NASA GRANT NGL-44-005-084

AEROSPACE-RELATED HYBRID COMPUTING FACILITY

Interim Report

Get DRA

N71 70630

(ACCESSION NUMBER) (THRU)

444 (PAGES) (CODE)

CR-116203 (NASA CR OR TMX OR AD NUMBER) (CATEGORY)

FACILITY FORM 602

University of Houston

Cullen College of Engineering

Houston, Texas 77004

INTERIM REPORT

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GRANT N44-005-084

"AEROSPACE-RELATED HYBRID COMPUTING FACILITY"

January, 1971

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GRANT N44-005-084

AEROSPACE-RELATED HYBRID COMPUTING FACILITY

Interim Report

1. INTRODUCTION

The Cullen College of Engineering, University of Houston with the purchase of an analog computer supported by NSF Grant GK-2436 "Hybrid Computing Facility" completed the installation of a hybrid computing facility in July, 1969. Since its installation some fifty graduate students have had substantial exposure to this research tool in the form of a graduate course or individual tutorial. Three students have received degrees (2 M.S. and 1 Ph.D.) essentially on the basis of analog/hybrid-computer oriented topics with laboratory costs supported by NASA Grant N44-005-084 "Aerospace-Related Hybrid Computing Research." Several other projects are currently active.

All phases of the operation such as software, system performance, maintenance and supervision are on a firm footing. Plans for future extension of capabilities include:

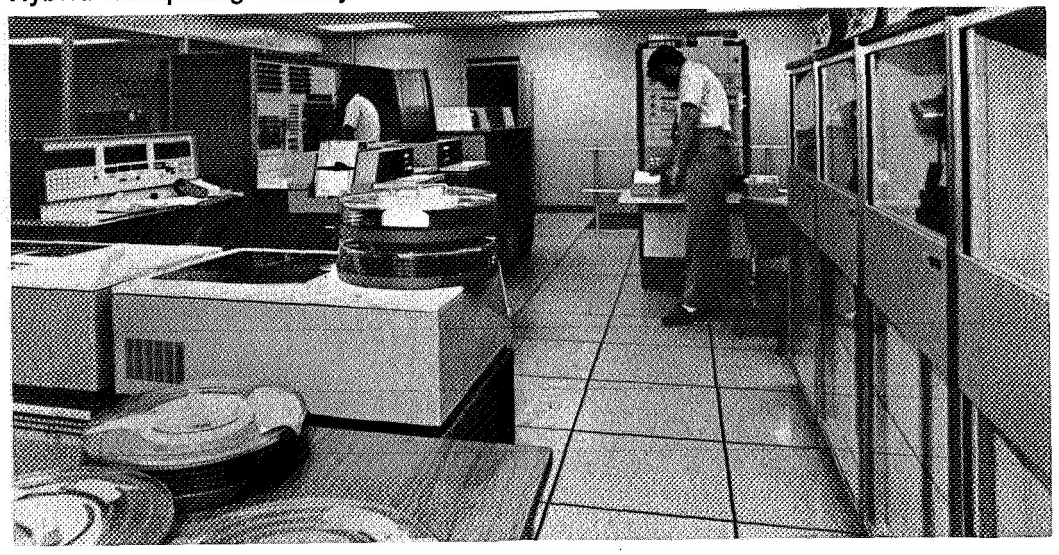
1. Expansion of disc memory from 21.75 million bytes to 116 million bytes
2. Expansion of core memory from 128K bytes to 256K bytes

3. Additional plug in components for the analog computer: multipliers, relays, digital coefficient units
4. A mini-computer as a function generator
5. A graphic console for data display, programming and control
6. Remote display facility for classroom instruction
7. Data communication interface for on-line data acquisition and network operation with UNIVAC 1108 of the University Computing Center

These expansion items are contingent on funding of additional grants or contracts now in the proposal stage.

2. DESCRIPTION OF HