senses ADC channels when timer interrupts arrive, and then according to the ADC channel value, decides which sentences are to be spoken and dumps them into output-buffer. The subprogram will send phoneme and pitch codes from the outputbuffer to the SC01 and wait for the phoneme to finish its sound generation. After a phoneme code has completed its sound generation process in the SC01, the SC01 will send back a parallel port interrupt request to execute the subprogram SMALLMOUTH again and get next phoneme and pitch code.

The program VOICE is classfied under the following headings: (1) Initialization procedures. These include the procedures to set up the sentence table, to set up the real-time interrupt clock, to set initial conditions and allow interrupts. (2) a MAIN body. This includes a procedure NEW-CASE which resets software message flags, two external procedures GET-DISCRETE and GET-VALUE to get the discrete values from parallel port #2 and get YAW RATE, ALTITUDE, IAS, AOA, AY , AIRSPEED and TEMPERATURE from ADC channels. After obtaining those ADC valuds, procedure MAIN which contains many CASE blocks, will select the sentences to be spoken, a procedure BIGMOUTH-SPEAK will copy those sentences from the sentence table into buffers in translation-queue.

A procedure SPEAK is responsible for dumping the sentence buffers in the translation-queue into the output buffer. An external function OUTPUT-BUFFER-EMPTY will check the buffer-full flag of both output buffers. If any of the output buffers are empty, the procedure LOAD-OUTPUT-BUFFER will load a sentence buffer from translation-queue into the output buffer. The way to load a sentence buffer to the output buffer is as following: First find which of the sentence buffers in the translation-queue needs to be dumped, then convert the phoneme and pitch in the sentence buffer to ASCII code, then an external procedure DUMP will dump the phoneme and pitch codes into the output buffer. After the whole sentence is loaded into the output buffer, an external procedure MARK-OUTPUT-BUFFER-FULL will set the buffer-full flag of the output buffer and also wakes up the subprogram SMALLMOUTH. The subprogram SMALLMOUTH will send the phoneme and pitch codes from the output buffer one by one to the SC01 for voice synthesis.

After the procedure SPEAK dumps all the sentences into the output buffer, the program VOICE completes its job in this time frame and then the program goes into an external procedure HALT. The procedure HALT has a loop body. The CPU will keep looping in this loop body until the next timer interrupt comes.

B. Abstract Data Structure

An abstract data type is given as follows:

Abstract Data Type

```
Sentence-buffer = Record
                            Buffer-full flag
                            Phoneme symbol array [ 1 .. 50 ]
                            Pitch symbol array [ 1 .. 50 ]
                  End
```

Translation queue = Array [0 ..2] of sentence buffer.

```
Output-Buffer = Record
          Buffer-full flag
          ASCII phoneme code [ 1 .. 50 ]
          Pitch code [ 1 .. 50 ]
      End
```

Abstract Program Structure

```
Declare

    VOICE ( Ø ) ——> OUTPUTBUFFER
    SMALLMOUTH ( OUTPUTBUFFER ) ——> Voice generation

Begin

    When timer-interrupt comes
         do VOICE
    When parallel port interrupt comes
         do SMALLMOUTH

End
```

Program VOICE

```
Begin
    INITIALIZATION;
        Repeat
            MAIN (* Get ADC value, decide sentences *)
            SPEAK (* Dump sentence to output buffer *)
            HALT (* Wait for next timer interrupt *)
        Until FOREVER (* or turned off *)
End
```

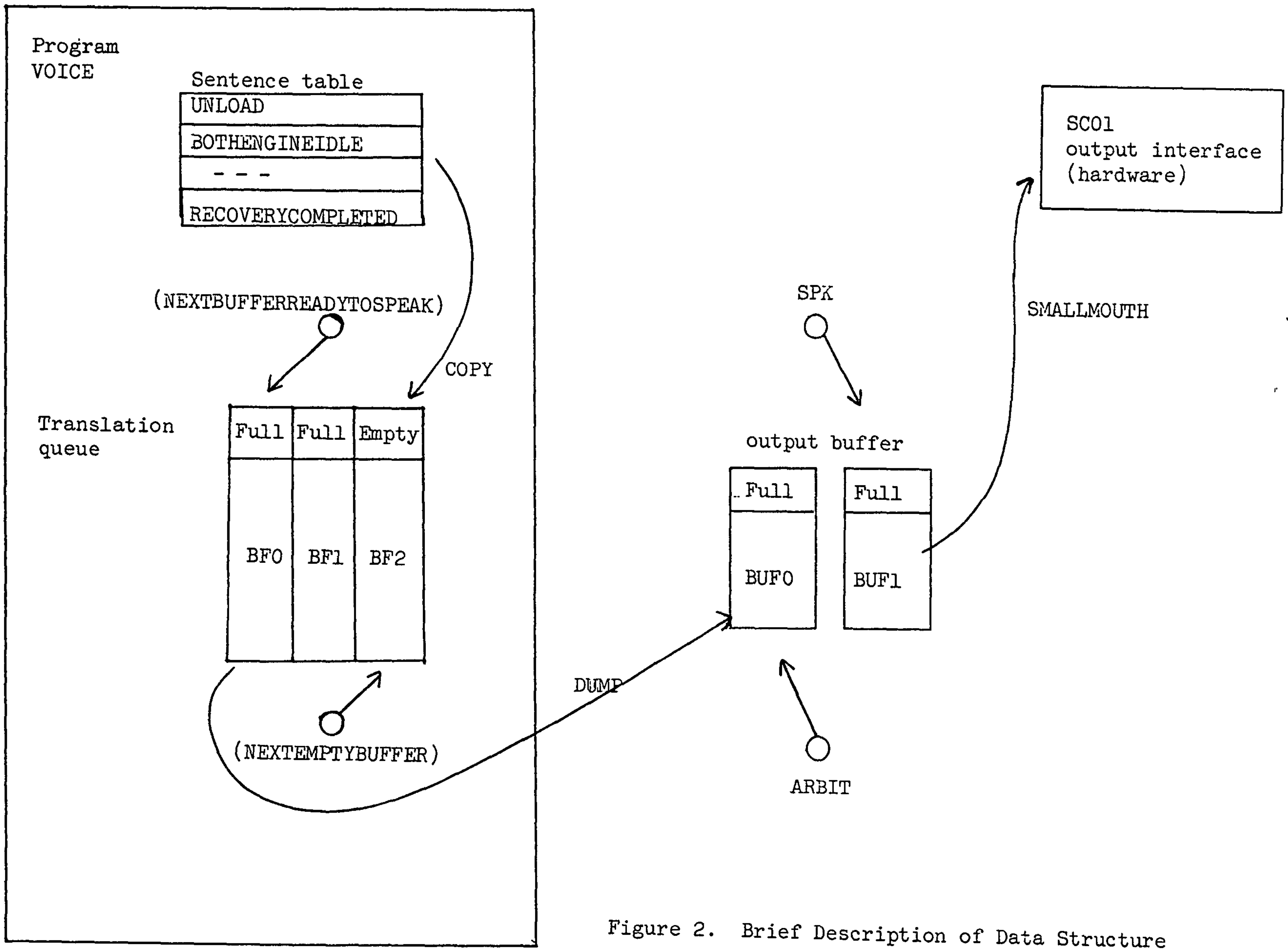


Figure 2. Brief Description of Data Structure

Subprogram SMALLMOUTH

```
Begin

     Declare Waked up
     if output buffer is empty
          then return interrupt
          else
               send out pitch code
               send out phoneme code
               increase pointer
     endif

End
```

C. Implementation

The first level of the abstract data structure implementation contains four procedures: INITIALIZATION, MAIN, SPEAK, and HALT. The process SMALLMOUTH is also implemented. The SMALLMOUTH contains a set of instruction to send out phoneme and pitch codes to the parallel port for the SC01 to speak. This subprogram is called up by a parallel port interrupt or a trap from VOICE. SMALLMOUTH send out the next phoneme along with pitch data, then returns control to VOICE. Under normal conditions, no warning signal need be generated. VOICE finds nothing to speak, then resets the runningflag and halts. SMALLMOUTH recognizes the situation from the runningflag and halts also. The whole system is in an inactive situation, only the procedure VOICE will be called by the timer every half second. If the situation does not require a voice message, the system becomes inactive again.

A more detailed description of the different levels of implementation is shown in Figure 2 and the attached program listing.

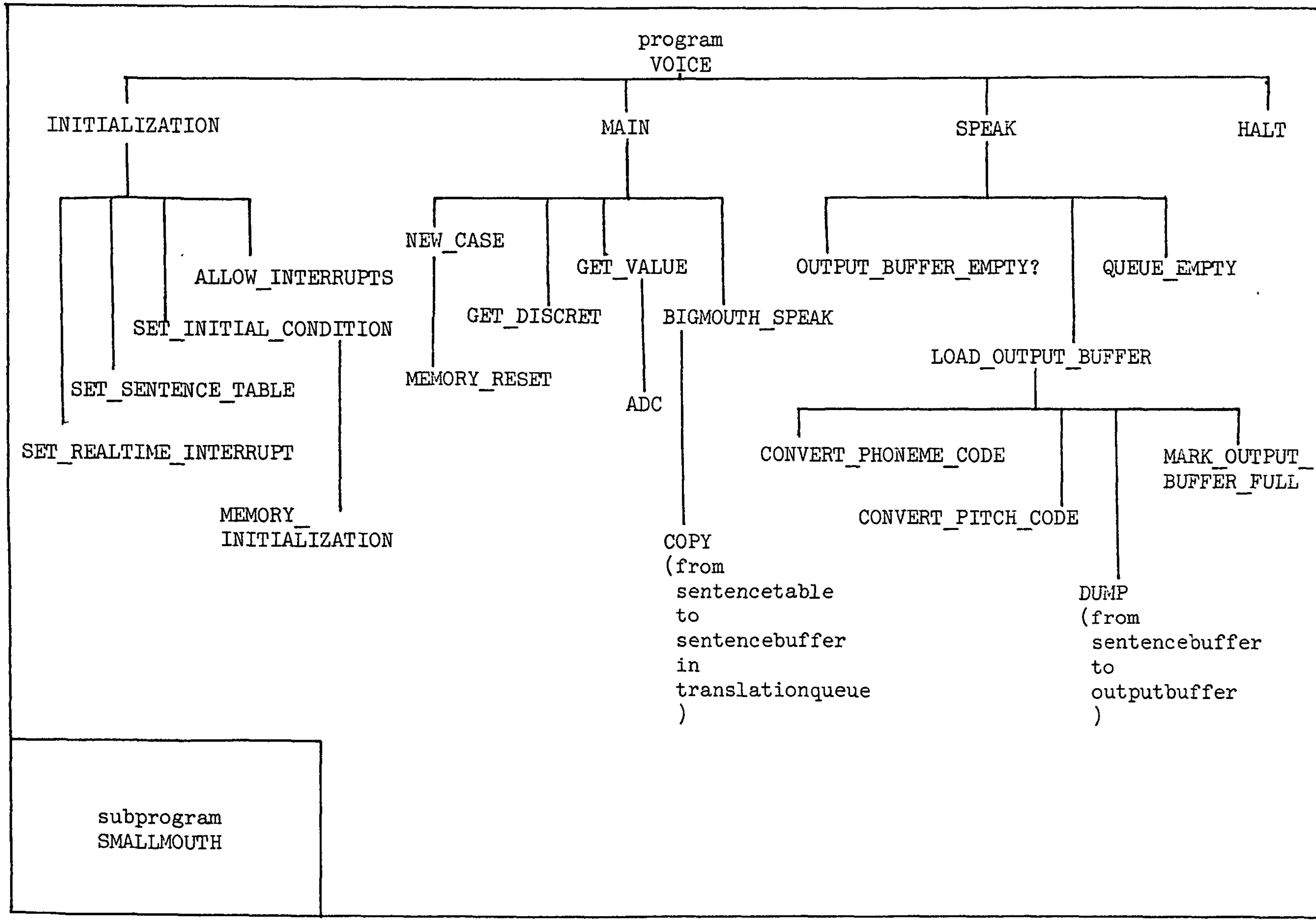


Figure 3. Software Implementation

C-2

D. Detail Explanation of Each Procedure

1. Program VOICE.

The program VOICE will dump the sentence need to be spoken into outputbuffer as follow:

Step 1. a) initialize the timer clock as 0.5 second by calling external procedure SET-REALTIME-INTERRUPT.

b) set a look up sentence table by calling internal procedure SET-SENTENCE-TABLE.

c) set initial condition for variables by calling internal procedure SET-INITIAL-CONDITION.

d) allow interrupts by calling external procedure ALLOW-INTERRUPTS.

Step 2. a) initialize a new case condition when a new timer interrupt comes by calling internal procedure NEW-CASE.

b) get the value of discrete by calling external procedure GET-DISCRETE.

c) get value of YAW RATE, ALTITUDE, IAS, AIRSPEED, AOA, TEMP.
find the sentence and pass this sentence as a parameter to internal procedure BIGMOUTH-SPEAK and load the sentence into a sentence buffer in the translation-queue by calling this procedure.

Step 3. dump all the sentence buffers into output buffer.

Step 4. wait for next timer interrupt.

Step 5. go to step 2.

Step 6. end.

2. Procedure INITIALIZATION

This procedure sets the initial conditions such as sentence table, PTM clock rate, etc. This procedure will be executed only when the system is turned on.

Step 1. set realtime interrupt.

Step 2. set sentence table.

Step 3. set initial condition

Step 4. allow interrupts.

3. Procedure SET-REALTIME-INTERRUPT

This routine sets the cascading timeout interrupt of the MC6840 PTM, connect 03 to C2 and get realtime interrupt from timer #2. It also sets the realtime interrupt clock from clock #1 of PTM. The timer will wake the procedure MAIN every half second, get ADC value, and decide what sentences are required to speak.

4. Procedure SET-SENTENCE-TABLE

This procedure sets up a sentence table where all phoneme and pitch symbol are stored in 14 SOUND-BUFFER as a look up table.

5. Procedure SET-INITIAL-CONDITION

This routine sets the initial condition for variables TEMP-COUNT, COUNT-30, COUNT-40, and flags such as WARNINGFLAG, FOUND, etc.

Step 1. reset all software synchronization flags

Step 2. call external procedure MEMORYINITIALIZATION to initialize the variables and flags in assembly part of the programs.

6. Procedure ALLOW-INTERRUPTS

When power is turned on, all interrupts are disabled. This assembly routine store a $0480 in the control register of M68K000 versa system which will enable the timer and parrallel port interrupt. Also, this routine store autovector addresses for timer interrupt, parrallel port interrupt, and trap #6.

7. Procedure MEMORYINITIALIZATION

This routine is called by the procedure of SET-INITIAL-CONDITION to reserve block space in memory for output buffers. It also clear buffer-full flags of output buffers, set ARBIT point to the buffer #0, declare that the subprogram SMALLMOUTH in not running.

Step 1.
- a) ARBIT: reserve one byte for ARBIT, ARBIT point to the output buffer is dumping.
- b) SLEEP: reserve one byte for SLEEP, SLEEP flag to indicate the SMALLMOUTH is not active.
- c) BFOFUL: reserve one byte for BFOFUL, BFOFUL flag to indicate that buffer 0 is full.
- d) BF1FUL: reserve one byte for BF1FUL, BF1FUL flag to indicate that buffer 1 is full.
- e) BUF0: reserve 50 word to store phoneme and pitch codes.
- f) BUF1: reserve 50 word to store phoneme and pitch codes.
- g) QUEPTR: reserve one byte for QUEPTR, QUEPTR point to the offset in output-buffer is speaking.

Step 2. ARBIT #0 : set ARBIT to point to output-buffer #0.

Step 3. SPK #0 : set speak point to output buffer #0.

Step 4. QUEPTR #0: point to the first word in output buffer.

Step 5. SLEEP #1: declare that the SMALLMOUTH is not active.

Step 6. BFOFUL #0: reset buffer-full flag of output buffer #0

Step 7. BF1FUL #0: reset buffer-full flag of output buffer #1.

Step 8. End.

8. Procedure MAIN

The procedure MAIN will get input conditions such as YAWRATE, AOA, TEMPERATURE, IAS, AIRSPEED, AY, ALTITUDE, from ADC channels and decide which sentences need to be spoken. Then copy the sentences from the sentence table into buffers in translation-queue. Another procedure SPEAK will move the sentences in these buffers into an outbuffer which is located at some relocatable address in memory. Then the subprogram SMALLMOUTH will move each phoneme and pitch codes to the SC01.

Step 1. declare new case.

Step 2. get DISCRETE.

Step 3. get value of

a. IAS.
b. AIRSPEED.
c. AOA.
d. AY.
e. TEMPERATURE.
f. YAWRATE.
g. ALTITUDE.

Step 4. determine YAWRATE is positive or negative.

Step 5. if TEMPERATURE > 1215°C for 3 second , BIGMOUTH speak 'LEFT OR RIGHT ENG OFF'.

Step 6. if AOA < -10° and AY > .1g, BIGMOUTH speak ' STICK HALF AFT AND HOLD'.
then if AOA>0 for 5 second, BIGMOUTH speak ' RECOVERY COMPLETE'.

Step 7. if warning flag = true
then begin if YAWRATE < 40°/sec for 2 seconds, BIGMOUTH speak 'CENTER CONTROLS'.
if YAWRATE < 30°/sec for 2 seconds, and AIRSPEED > 120 KTS, BIGMOUTH speak 'RECOVERY COMPLETE'.
end

Step 8. if any condition in Table 1 occured, then BIGMOUTH speak sentences according Table and set warning flag.

9. Procedure NEW CASE

This procedure initialize a new case condition when a new timer interrupt comes.

Step 1. reset all buffer-full flag in three sentence table of translation queue.

Step 2. call procedure MEMORY-RESET to initialize a new case condition in assembly part, also clear interrupt flags.

10. Procedure MEMORY-RESET

This procedure will be merged in the procedure of NEW-CASE to reset the initial condition and forget the old sentence in the buffer when a new case happens. It first clears the interrupt request flag and then allows all timer to operation mode by writting #0 in control register 1 (CR20 = 0). Next, it clears PTM status register by clearing interrupt request flag. It continue by reading timer #2 counter, then by testing ARBIT which is pointer to one of the output buffers, then it clears the output buffer which ARBIT is pointing to. It returns after reseting port interrupt flag.

11. Function GET-DISCRETE: integer

This function get the value of discrete from parallel port and return the value as an integer.

12. Function GET-VALUE

The function GET-VALUE converts the variable name to channel number, gets the ADC value from the forward function ADC, then returns the channel value to the specified variable.

Step 1. convert variable name to channel number.

Step 2. get ADC value from the according channel.

13. Function ADC

The function ADC will return the ADC value from the specified channel.

Step 1. set address of base, control register, MSB and LSB.

Step 2. load control register to slect channel number.

Step 3. initial conversion.

Step 4. wait until ready.

Step 5. move MSB to data register #0.

Step 6. shift data register #0 left 8 bit.

Step 7. move LSB to data register #0 .

Step 8. arithmetic shift data register #0 right 4 bit to get 2's implement value.

Step 9. store data register #0 to functional value return address.

Step 10. return.

14. Procedure BIGMOUTH-SPEAK

This procedure gets a sentence name from the procedure MAIN, then copies the sentence from sentence table into a sentence buffer in translation queue.

Step 1. copy sentence from sentence table to sentence buffer which is pointed by NEXTEMPTYBUFFER.

Step 2. set NEXTEMPTYBUFFER point to the next sentence buffer.

15. Procedure COPY

This procedure copies each phoneme and pitch symbol from one sentence is sentence table to a sentence buffer in translation-queue. After all phoneme and pitch symbol are copied, the buffer-full flag of the sentence buffer is set.

16. Procedure SPEAK

This procedure sends out sentences from translation queue to output buffer whenever the output buffer is empty. After all sentences have been sent out, the queue is empty.

Step 1. find if any one output buffer is empty.

Step 2. then load output buffer

Step 3. if translation queue is not empty, go Step 1.

17. Function OUTPUT-BUFFER-EMPTY: boolean

This function responds with a boolean value to indicate if there is an empty output buffer. If it finds an empty output buffer, it returns a function value of true. This function starts by finding if ARBIT points to BUF1 or BUF0. The next step is to find out if the buffer-full flag of either BUF0 or BUF1 is set. If it is not set then it returns the value of true.

18. Procedure LOAD-OUTPUT-BUFFER

This procedure dumps the sentence from the translation-queue into output buffer, each phoneme and pitch symbol will be converted into ASCII code and then dumped into the output buffer.

Step 1. find next sentence-buffer ready to speak.

Step 2. a) convert phoneme code.
b) convert pitch code.

Step 3. dump to output buffer.

Step 4. mark output buffer full.

Step 5. reset the buffer-full flag of the sentence buffer.

19. Function CONVERT-PHONEME-CODE

This function convert phoneme symbols into ASCII code before dumping them into the output buffer.

20. Function CONVERT-PITCH-CODE

This function converts pitch codes from decimal to octal before the pitch is dumped into output buffer. The first digit in decimal form will send to SC01 for pitch control. The next two digits will be used as frequency control for the SC01 clock input.

21. Procedure DUMP

This procedure dumps pitch and phoneme codes into the empty output buffer at the offset location INT. The pitch code is dumped in the lower byte, the phoneme is dumped in the higher byte of the word. The procedure starts by getting the offset INT, pitch code and phoneme code. Then it test ARBIT to see which buffer it is pointing to. If it is pointing to BUF1 then load buffer #1 and if it is pointing to buffer #0 then load buffer #0.

Step 1. a) get offset INT.
b) get pitch code
c) get phoneme code.

Step 2. test ARBIT.

Step 3. if ARBIT = 1, goto step 5.

Step 4. load base address of BUF0 to register A2, goto step 6.

Step 5. load base address of BUF1 to register A2.

Step 6. a) move pitch code to buffer.
b) move phoneme code to buffer.

Step 7. end.

22. Procedure MARK-OUTPUT-BUFFER-FULL

This procedure is called at end of the procedure of LOAD-OUTPUT-BUFFER when the whole sentence in translation buffer have been dumped into output buffer. The buffer-full flag of the output buffer which is pointed by the ARBIT will be set. Then, this procedure will check the flag SLEEP to determine that if the subprogram SMALLMOUTH is in active or not. If the SLEEP flag is set, go trap #6 to wake the subprogram which will send phoneme and pitch codes to parallel port #1.

23. Function QUEUE-EMPTY: boolean

This function checks the three sentence buffers in the translation-queue. If all buffers in the queue are empty, it returns a true variable.

24. Procedure HALT

After execution is completed, this procedure is called in the main program that cause the main program to become idle. It first resets the interrupt request flag to wait for next timer interrupt. Then, it keep checking to see if there is an interrupt from the timer and then it exits after timer interrupt and it also sets the flag after the timer interrupt arrives.

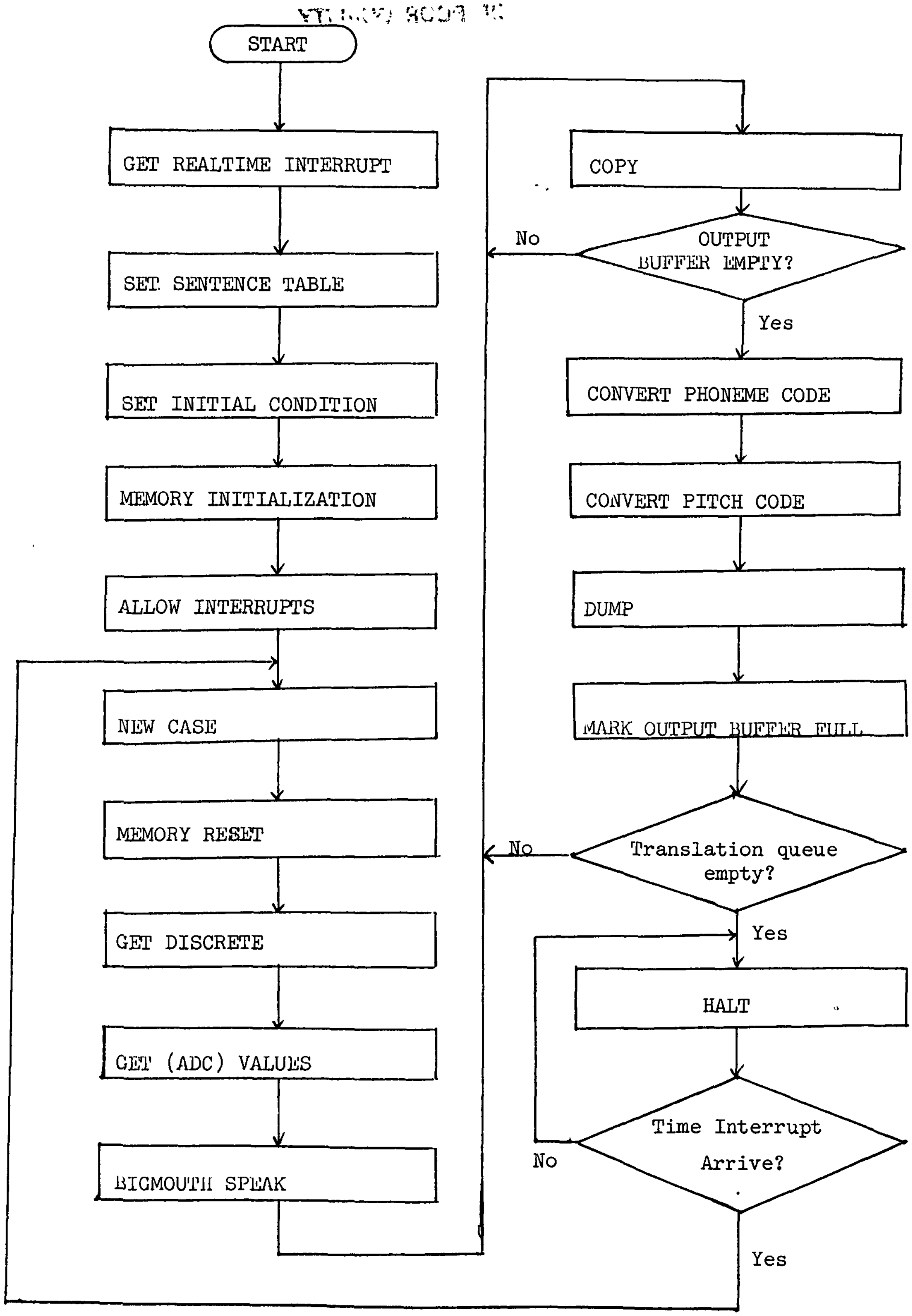
Step 1. reset flag to wait for timer interrupt.
Step 2. test'timer has interrupted ?'
Step 3. if timer has not interrupted go to step 2.
Step 4. a) exit after timer interrupt.
b) set flag.
Step 5. return from interrupt.
Step 6. end.

25. Subprogram SMALLMOUTH

This subprogram is initialized by parallel port interrupt or trap #6. The parallel port #1 is used to output phoneme and pitch codes. When a phoneme is completely spoken by the SC01, the SC01 returns an interrupt request via the lower byte of #1 parallel port. It will initialize this subprogram to send out the next phoneme and pitch. Another chance to execute this subprogram is after a sentence is dumped into an output buffer, the procedure MARK-OUTPUT-BUFFER-FULL will check the flag SLEEP and generate a #6 trap which autovector address is the starting address of SMALLMOUTH. After a sentence is completely sent out from output buffer to SC01, the flag SPK is changed to indicate that another output buffer is the next to be send out.

E. Flowchart

ORIGINAL PAGE IS
OF POOR QUALITY



F. Code Generation

The software including all assembly and pascal routine, was developed and debugged on the TOP20 system. Next, all the assembly and pascal routines were transferred to IBM 4341 system by operator of the USC Engineering Computer Lab. The purpose of transferring our software from TOP20 to IBM 4341 system was to sue the MC68000 support software package on IBM system to get the object codes for all assembly and pascal routine and link them together.

a. MC68000 Support Software Package.

This package can be used as a powerful software development tool for MC68000 based system. The package utilities enable the production of relocatable machine code for the Motorola MC68000 microprocessor. The source code can be written in pascal or the assembly language of the microprocessor. A linkage editor can create execution modules in which library functions can be selectively included. The package provides libraries for floating point operations as well as for runtime routines for three popular MC68000 based configurations including versamodule, the one used for this project.

The support software is designed to operate in a host system. In this case, the host is an IBM 4341 called VIRGIL.

VIRGIL is part of the USC Engineering Computer Lab. It has 8 mega bytes of main memory and it runs under CMS (conversational machine system).

b. Pascal Cross Compiler

Processes pascal programs and produces relocatable code that can be subsequently linked with other modules. The compiler works in two stages. The first one, phase 1, checks the syntax and semantics of the source code and produces and intemediate pcode file. The second stage, phase 2, processes this intermediate code and produces a relocatable and position independent object code module.

c. M68000 Cross Macro Assembler

It processes 68000 assembly language files and produces object code files that can be subsequently linked with other files.

After using the linker to produce the complete and final object code package, the code was down loaded to the lab's computerm computer (an Intel 8085 based CP/M microcomputer) Finally, the code is dumped into the 68000 for execution.

III. Hardware Development

A. MC68000 System:

The MC68000 monoboard system with the PTM has the following capability: (1) Vector Interrupt Handler. (2) Programmable Timer module which enables the real time interrupt and adjustable pitch control for SC01 Voice Synthesizer. (3) The processor instruction set provides software interlocks for processes to interleave the CPU. Trap instruction and TAS (test and set) are useful for multiprocess communication that support modern structured programming techniques.

B. Input Interface

Yawrate, Altitude, IAS, AY, Airspeed, AOA and Temperature are transfered via the M68MM15A 16 channel high-level A/D module. The 68000 processor also gets the discrete signals from the #2 parallel port without handshaking.

C. Output Interface

Phoneme and pitch codes (two bytes) are transfered from the 68000 system to the SC01 Voice Synthesizer utilizing the 16bit capability of the parallel port. Design and layout of both the discrete fetch and Voice Synthesizer was completed as the attached schematic figure. The driver LS244 receives the phoneme code from lower byte of parallel port #1. These phoneme lines will be pulled up by a 4.7K resistor array, then sent to the CMOS SC01 device and to the displays. A strobe signal from pin P1CA2 of the parallel port interface will initiate the voice conversion. After the phoneme has completed, the SC01 will request the next phoneme code via P1CA1 vector interrupt request. MC68C00 system also provide a 2Mhz clock from PTM #3 output line to the 7497 Binary Rate Multiplier for pitch control. A 8-bit pitch control code will be provided by the upper byte of parallel port #1, which will be latched by LS374 D-type transparent latches. Two MSB bits are sent to the SC01 for direct pitch control. The LSB 6 bits sent to the Binary Rate Multiplier will be able to adjust the SC01 input clock rate from 30K hz up to 2M hz which will provide a minor adjustment of pitch control. An amplifier LM386 will provide audio amplification to drive a speaker. A 10K ohm potentiometer provides volume control.

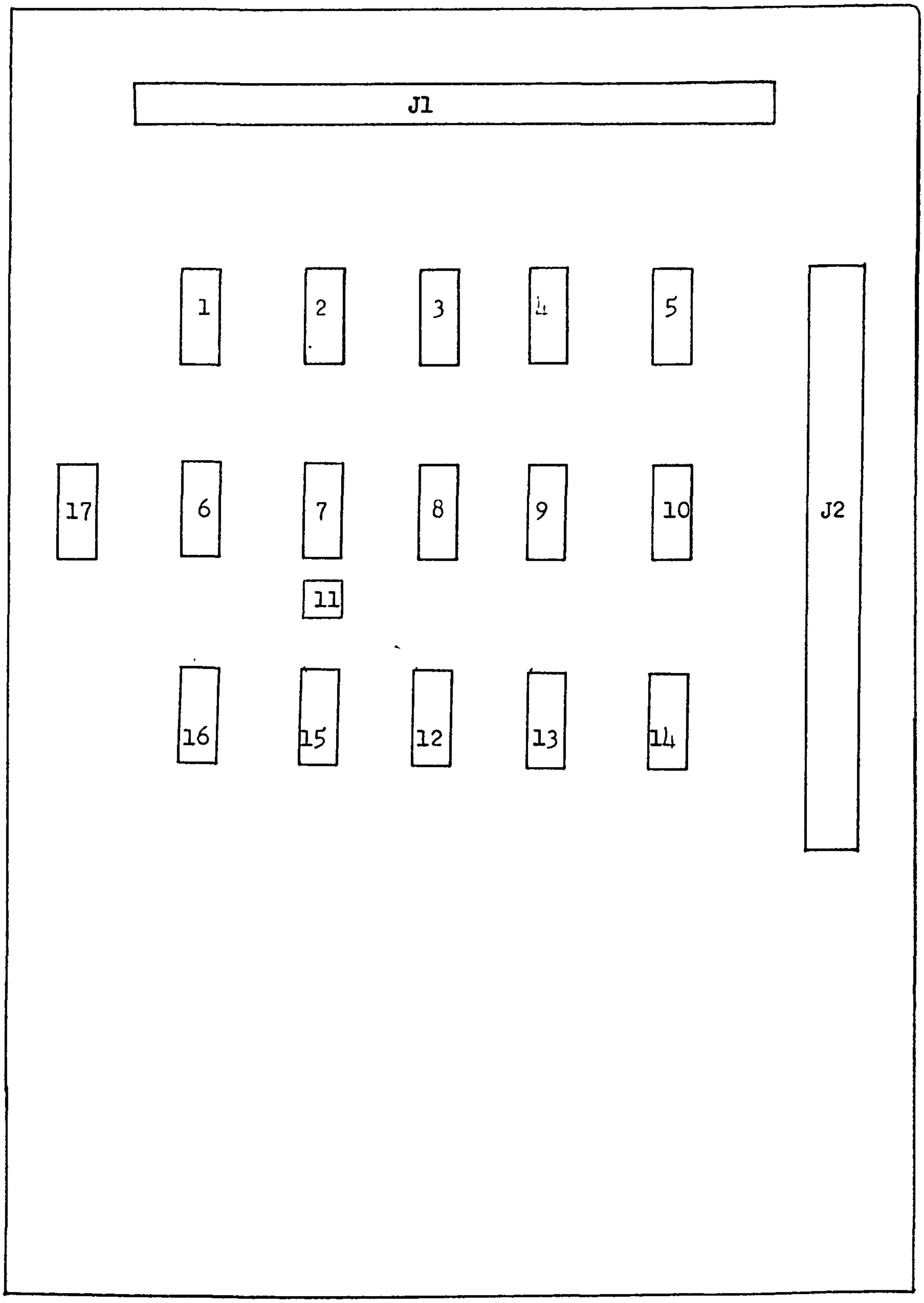


Figure 4. Board Layout

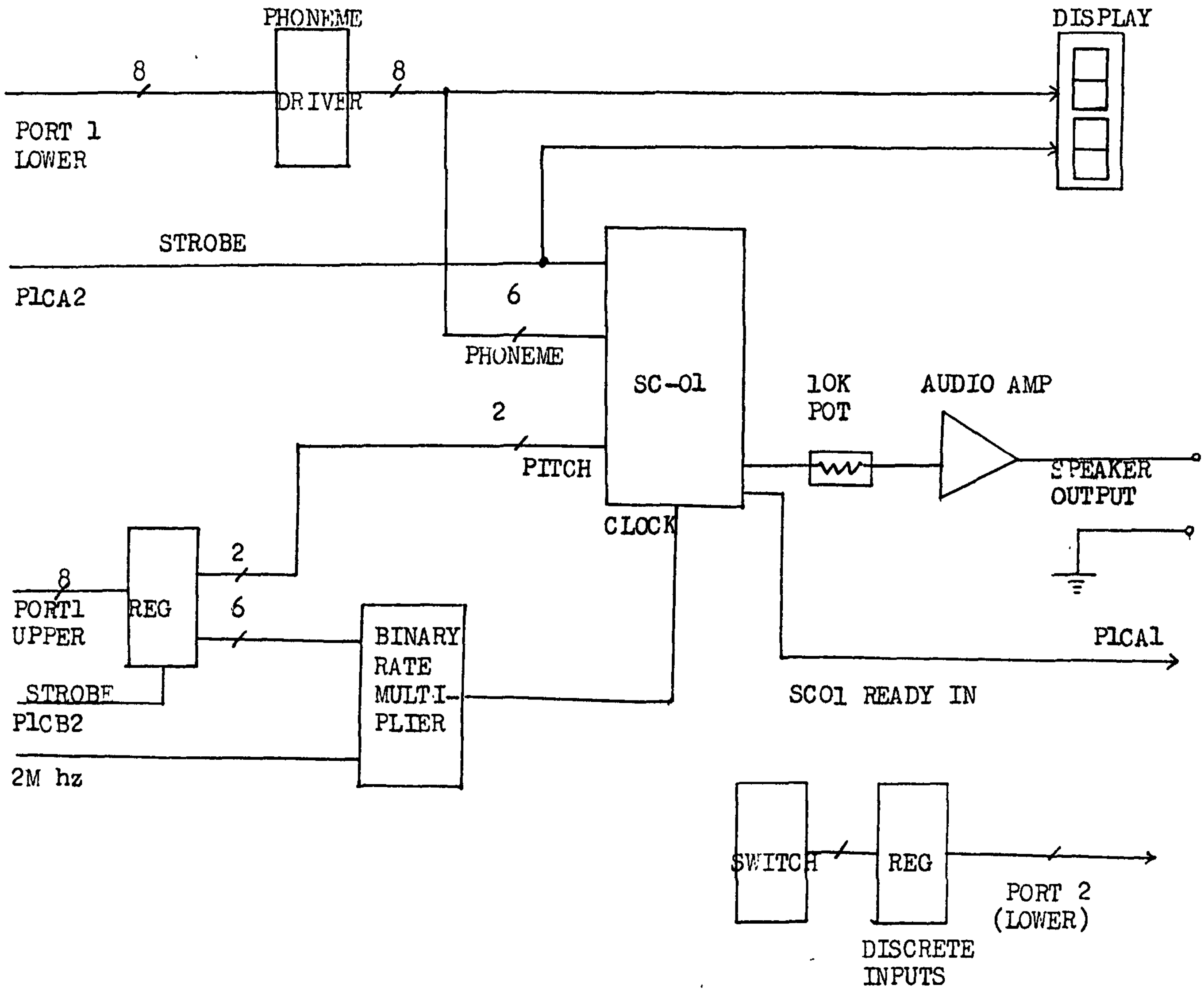


Figure 5. External Hardware Block Diagram

IV. Results / Conclusions

A. Test Setup

For test purposes, a modification of the routine to obtain data from the ADC board was made. Instead, switches on the external hardware board were used to simulate certain combinations of aircraft parameters. The switch pattern was input through the discrete data input port, and the system responds to this input.

To initiate the system testing, a small assembly language test routine was used to test out the interface between the 68000 and the external hardware board. This routine sent the word "unload" continuously to the SC01 display devices, and provide valuable in trouble shooting the interface.

B. Results / Problems Encountered

Successful operation of the system was never achieved. The last of a series of problems encountered was the inability of the linkage editor to provide the start address of the loaded object code.

We were able to get the small test routine to get the system to speak "unload" for a short time, but then the SC01 failed for no apparent reason.

Work yet to be completed includes determining the start address of the run-time package, system software checkout, voice pitch control adjustments, and system integration testing with sensors providing the ADC inputs.

Votrax

A Division of Federal Screw Works
500 Stephenson Highway
Troy, Michigan 48084

SC-01 SPEECH SYNTHESIZER DATA SHEET

Votrax® CMOS Phoneme Speech Synthesizer

GENERAL DESCRIPTION

The SC-01 Speech Synthesizer is a completely self-contained solid state device. This single chip phonetically synthesizes continuous speech, of unlimited vocabulary, from low data rate inputs. Figure 1.

Speech is synthesized by combining phonemes (the building blocks of speech) in the appropriate sequence. The SC-01 Speech Synthesizer contains 64 different phonemes which are accessed by a 6-bit code. It is the proper sequential combination of these phoneme codes that creates continuous speech.

The SC-01 Speech Synthesizer is cost-effective, consumes minimal power and enables in-house product development without vendor dependency. Signals from the SC-01 are applied to an audio output device to amplify and distribute the synthesized speech. See Figure 2.

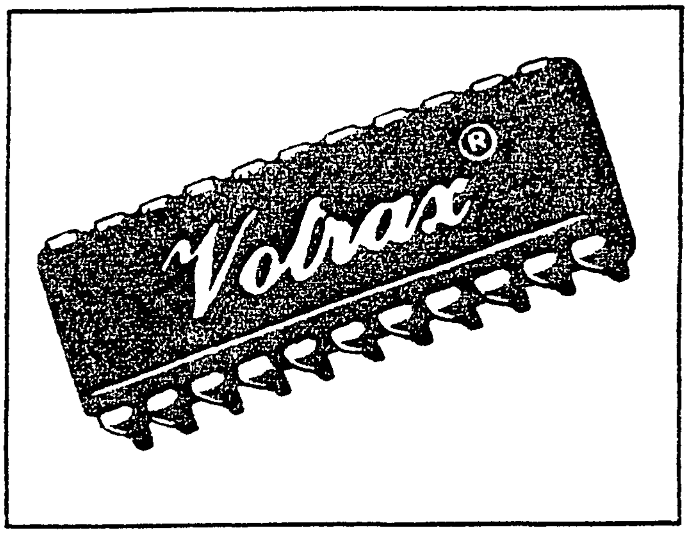


Figure 1. Votrax® SC-01 Speech Synthesizer

FEATURES

- Single CMOS chip
- 70 bits per second
- 22 pin package
- 9 ma. current drain
- Wide voltage supply range
- Latched 5V. compatible inputs
- Digital pitch level inputs
- Automatic inflection
- On-chip master clock circuit
- Optional external master clock
- Variety of voice effects
- Sound effects
- Customer product security

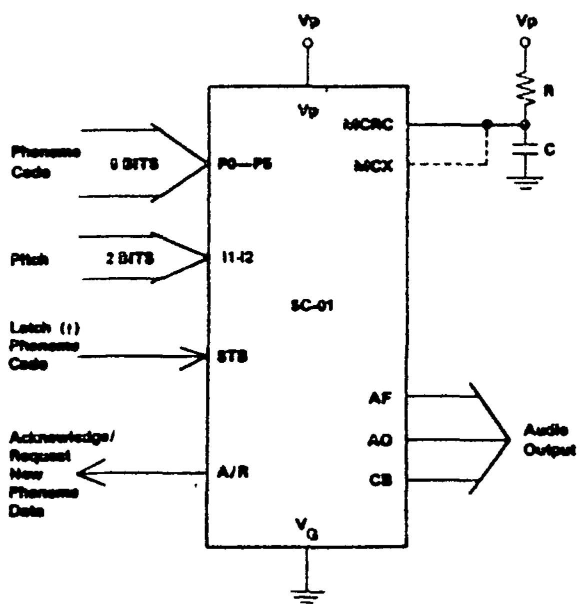


Figure 2. SC-01 Flow Diagram

Votrax® reserves the right to alter its product line at any time, or change specifications or design without notice and without obligation.

PHYSICAL DESCRIPTION

The SC-01 Speech Synthesizer is a 22 pin Large Scale Integrated Circuit which contains all the circuitry necessary to generate phonetically synthesized speech. The SC-01 is fabricated using CMOS technology, which offers high input impedance and low power drain

ELECTRICAL DESCRIPTION

The SC-01 Speech Synthesizer is a program-compatible with existing Votrax® phoneme synthesizers. It requires 70 bits of data per second for continuous speech production. The 6-bit phoneme codes are 5 volt logic compatible and are latched for data bus applications A phoneme-construction algorithm and filters, within the chip, create the synthesized audio output.

ORIGINAL PAGE IS OF POOR QUALITY

PHONEME DESCRIPTION

Table 1 lists the 64 phonemes produced by the SC-01. Each phoneme code is accompanied by its symbol, average duration time, and an example. The underlined segments of the example word demonstrate the phoneme use, i e., sound to be pronounced.

Table 2 subdivides the 64 phoneme symbols into seven categories. Each category represents a different production feature. The first six categories are characterized by voiced, fricative (expired voice), and nasal sounds. The seventh category is characterized by phonemes with no sound output

PHONEME PROGRAMMING

Manual Operations: Votrax® maintains a library of phonetically programmed words. Reference to this library and programming manuals will aid in word synthesis.

Automatic Operations: Votrax® can supply a micro-computer system for automatic conversion of English text into phoneme sequences. This system is particularly useful for in-house vocabulary development and product security Contact Votrax® for further information.

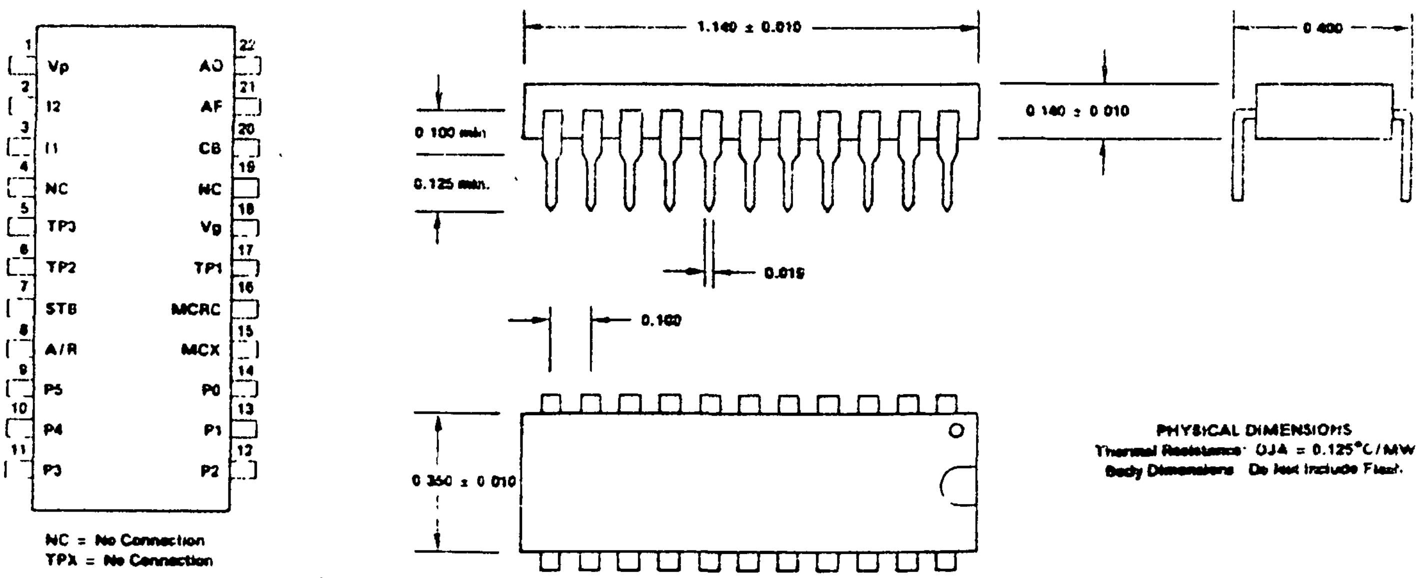


Figure 3 SC-01 Footprint and Outline Dimensions

Votrax® reserves the right to alter its product line at any time, or change specifications or design without notice and without obligation

Table 1 Phoneme Chart

Phoneme Code	Phoneme Symbol	Duration (ms)	Example Word	Phoneme Code	Phoneme Symbol	Duration (ms)	Example Word
00	EH3	59	jacket	20	A	185	day
01	EH2	71	enlist	21	AY	65	day
02	EH1	121	heavy	22	Y1	80	yard
03	PA0	47	no sound	23	UH3	47	mission
04	DT	47	butter	24	AH	250	mop
05	A2	71	made	25	P	103	past
06	A1	103	made	26	O	185	cold
07	ZH	90	azure	27	I	185	pin
08	AH2	71	honest	28	U	185	move
09	I3	55	inhibit	29	Y	103	any
0A	I2	80	inhibit	2A	T	71	tap
0B	I1	121	inhibit	2B	R	90	red
0C	M	103	mat	2C	E	185	meet
0D	N	80	sun	2D	W	80	win
0E	B	71	bag	2E	AE	185	dad
0F	V	71	van	2F	AE1	103	after
10	CH*	71	chip	30	AW2	90	salty
11	SH	121	shop	31	UH2	71	about
12	Z	71	zoo	32	UH1	103	uncle
13	AW1	146	lawful	33	UH	185	cup
14	NG	121	thing	34	O2	80	for
15	AH1	146	father	35	O1	121	aboard
16	OO1	103	looking	36	IU	59	you
17	OO	185	book	37	U1	90	you
18	L	103	land	38	THV	80	the
19	K	80	trick	39	TH	71	thin
1A	J*	47	judge	3A	ER	146	bird
1B	H	71	hello	3B	EH	185	get
1C	G	71	get	3C	E1	121	be
1D	F	103	fast	3D	AW	250	call
1E	D	55	paid	3E	PA1	185	no sound
1F	S	90	pass	3F	STOP	47	no sound

/T/ must precede /CH/ to produce CH sound.
/D/ must precede /J/ to produce J sound.

Table 2. Phoneme Categories According to Production Features

Voiced					'Voiced' Fricat.	'Voiced' Stop	Fricative Stop	Fricative	Nasal	No Sound
E	EH	AE	UH	OO1	Z	B	T	S	M	PA0
E1	EH1	AE1	UH1	R	ZH	D	DT	SH	N	PA1
Y	EH2	AH	UH2	ER	J	G	K	CH	NG	STOP
Y1	EH3	AH1	UH3	L	V		P	TH		
I	A	AH2	O	IU	THV			F		
I1	A1	AW	O1	U				H		
I2	A2	AW1	O2	U1						
I3	AY	AW2	OO	W						

Votrax® reserves the right to alter its product line at any time, or change specifications or design without notice and without obligation.

SIGNAL DESCRIPTION (See Figures 4 and 5)

Phoneme 6-Bit Selection Code (P0-P5): Data input is to six pins. Latching is controlled by the strobe (STB) signal.

Strobe (STB): Latching occurs on rising edge of strobe signal.

Inflection Level Setting (I1, I2) Instantaneously sets pitch level of voiced phonemes

Acknowledge/Request ($\bar{A}$/R): Acknowledges receipt of phoneme data (signal goes from high to low one master clock cycle following active edge of STB signal). Also indicates timing out of old phoneme concurrent with request for new phoneme data (signal goes from low to high)

NOTE

If external phoneme timing is desired, phoneme requests can be ignored. However, best speech is realized with internal timing.

Master Clock Resistor-Capacitor (MCRC): This input determines the internal master clock frequency. Select R-C values for 720 kHz to achieve standard phoneme timing. Connect this input to MCX when using internal clock, ground when using external clock.

NOTE

Varying clock frequency varies voice and sound effects. As clock frequency decreases, audio frequency decreases and phoneme timing lengthens. Figures 6 and 7 illustrate manual and DAC (Digital to Analog Converter) voice variation schematics, respectively.

Master Clock External (MCX): Allows control by an external clock signal.

NOTE

Ground MCRC during MCX operation.

Audio Output (AO): Supplies analog signal to audio output device.

Audio Feedback (AF): Used with Class A or Class B transistor audio amplifiers for added stability.

Class B (CB): Current source for Class B transistor audio amplifier.

Table 3. Timing Specifications

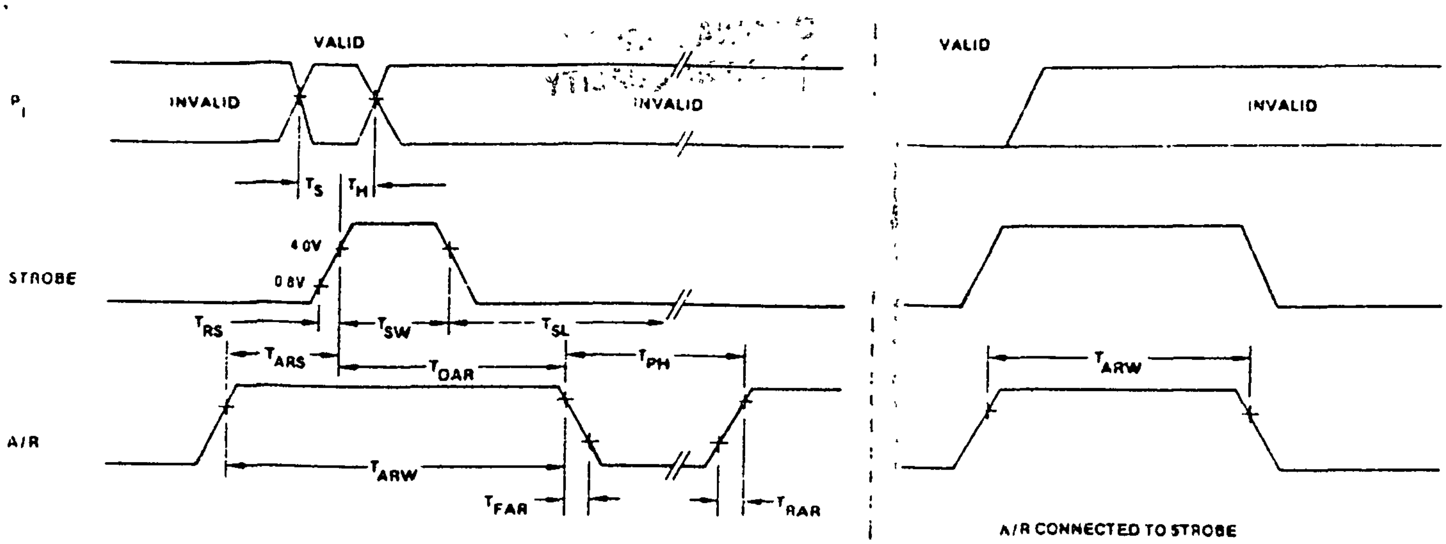
CHARACTERISTIC	SYMBOL	MIN	TYP	MAX	UNIT
Input Setup Time (P_i to STB)	T_S	450			NS
Input Hold Time (P_i to STB)	T_H	0			NS
Rise Time of STB Edge (.8V to 4V)	T_{RS}			100	NS
A/R Width ($\bar{A}$/R Connected to STB) +	T_{ARW}	1	1.3	2	µs
STB Width	T_{SW}	200			NS
STB Low*	T_{SL}	*			NS
Propagation Delay (STB to A/R after T_{ARW})	T_{DAR}			500	NS
A/R Rise Time (Capacitive load = 30pf)	T_{RAR}			100	NS
A/R Fall Time (Capacitive load = 30pf)	T_{FAR}			100	NS
Time from $\bar{A}$/R Request to STB Service)	T_{ARS}	0		500	µs
Time of Phoneme Duration +	T_{PH}	47	107	250	MS

+ Dependent on Master Clock frequency 720kHz

* Strobe must remain low (72x Master Clock Period) before rising edge

Votrax® reserves the right to alter its product line at any time, or change specifications or design without notice and without obligation

B-35



ORIGINAL PAGE IS OF POOR QUALITY

Figure 4. Timing Diagram

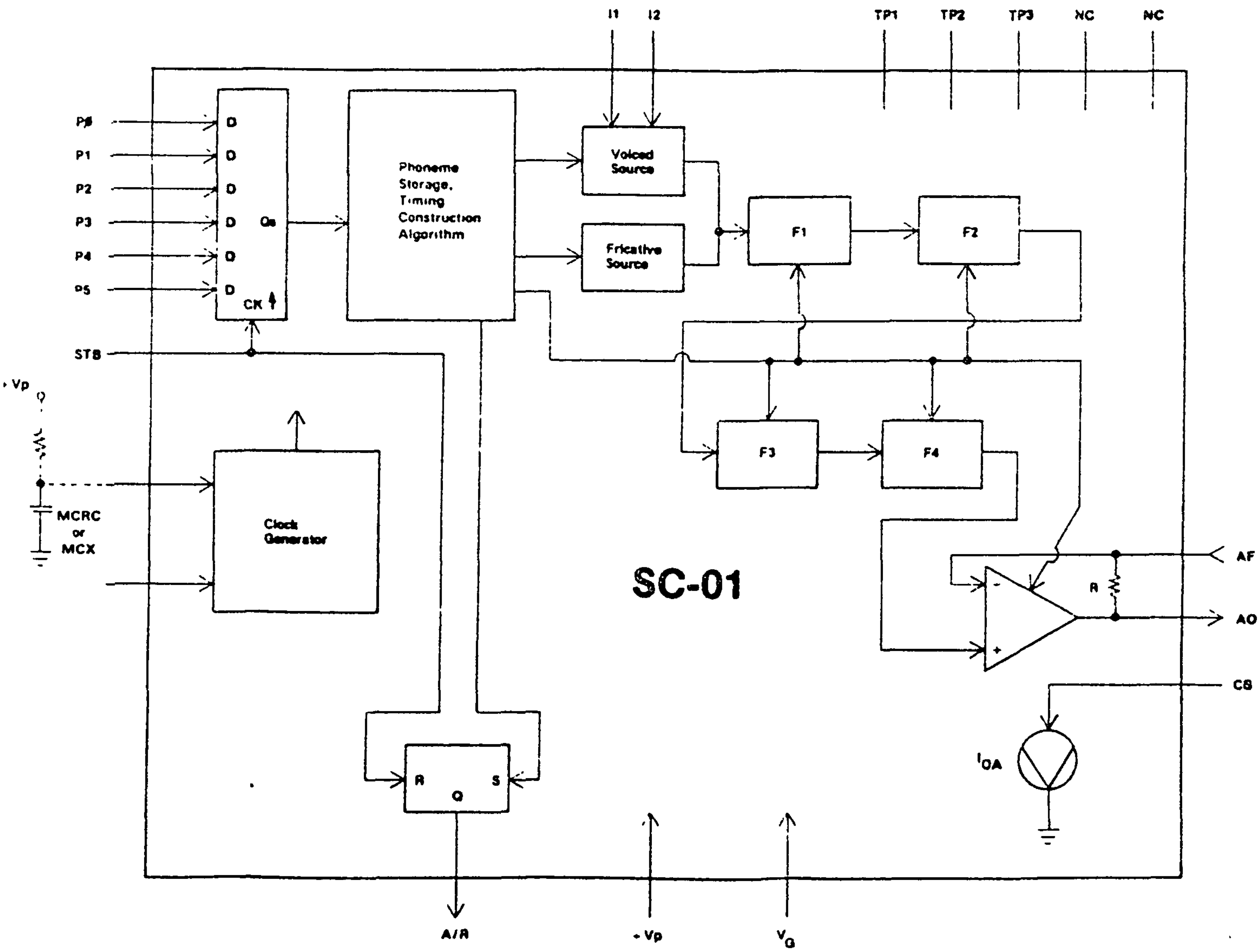


Figure 5. SC-01 Block Diagram

Votrax® reserves the right to alter its product line at any time, or change specifications or design without notice and without obligation.

B-36

ORIGINAL PAGE IS
OF POOR QUALITY

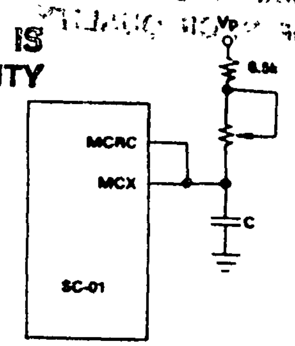


Figure 6. Variable Voice by Potentiometer Control

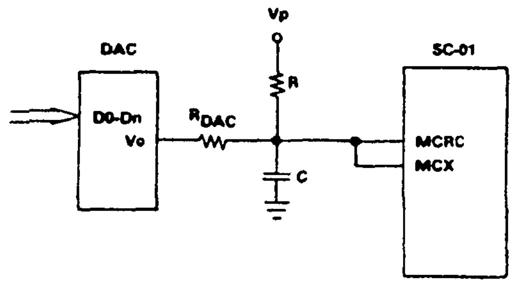


Figure 7. Variable Voice by DAC Current Injection

TYPICAL APPLICATIONS

General: The SC-01 Speech Synthesizer is easily designed into systems ranging in complexity from ROM/counters to microprocessor controllers.

Single Message System: See Figure 8. When the counter is released (START is TRUE), the message is clocked out of the ROM by the A/R signal. The system must be stopped when DONE is TRUE. Note: When using A/R tied to STB, connect a .01 uf capacitor to TP3 to insure power up reset of SC-01.

NOTE

Data at address Ø must be a pause phoneme code.

Multiple Message, Fixed Block Size: See Figure 9. Message address block is loaded into the counter. The message is then clocked out of the ROM by the A/R signal.

NOTE

Message Block = 2^n maximum.

Multiple Message, Variable Block Size: See Figure 10. The microprocessor loads phonemes into a data bus. The A/R signal generates an interrupt request for each new phoneme.

CONNECTING THE AUDIO OUTPUT DEVICE

Audio Output: The AO signal has a maximum peak to peak voltage swing of .26 times Vp, depending upon the phoneme selected, and the AO signal is D.C. biased.

Class A Amplifier: See Figure 11. For a single transistor amplifier, the selection of R, C, or R_s values depends upon the value of Vp and the desired audio level.

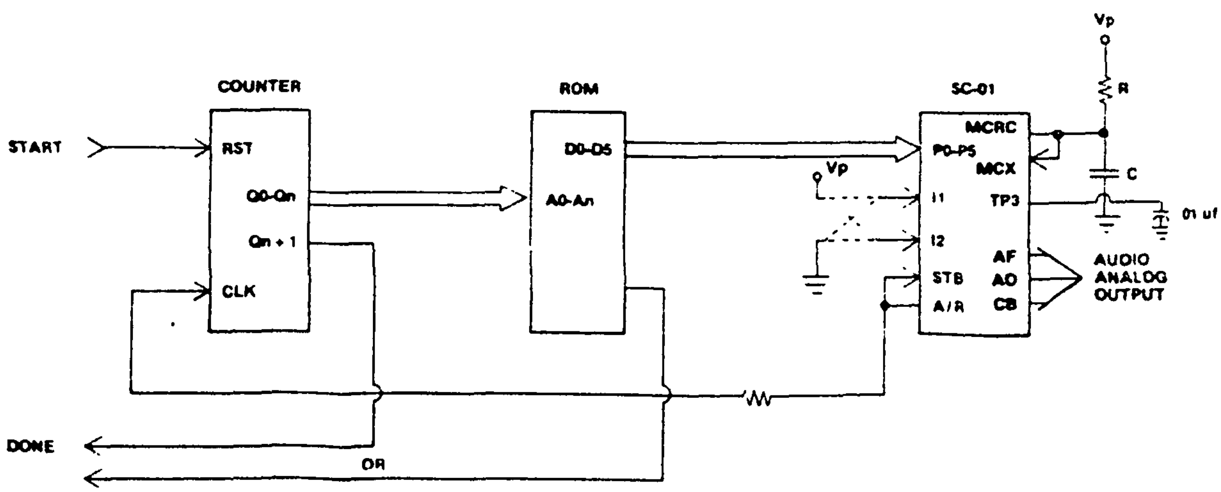


Figure 8. Single Message System

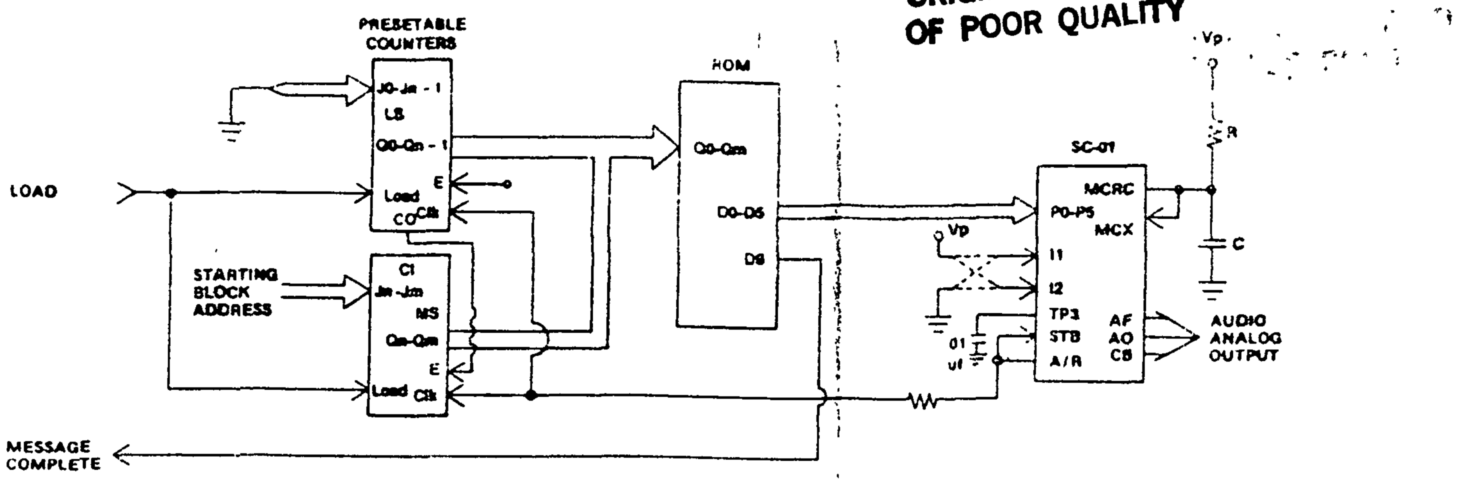


Figure 9. Multiple Message, Fixed Block Size

Class B Amplifier: See Figure 12. A current source (CB) is required for this push-pull amplifier.

NOTE

Minimum power is consumed when speech is inactive. When Vp = +12.0 volts and R_s = 40 ohms, the bias current drain is approximately 3.5 milliamps.

Controlling Audio Output Power: See Figure 13. A resistor or potentiometer from the speaker to ground can be used to control the audio output power.

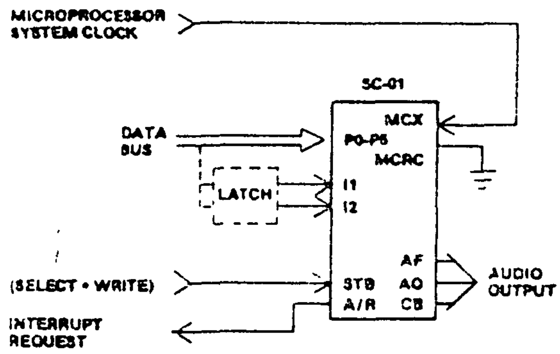


Figure 10. Multiple Message, Variable Block Size

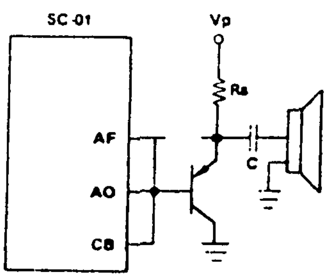


Figure 11. Class A Amplifier

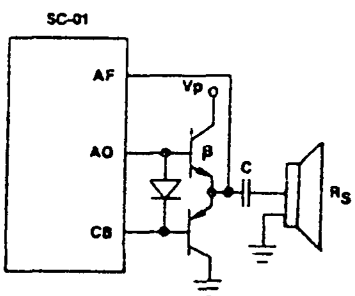


Figure 12. Class B Amplifier*

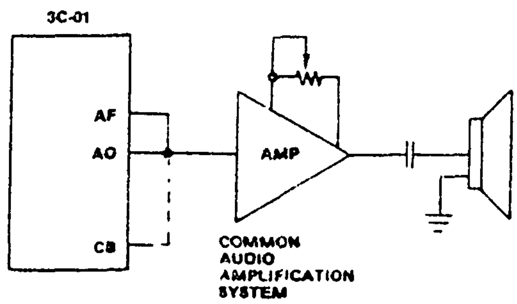


Figure 13. Controlling Audio Output Power

*For Class B Amplifier: (β) x (R_S min.) = 81.6 x (Vp) where β is beta or current gain of transistor. The AO line is protected by an internal series current limiting resistor of 90 ohms maximum. If more current is required of the SC-01, then the above formula indicates distortion will occur.

Table 4. Analog Output Specifications

CHARACTERISTIC	MIN	MAX	UNIT
Output Voltage (AH Phoneme)	.18 x Vp	.26 x Vp	Vp-p
Output Bias Current ** (.6V < CB < Vp)	3.5	7.3	mA

ELECTRICAL CHARACTERISTICS: T_o = Ø to 7Ø°C, Vp = 7 to 14 V_{DC}

CHARACTERISTIC		MIN	TYP	MAX	UNIT
Digital Input Impedance		1 meg			Ohm
Input Capacitance (P_I, STB)				3	pf
Input Capacitance (I1, I2, MCX)				8	pf
Digital Input Logic "Ø" (except I1, I2, MCX)		V_G - Ø.5		V_G + Ø.8	V_{DC}
Digital Input Logic "Ø" (MCX)				V_G + 1.Ø	V_{DC}
Digital Input Logic "Ø" (I1, I2)				.2 x Vp	V_{DC}
Digital Input Logic "1" (except I1, I2, MCX)		V_G + 4.Ø		Vp - Ø.5	V_{DC}
Digital Input Logic "1" (I1, I2)		.8 x Vp			V_{DC}
Digital Input Logic "1" (MCX)		4.6			V_{DC}
Digital Output Logic "Ø" (I sink = Ø.8mA)				V_G+Ø.5	V_{DC}
Digital Output Logic "1" (I source = Ø.5mA)		Vp–Ø.5			V_{DC}
Power Supply Current	Vp = 9V		9.1		mA
	Vp = 9V**		11	18	mA
	Vp = 14V**		18	27	mA
*Master Clock Frequency			720K		Hz
MCX Input Duty Cycle		60:40		40:60	%
Master Clock Resistor Value (MCRC)***		6.5k			Ohm
Master Clock Capacitor Value (MCRC)***				300	pf

**Variable*

***With CB, AF, AO connected for Class B audio amplifier (see APPLICATION NOTES)*

****Frequency of Master Clock* ≃ 1.25 / RC

Note TP1, TP2 must be left open for normal operation.

B-34

Table 5. Absolute Maximum Ratings

ORIGINAL PAGE IS
OF POOR QUALITY

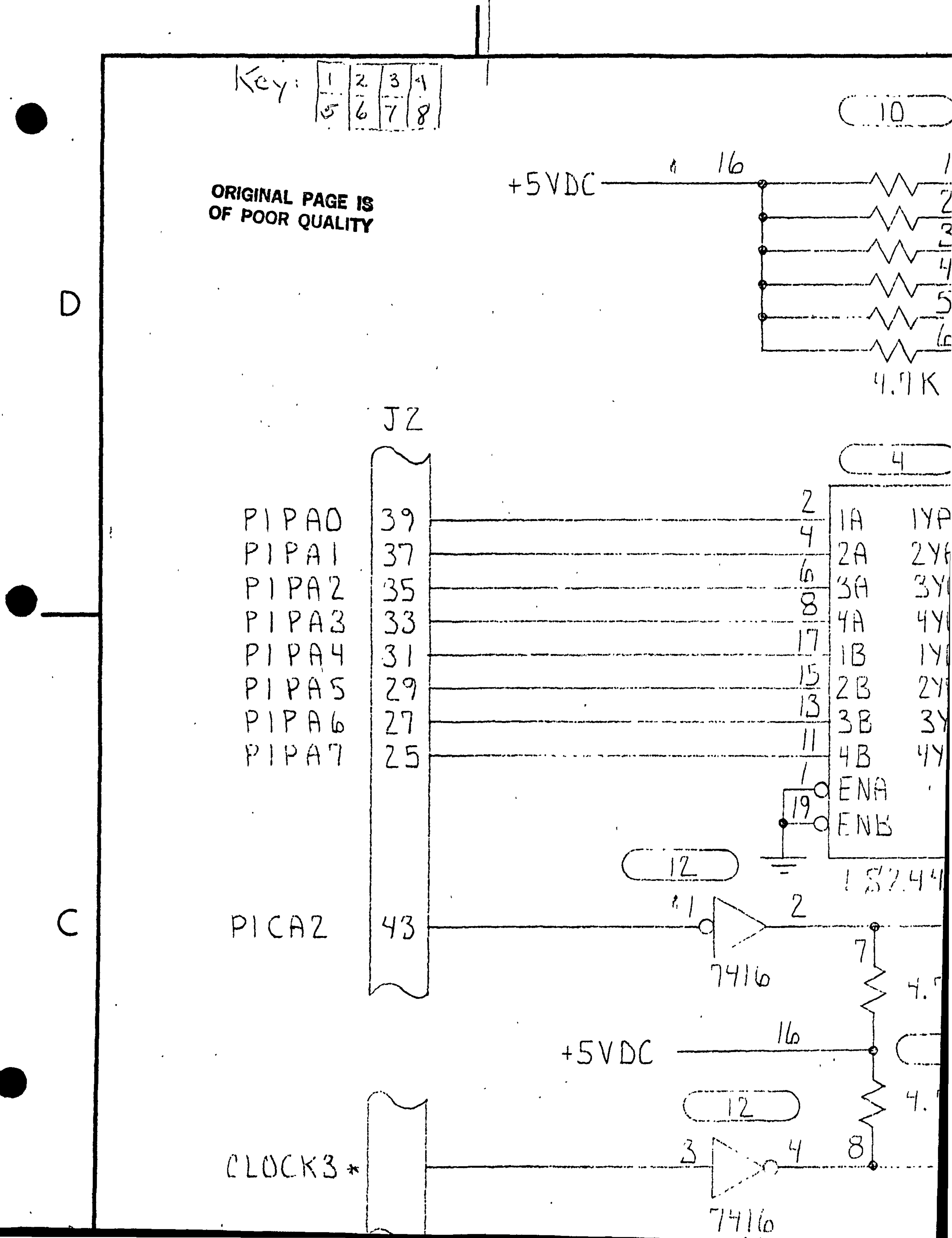
ABSOLUTE MAXIMUM RATINGS *

RATING	SYMBOL	VALUE	UNIT
Power Supply Voltage	V_P	20	V_{DC}
Power Dissipation at 25°C	P_{DM}	650	mW
Derating Above 25°C		5	mW/°C
Operating Ambient Temperature	T_o	0 to 70	°C
Storage Temperature	T_{STG}	-55 to 125	°C
Input Voltage	V_{INM}	-0.5 to Vp+0.5	V_{DC}
DC Current Max. Above Vp+0.5V	I_{INM}	10	ma
Lead Temperature (soldering 10 sec.)	T_L	300	°C

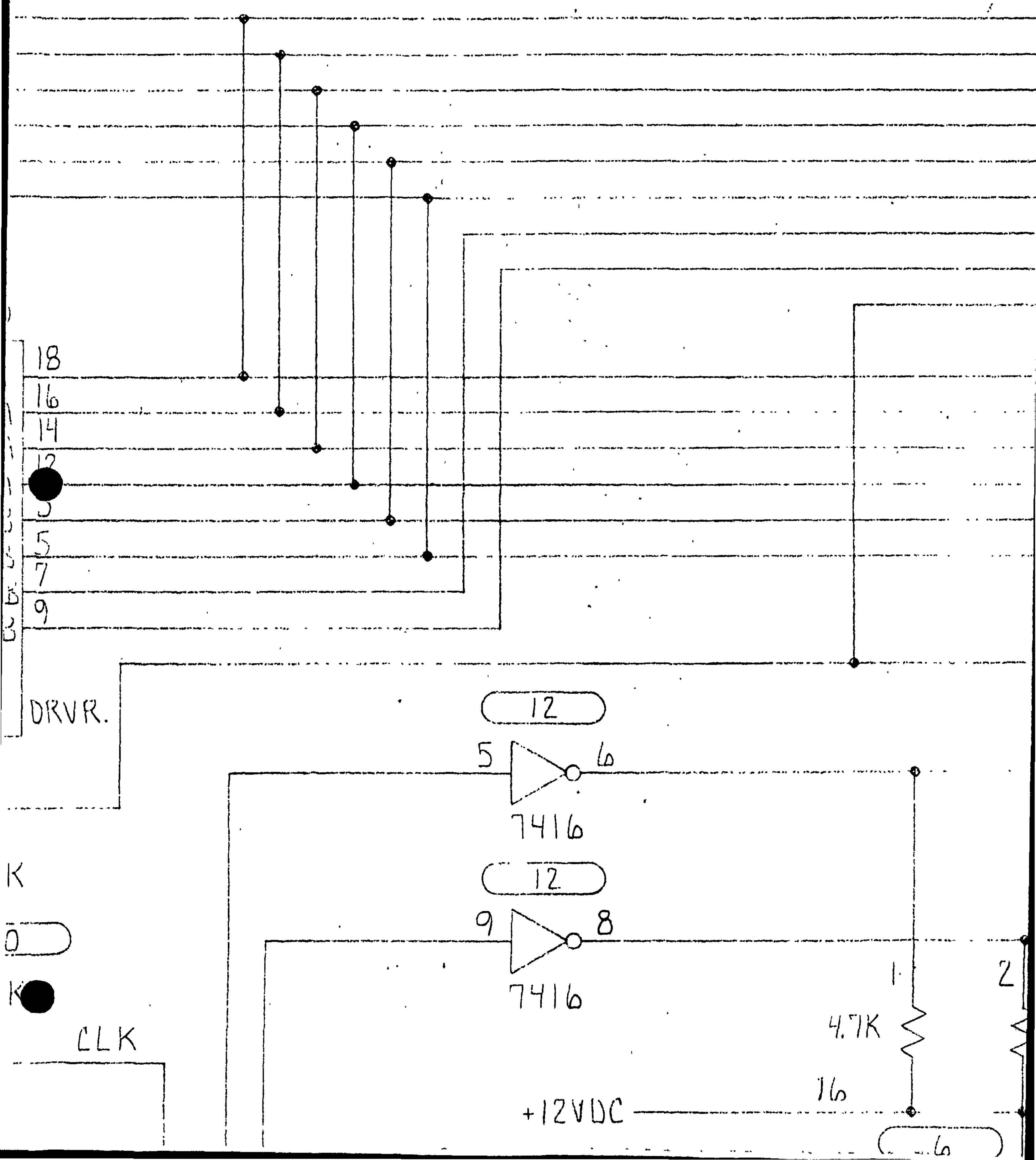
* *Operation above these limits could damage the device.*

NORMAL OPERATING CONDITIONS: $7v \leq V_P \leq 14v$, $0°C \leq T_o \leq 70°C$

B-40


Key: 1 2 3 4 5 6 7 8
D
C
10
+5VDC
16
4.7K
J2
4
P1PA0 39
P1PA1 37
P1PA2 35
P1PA3 33
P1PA4 31
P1PA5 29
P1PA6 27
P1PA7 25
2
4
6
8
17
15
13
11
1
19
1A
2A
3A
4A
1B
2B
3B
4B
ENA
ENB
12
P1CA2 43
1
2
7416
7
+5VDC
16
12
CLOCK3 *
3
4
8
7416

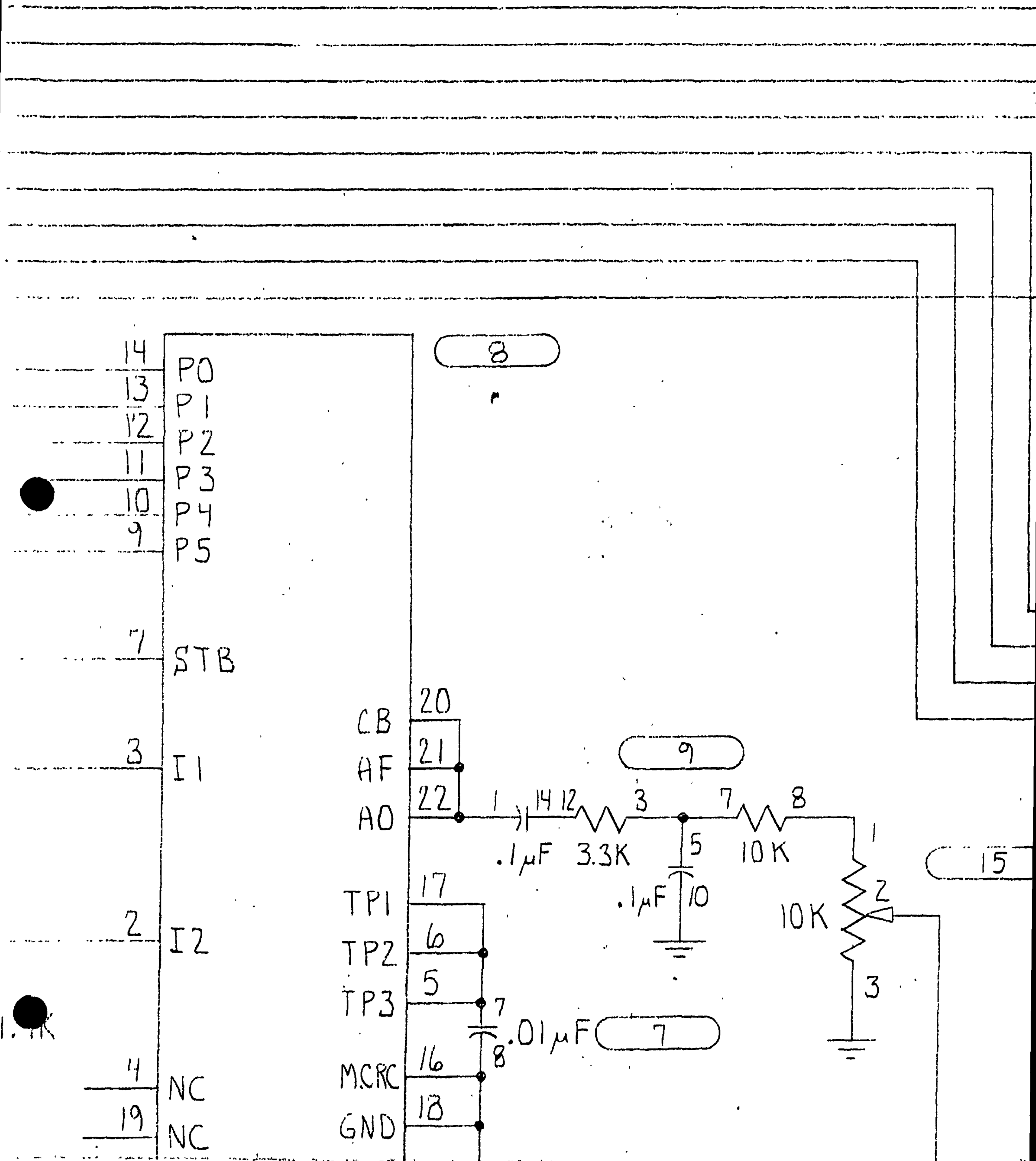
ORIGINAL PAGE IS OF POOR QUALITY



REV | SH | DWG NO

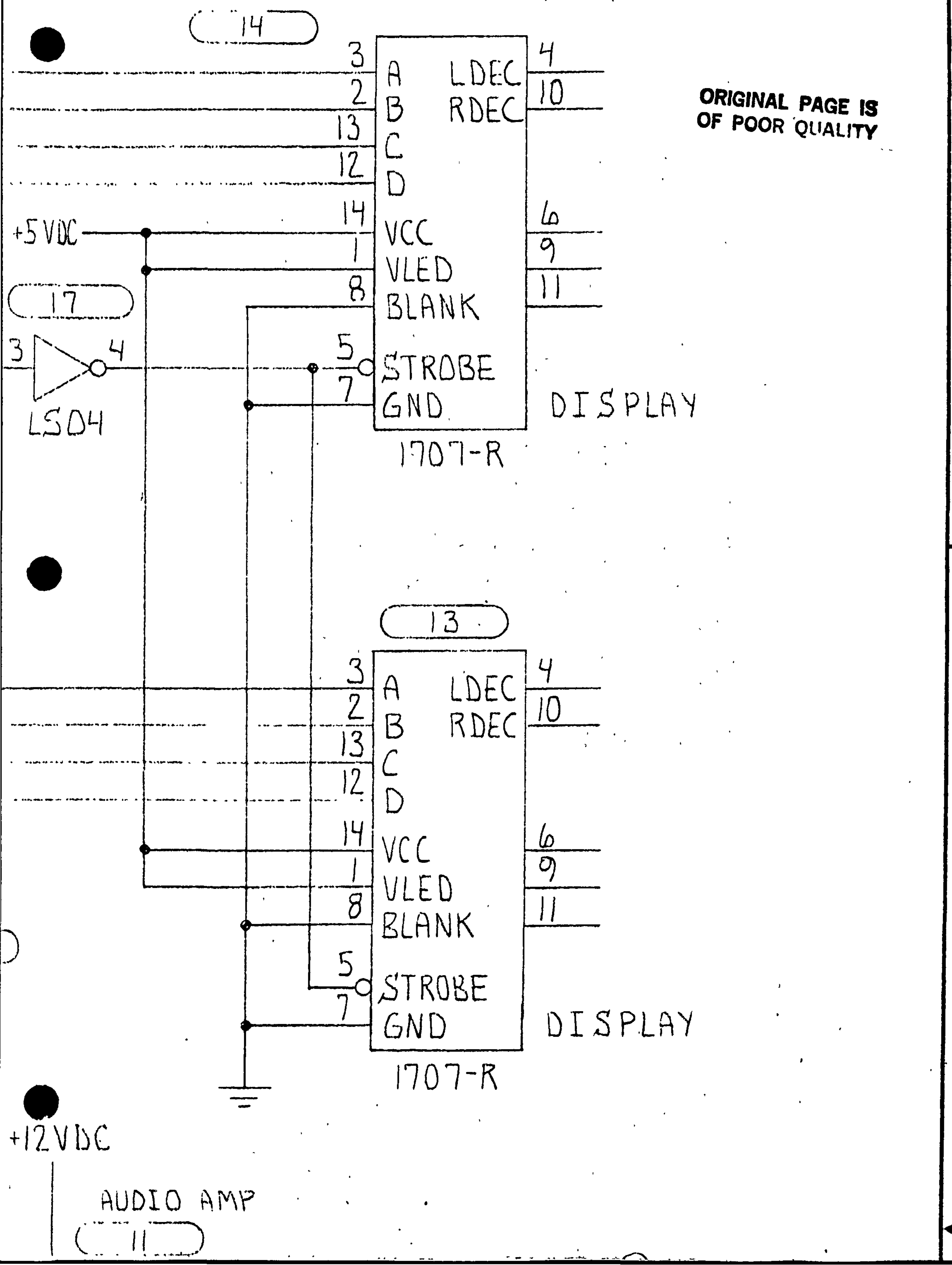
AUTHOR

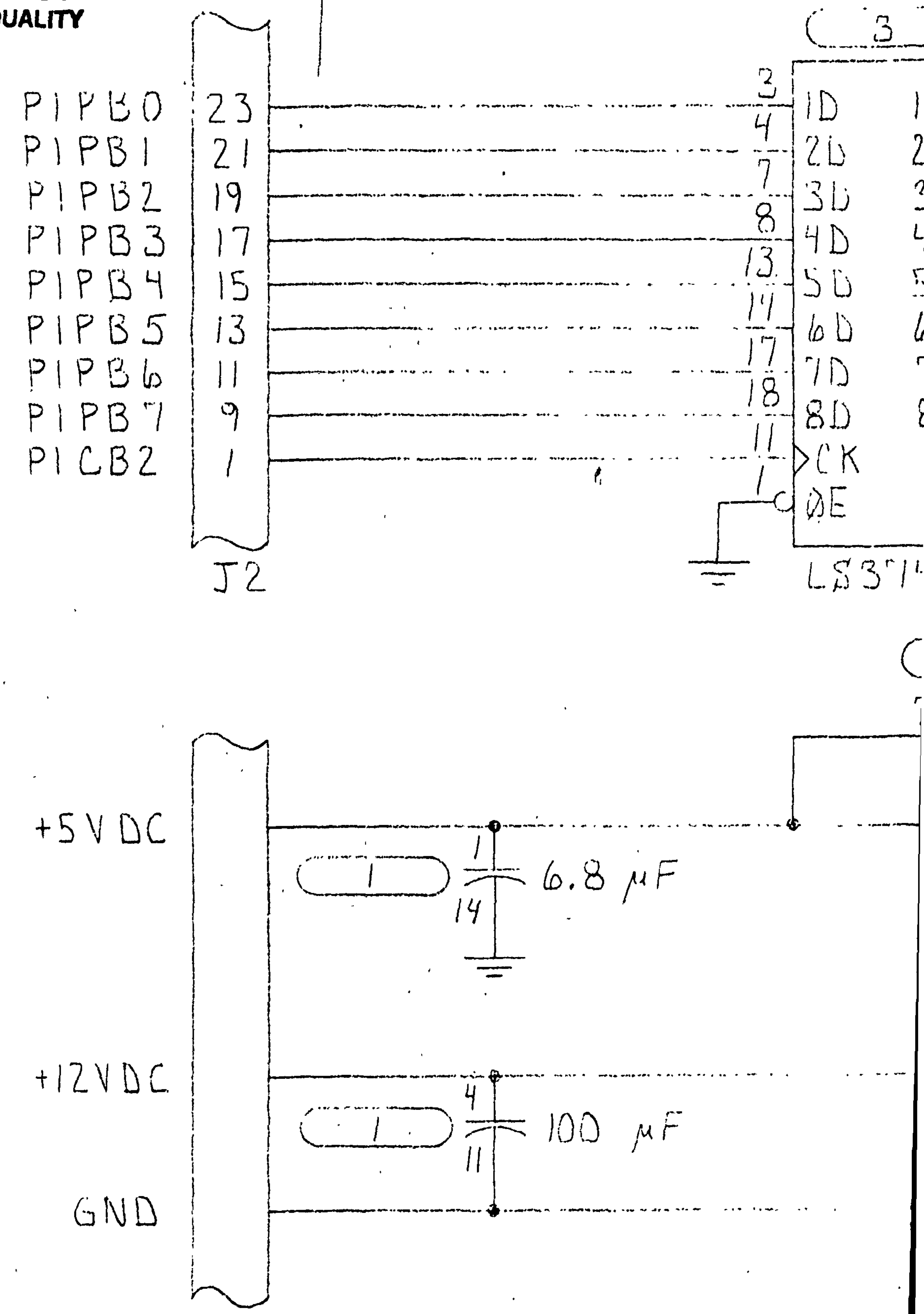
ORIGINAL PAGE IS
OF POOR QUALITY



ITY	ZONE	LTR	REVISIONS DESCRIPTION	DATE	APPROVED

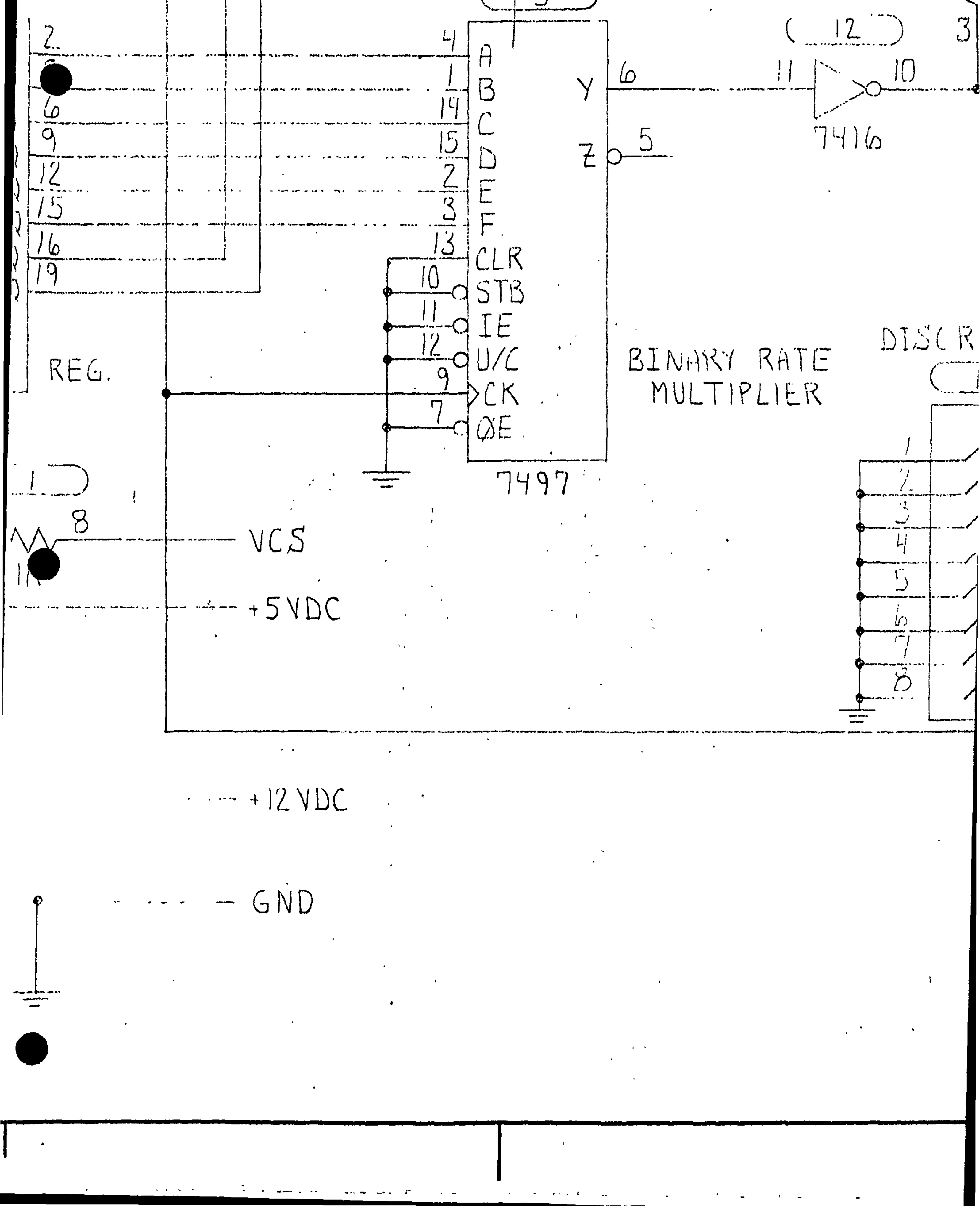
ORIGINAL PAGE IS OF POOR QUALITY

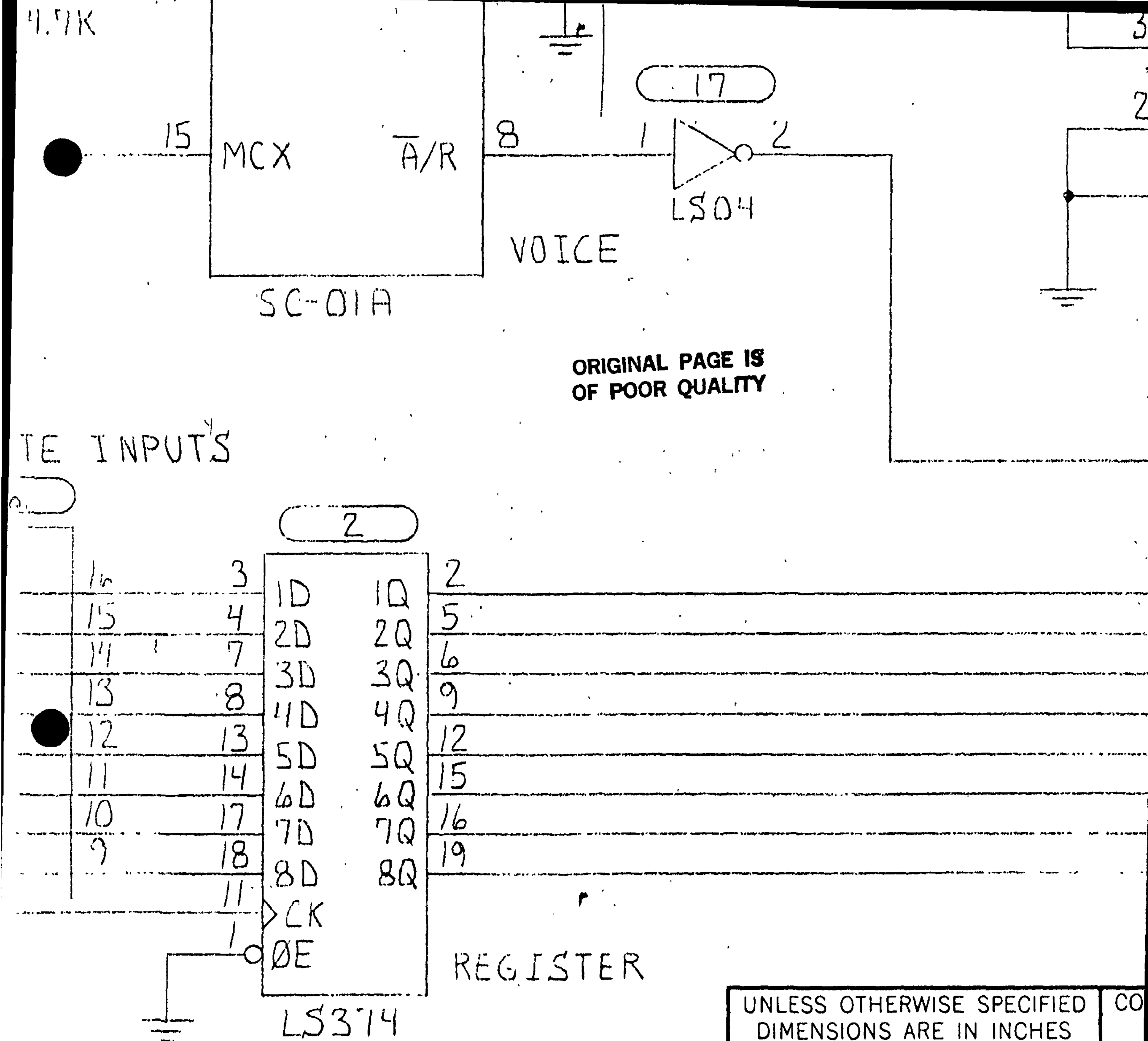



P1PB0
23
P1PB1
21
P1PB2
19
P1PB3
17
P1PB4
15
P1PB5
13
P1PB6
11
P1PB7
9
P1CB2
1
J2
3
4
7
8
13
14
17
18
11
1
1D
2D
3D
4D
5D
6D
7D
8D
CK
ØE
+5VDC
6.8 μF
14
+12VDC
100 μF
11
GND

B

A


5
12
3
REG.
A
B
C
D
E
F
CLR
STB
IE
U/C
CK
ØE
Y
Z
7497
7416
BINARY RATE
MULTIPLIER
DISCR
VCS
+5VDC
+12VDC
GND



UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
AND PER ANSI Y14 5

.XXX	XX	ANGLES
± .010	± 03	±0 5°

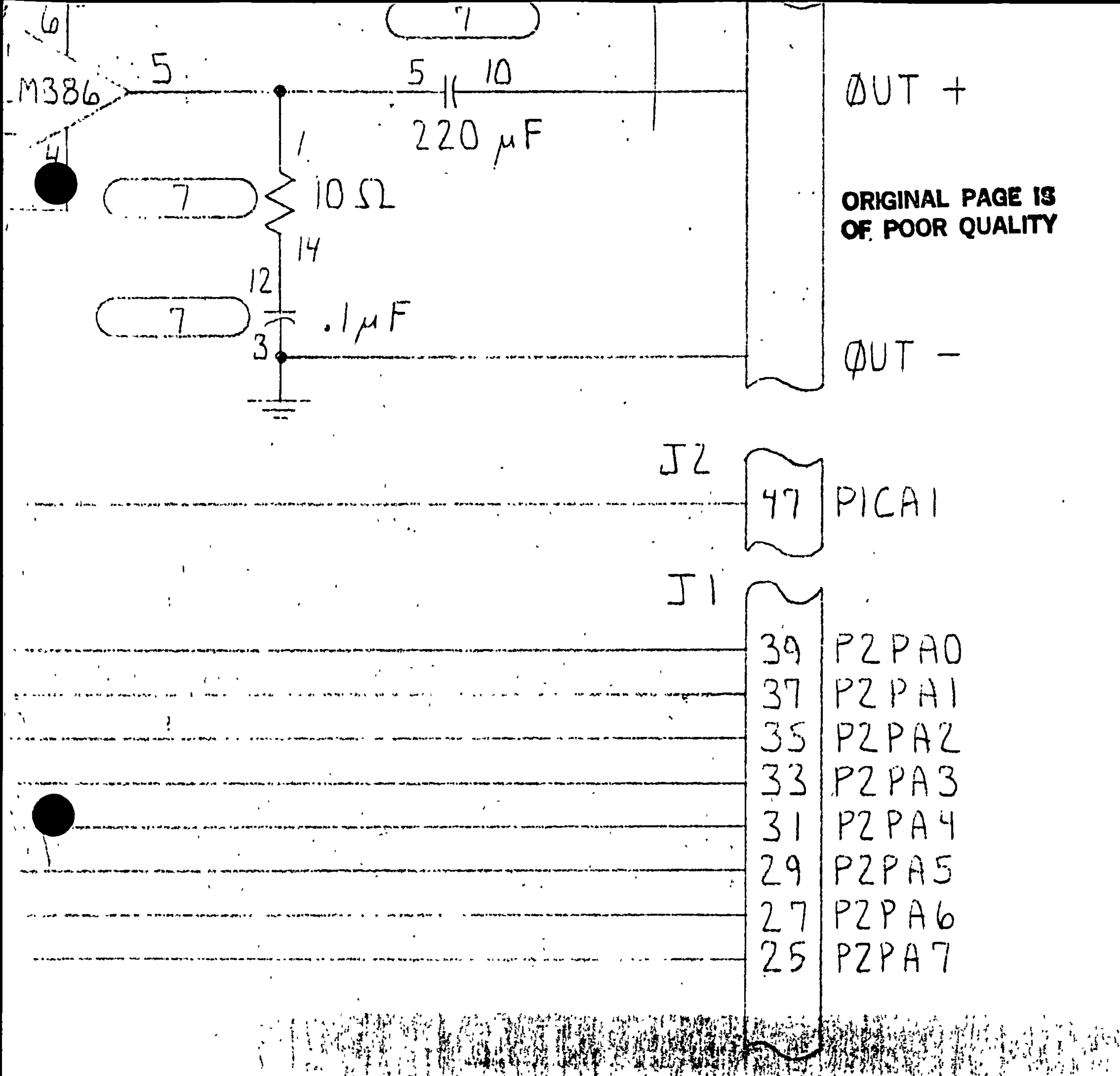
MATERIAL

CO

DR

CH

AP



B

REV

SH

DWG NO

A

RACT
EE 560L

ROUP 2

DATE 1/3/83

SPIN WARNING SYSTEM:
VOICE MODULE

SIZE	FSCM NO	DWG NO	REV
D	05869		

SCALE

SHEET 1 OF 1

```
************************************************************************
*
*
**      PROGRAM TEST1
*
*
************************************************************************
*         THE PROGRAM TEST1 OUTPUT A SENTENCE "UNLOAD" TO SC01
*         WITHOUT USING INTERRUPT AND TIMMER.


          SECTION   9

PHNPOT    EQU       $F70021   PORT #1 LOWER BYTE
PITPOT    EQU       $F70020   PORT #1 UPPER BYTE
PHNCTR    EQU       $F70025   PORT #1 LOWER BYTE CONTROL REGISTER
PITCTR    EQU       $F70024   PORT #1 UPPER BYTE CONTROL REGISTER
LATCH     EQU       $FF

START     MOVE.B    TABLE2,PITPOT            SEND OUT PITCH CODE
          MOVE.B    #0,PITCTR
          MOVE.B    #LATCH,PITCTR            SEND OUT LATCH SIGNAL
          MOVE.B    #0,PITCTR

          CLR.L     D0
          CLR.L     D1
          CLR.L     D2
          MOVEA.L   #TABLE1,A2               SET BASE ADDRESS OF TABLE1
          MOVEA.L   #TABLE3,A4               SET BASE ADDRESS OF TABLE2
          BRA.S     SPEAK
LOOP      MOVE      #570,D2                  SET INSIDE LOOP COUNT
COUNT     SUBQ      #1,D2                    INERT LOOP BODY
          BNE.S     COUNT                    KEEP LOOPING
          SUBQ      #1,D1
          BNE.S     LOOP
SPEAK     MOVE.B    0(A2,D0),PHNPOT
          MOVE.B    #0,PHNCTR
          MOVE.B    #LATCH,PHNCTR
          MOVE.B    #0,PHNCTR
          MOVE.B    0(A4,D0),D1
          CMPI.B    #'?',0(A2,D0)
          BEQ.S     EXIT
          ADDQ      #1,D0
          BRA       LOOP
EXIT      BRA       START
TABLE1    DC        '2MX57'
          DC        $1E
          DC        '?'
TABLE2    DC        $5B
TABLE3    DC        103
          DC.L      71
```

ORIGINAL PAGE IS
OF POOR QUALITY


```
          DC.L      103
          DC        121
          DC.B      90
          DC        55
          DC.B      127
          END
```

APPENDIX C

AIR DATA SYSTEM

SOFTWARE SOURCE CODE

FILE: EQU2   PASCAL   A1   VM/SP CONVERSATIONAL MONITOR SYSTEM

```
PROGRAM MCH ;
FUNCTION MI (PTI,PSI : REAL):REAL;
CONST A,B,C,D,E,F,G;
VAR X: REAL ;
BEGIN
  X:= PTI/PSI ;
  IF X <= 1.893 THEN
      MI := SQRT(5.0)*SQRT(((PTI/PSI)**(2/7)) -1)
   ELSE
 MI :=((A*X-B*X-C*X*X-D*(X**3)-E*(X**4)-F*(X**5)-G*(X**6)-G*(X**9))/X);
END;
BEGIN
END.
```

FILE: EQU3   PASCAL   A1   VM/SP CONVERSATIONAL MONITOR SYSTEM

```
subPROGRAM STATPR ;
  FUNCTION FPSINF (PTI,MI:rEAL):REAL;
   VAR A:REAL;
begin
   A :=1.0/(7.0*MI*MI);
   IF MI <= 1.0 THEN
      FPSINF := PTI/exp(3.5*ln(1+0.2*MI*MI))
   ELSE
    FPSINF := PTI*A*exp(2.5*ln(1-A))/0.1839371 ;
  END.
```

FILE: EQU5   PASCAL   A1   VM/SP CONVERSATIONAL MONITOR SYSTEM

```
subPROGRAM QUCC ;
FUNCTION FQCC (PTI,PSINF:REAL):REAL;
BEGIN
FQCC:= PTI-PSINF;
END.
```

```
subPROGRAM KNOTS;
FUNCTION FKEAS (MINF,PSINF:REAL):REAL;
BEGIN
FKEAS := MINF*661.48*SQRT(PSINF/2116.22);
END.
```



```
subPROGRAM CALSPEED;
FUNCTION FKCAS (QCC:REAL):REAL;
BEGIN
FKCAS := 1479.1*SQRT(exp(0.28571*ln(1+QCC/2116.22))-1);
END.
```



```
subPROGRAM GEO;
FUNCTION FHP(PSINF,PSI:REAL):REAL;
VAR A,R:REAL;
BEGIN
  R:=PSINF/2116.22;
  IF R > 0.223361 THEN FHP:=145442*(exp(0.1902632365*ln(r))-1)
  ELSE IF R > 0.0540328 THEN FHP:=164219-20805*LN(PSI)
       ELSE BEGIN
       A:=exp(0.01463563358*ln(0.0540328/R));
       IF R > 0.00856663 THEN FHP:=710794*A*A-645177
         ELSE BEGIN
         A:=exp(0.04097977*ln(0.00856663/R));
         FHP:=81660.7*A*A-162928.8;
       END
     END;
END.
```

```
*****************************************************************************
*   ADSTIM ASM                                                              *
*   WRITTEN BY : FADI KURDAHI AND CARPO SOSA                                *
*   PURPOSE    : THESE ROUITNES ARE USED TO MEASURE THE EXECUTION TIME*
*                OF EQUATIONS IMPLEMENTED IN ADS.  THE TIMMING IS DONE*
*                IN THE FOLLOWING MANNER:                                   *
*                  SETIME ; (* INITIALIZE, LOAD AND START TIMER *)          *
*                  EQU#; (* INVOKE OPERATION TO BE TIMED *)                 *
*                  COUNT := READTIM ; (* STOP TIMER, READ COUNTER *)        *
*****************************************************************************
*****************************************************************************
*   ROUTINE SETIME                                                          *
*   PURPOSE    : INITIALIZE AND START TIMER #3 OF THE MONOBOARD'S PTM *
*   FORMAT     : USED AS A PASCAL PROCEDURE DECALRED AS:                    *
*                PROCEDURE SETIME ;                                         *
*   METHOD     : TIMER MODE IS CONTINOUS WITH AN INPUT OF 2 MHZ.            *
*                TIMER COUNTER IS LOADED WITH ITS MAXIMUM VALUE (FFFF)*
*****************************************************************************
MSBBUF    EQU       $F70009    ; MSB BUFFER REGISTER (WRITE ADDRESS)
LSBBUF    EQU       $F7000F    ; LSB BUFFER REGISTER (READ ADDRESS)
T3LTCH    EQU       $F7000F    ; WRITE TIMER # 3 LATCHES
T2LTCH    EQU       $F7000B    ; WRITE TIMER # 2 LATCHES
T3CONT    EQU       $F7000D    ; READ TIMER # 3 COUNTER
CNTRL3    EQU       $F70001    ; WRITE CONTROL REGISTER # 3(CR20=0)
CNTRL2    EQU       $F70003    ; WRITE CONTROL REGISTER # 2
CNTRL1    EQU       $F70001    ; WRITE CONTROL REGISTER # 1(CR20=1)
T3MODE    EQU       $80        ; CONTINUOUS MODE CODE
T2MODE    EQU       $A0        ; SINGLE SHOT MODE CODE

          XDEF      SETIME
          SECTION   9
SETIME    EQU       *
          MOVE.L    (A7),A4
          MOVE.B    #T3MODE,CNTRL3
          MOVE.B    #1,CNTRL2
          MOVE.B    #0,CNTRL1
          MOVE.B    #$FF,MSBBUF
          MOVE.B    #$FF,T3LTCH
          JMP       (A4)

*****************************************************************************
*   ROUTINE READTIM                                                         *
*   PURPOSE    : READ THE CONTENTS OF THE COUNTER OF TIMER #3               *
*   FORMAT     : USED AS A PASCAL FUNCTION DECLARED AS:                     *
*                FUNCTION READTIME : INTEGER ;                              *
*   OUTPUT     : READTIM : CONTENTS OF THE COUNTER.                         *
*****************************************************************************
L1        EQU       0

          XDEF      READTIM
          SECTION   9
READTIM   EQU       *
          MOVE.L    12(A5),-(A7)
          LINK      A6,#-L1
          MOVE.L    A6,12(A5)
```

ORIGINAL PAGE IS OF POOR QUALITY


```
          CLR.L     D1
          MOVE.B    T3CONT,D1
          LSL       #8,D1
          MOVE.B    LSBBUF,D1
          MOVE.L    D1,12(A6)
          UNLK      A6
          MOVE.L    (A7)+,12(A5)
          RTS
          END
```

APPENDIX D

USER'S DOCUMENTATION

FOR SYSTEM SUPPORT SOFTWARE TOOLS

Communication between COMPUTERM (EE-560L) and USC-ECL (Engr. Computer Lab)

I.- Network Description:

The COMPUTERM system can be connected to three different systems of the USC-ECL network. This connection allow a COMPUTERM user to work interactively with ECL and transfer files to COMPUTERM.

The COMPUTERM has one of its UART's connected to a 300 bps modem that links this micro computer to the USC-ECL Micom port selector. This selector routes connection to one of the nine systems available at this network. Currently, the lab has access to the following systems:

* ECLA : PDP/10 computer running under TOPS/20 operating system. It has two accounts available for lab use:
  account: sosa password: ask instructor
  account: kurdahi password: ask instructor
* GUMBY : PDP/10 computer running under TOPS/20 operating system.
  account: smyth password: EE560L
* VIRGIL : IBM 370 series 4341 running under CMS operating system.
  account: smyth password: EE560

The following diagram describes the interconnection between the lab and USC-ECL:

```
                                                                       **********
                                                                 +-----*  ECLA  *
                                                                 |     * PDP/10 *
                                                                 |     **********
*************     *********                      ************    |
*           *     *       *        telephone     *  Micom   *-+        ************
* COMPUTERM * <=> * modem * <=//     line   //=> *   Port   *--------*  VIRGIL  *
*           *     *       *                      * selector *-+      * IBM/4341 *
*************     *********                      ************ |      ************
                                                              |
                                                              |      **********
                                                              +-----* GUMBY  *
                                                                    * PDP/10 *
                                                                    **********
```

II.- <u>CPM communication utilities:</u>

* TALK : Program talk.com. Allows interactive communication with any of the ECL-systems mentioned above.
* TOPCPM : Program topcpm.com. Allows text file transfer from ECLA or GUMBY (TOPS/20) to COMPUTERM.
* CMSCPM : Program cmscpm.com. Allows file transfer from VIRGIL (CMS) to COMPUTERM. File transfer is limited to S-record files only. S-record is a file format created by Motorola and used to allow easy inter-computer transportation of object code for their microprocessors.

III.- <u>How to establish connection:</u>

1.- Boot COMPUTERM system and turn on modem on auto-answer mode.

2.- Dial to ECL: 743-5030 or 743-7646. If you are on campus omit dialing the prefix 743.

3.- Invoke the program TALK.COM, as soon as the carrier is detected by the modem:

A>TALK

4.- Wait for five seconds and then press hit <CR>. The following message will be prompted:

USC-ECL Micom port selector.
Which system?

5.- At this point, you should select your system by typing IBM, GUMBY or ECLA.

6.- System identification will be prompted indicating that you can now logon to the account. The ECL-TOPS/20 and ECL-IBM/CMS manuals provide a description of the main commands available in these operating systems.

7.- At any point you can go back to CPM by typing control-tilde (ctrl-~). This will not disconnect you from ECL. Communication can be reestablished by invoking TALK.COM.

5.- When you are done, logoff, return to CPM (ctrl-") and turn modem off.

IV.- File transfer to COMPUTERM:

Refer to the documents:

* Communication between COMPUTERM and IBM 370/4341 (ECL-VIRGIL).
* Communication between COMPUTERM and PDP/10 (ECLA or GUMBY).

ORIGINAL PAGE IS OF POOR QUALITY

Communication between COMPUTERM and IBM 370/4341 (ECL-VIRGIL):

* This process requires two programs: TALK.COM and CMSCPM.COM
* To set the connection between the two systems follow the following steps:

  1.- Plug and turn modem on and dial 5030 or 7646.

  2.- On CPM: execute TALK.COM

  A>TALK

  this will cause a virtual attachment of your terminal to the MICOM port selector. When you select IBM and you are asked about your terminal type, you should answer DM1520. If you want to return to CPM, type ctrl-". This will not log you off nor detach you from the system. Communication can be set again by repeating step 2.

  3.- When you are done, log off from VIRGIL and return to CPM using ctrl-". Unplug the modem.

* File transfer is limited only to one direction, VIRGIL -> COMPUTERM and can only be used with S-format record files with less than 48K bytes. The steps that must be followed are:

  1.- Set communication between the two system using the procedure described above.

  2.- When you are ready to transfer the file, return to CPM and execute TOPCPM.COM:

  A>TOPCPM XX destination_file_in_CPM

  A>TOPCPM XX B:TEST.OBJ

  This will reconnect the two systems in the same way as TALK.COM does. If by some reason you have to return to CPM before step 3, do it in the usual manner (ctrl-").

  3.- On CMS: type the CLEAR key and execute the EXEC file CMSCPM EXEC:

ORIGINAL PAGE IS OF POOR QUALITY

CMSCPM source_filename_in_CMS

CMSCPM TEST

the file extension OBJCT is assumed, ie, TEST stands for TEST OBJCT. The file will be typed to the screen as it gets copied. The process is slow because every time that the screen is filled, VIRGIL waits 40 seconds before continuing the transfer. At the end a message will be displayed and control will be passed to CPM. If you notice that the program shows abnormal behavior or you want to abort it, you should press the RESET button in the COMPUTERM box. Then reboot CPM and use TALK.COM to retry or logoff.

4.- Use TALK.COM to continue or to logoff from VIRGIL.

5.- If your CMS file exceeds 48K bytes. You should split your file in a number of files with a permissible size for transfer. To reconstruct the file on CPM use the PIP.COM utility:

```
A>PIP Big_File=Small_File_1,...,SmallFile_n
A>PIP A:TEST.OBJ=B:TEST1.OBJ,TEST2.OBJ,A:TEST3.OBJ
```

ORIGINAL PAGE IS
OF POOR QUALITY

APPENDIX E

LISTING OF THE OBS SOURCE PROGRAM

```
PROGRAM SIM3DOF(INPUT,OUTPUT);


CONST

    CONST1 =1.0;
    CONST2 = 13.75 ;
    GRAV = 10.0;
    VELOCITY = 200.0;
    TVELOCITY = 200.0 ;
    CONST4 = 1.0;
    CONST5 = 2.0;
    TAU1 = 0.40;
    TAU2 = 0.25;
    TAU3 = 0.50;
    TFINAL = 20.0;
    PHILIM = 10.0;
    N = 5;
    DRAD = 57.3 ;
    GUIDGN = 1.0 ;
    CAPLAMBDA = 4.0 ;

VAR

    RANGE,ETA,A,RELRD,LAMBDA,GAMMADC,GAMMAH,GAMMAT : REAL;
    XT , YT , XH , YH , XDT , YDT , XDH , YDH , DELTAXT , DELTAYT : REAL ;
    DELTAXH , DELTAYH , NEWSIGMA , OLDSIGMA  : REAL ;
    SIGMAD : REAL ;

    (* VARIABLE NAMES STARTING WITH T REFER TO THE TARGET *)
    TEPS , TLEADX , TLEADXD , TDELT , TDELTD , TPHI , TPHID : REAL ;
    TPSI , TPSID , TPSIDD ,TPSIC : REAL ;
    (* VARIABLE NAMES RELATIVE TO THE HOST PLANE *)
    EPS,LEADX,LEADXD,DELT,DELTD,PHI,PHID,PSI,PSID,PSIDD,T: REAL;
    D : ARRAY [1..10] OF REAL;
    X : ARRAY [1..10,1..2] OF REAL;
    DT : ARRAY [1..10] OF REAL ;
    TX : ARRAY [1..10,1..2] OF REAL ;
    PRINTI : INTEGER;
    K,J,K : INTEGER;
    DELTAT : REAL;
    PSIC   : REAL ;
    PSIT   : REAL ;


BEGIN (* MAIN PROGRAM *)

    T := 0.0;
    LEADXD := 0.0;
    TLEADXD := 0.0 ;
    DELTD := 0.0;
    TDELTD := 0.0 ;
    PHID := 0.0;
    TPHID := 0.0 ;
    PSIDD := 0.0;
    TPSIDD := 0.0 ;
    PSID  :=0.0;
    TPSID := 0.0 ;
    PRINTI := 0;
    LEADX := 0.0;
    TLEADX := 0.0 ;
    DELT  := 0.0;
```

```
TDELT := 0.0 ;
PHI := 0.0;
TPHI := 0.0 ;
DELTAT := 0.05;
PSIT := 270.0/CRAD;
PSI := 90.0/CRAD;
SIGMAD := 0.0 ; (* LINE OF SIGHT *)
XT := 2000.0;
YT := 5000.0;
NEWSIGMA :=ARCTAN(YT/XT)-PSI;
GAMMAH := 90.0/CRAD;
GAMMAT := 270.0/CRAD;
WRITELN ('      ON BOARD  AIR TO AIR INTERCEPTION SIMULATOR' ) ;
WRITELN ;
XH := 0.0 ;
YH := 0.0 ; (* INITIAL POSITION OF THE HOST IS 0,0  *)

(* COMPUTATION OF THE INITIAL LINE OF SIGHT *)

  WRITELN ('  INITIAL LINE OF SIGHT  =========>     ',NEWSIGMA) ;

(* _____________SIMULATION AND GUIDANCE LOOP _____________________*)

WHILE (T <= TFINAL) DO
BEGIN

   (* INITIALIZATION WITHIN THE LOOP *)

        XOT := XT ;
        YOT := YT ;
        XOH := XH ;
        YOH := YH ;
        OLDSIGMA := NEWSIGMA ;


   (* GUIDANCE LAW CALCULATION *)

   RANGE  := SQRT ( SQR(XOT-XOH)+SQR(YOT-YOH) ) ;
  ETA := GAMMAH- NEWSIGMA;
   A := GAMMAT - NEWSIGMA;
   RELPD := (1.0 - ((TVELOCITY*COS(A))/(VELOCITY*COS(ETA))));
   LAMBDA := CAPLAMBDA * RELPD ;
   GAMMADC := LAMBDA * SIGMAD ;
   PSIC := GAMMADC * GUIDGN ;

   (* SET THE DERIVATIVE EQUATIONS *)

   EPS := PSIC-CONST4*PSI-CONST5*PSID;
   TEPS := TPSIC - CONST4*TPSI - CONST5*TPSID ; (* TARGET *)
   TLEADXD := (TEPS*CONST1 - TLEADX)/TAU2 ;     (* TARGET *)
   LEADXD := (EPS*CONST1 - LEADX)/TAU2;
   TDELT := TLEADX + TLEADXD * TAU1 ;
   DELT := LEADX + LEADXD*TAU1;
   PHID := DELT*CONST2;
   TPHID := TDELT * CONST2 ;                    (* TARGET *)
   (* SHUNT LIMIT THE AIRCRAFT ROLL ANGLE *)
   IF (PHI*PHID > 0.0) AND (ABS(PHI) > PHILIM) THEN PHID := 0.0;

  (* SHUNT LIMIT THE TARGET AIRCRAFT *)
   IF(TPHID*TPHI > 0.0) AND ( ABS(TP   > PHILIM) THEN TPHID:=0.0;
   PSIDD := ((GRAV/VELOCITY)*PHI-PSI   TAU3;
   TPSIDD := ((GRAV/TVELOCITY)*TPHI-TPSID)/TAU3 ;
```

E-2

ORIGINAL PAGE IS
OF POOR QUALITY

```
(*            AIRCRAFTS      GRID GEOMETRY  *)

DELTAXT := TVELOCITY * DELTAT * COS(PSIT);
DELTAYT := TVELOCITY * DELTAT * SIN(PSIT);
XT      := X0T + DELTAXT ;
YT      := Y0T + DELTAYT ;
DELTAXH := VELOCITY * DELTAT * COS(PSI);
DELTAYH := VELOCITY * DELTAT * SIN(PSI);
XH      := X0H + DELTAXH ;
YH      := Y0H + DELTAYH ;
GAMMAH  := ARCTAN((YH-Y0H)/(XH-X0H)) ;
GAMMAT  := ARCTAN((YT-Y0T)/(XT-X0T)) ;
NEWSIGMA := ARCTAN((YT-YH)/(XT-XH))  ;
SIGMAD   := (NEWSIGMA-OLDSIGMA)/DELTAT ;


(* ASSIGN *)
I[1,1] := LEADK ;
TI[1,1] := TLEADK  ;
I[1,2] := LEADKD ;
TI[1,2] := TLEADKD ;
I[2,1] := DELT ;
TI[2,1] := TDELT ;
I[2,2] := DELTD ;
TI[2,2] := TDELTD  ;
I[3,1] := PHI ;
TI[3,1] := TPHI ;
I[3,2] := PHID ;
TI[3,2] := TPHID ;
I[4,1] := PSID ;
TI[4,1] := TPSID ;
I[4,2] := PSIDD ;
TI[4,2] := TPSIDD  ;
I[5,1] := PSI ;
TI[5,1] := TPSI          ;
I[5,2] := PSID ;
TI[5,2] := TPSID ;

(* INTEGRATION *)
FOR J := 1 TO N DO
   O[J]:= I[J,1] + I[J,2]*DELTAT;
   OT[J] := TI[J,1] + TI[J,2]*DELTAT ;

(* REASSIGNMENT *)
LEADK := O[1];
TLEADK := OT[1] ;
PHI := O[3];
TPHI := OT[3] ;
PSID := O[4];
TPSID := OT[4] ;
PSI := O[5];
TPSI := OT[5] ;


(* PRINT OUT THE PARAMETERS EVERY (DELTAT*25) INTEGRATIONS *)
PRINTI := PRINTI + 1;
IF PRINTI = 1 THEN
BEGIN
   WRITELN('TARGET X,Y,PSI',XT,YT,PSIT*CRAD);
   WRITELN('HOST   X,Y,PSI',XH,YH,PSI*CRAD);
   WRITELN('HOST SIGMA,GAMMA,PSICOMMAND',NEWSIGMA*CRAD,GAMMAH*CRAD,PSIC*CRAD);
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
            WRITELN('TRACKING INFO RANGE,A,ETA,RELR,GAMDC',RANGE,A,ETA*CRAD,RELRD,GAMMADC*CRAD);
            WRITELN;
            FOR X := 1 TO N DO
                 BEGIN
                  WRITELN(T,I[X,1],I[X,2],D[X]);
                  WRITELN('TARGET PARAMETERS' ,TI[X,1],TI[X,2],DT[X]) ;
                 END;
         WRITELN;
            PRINTI:= PRINTI - 25;
         END;
         T := T + DELTAT;
      END;
      WRITELN('SUCCESS');
   END.
```

E-4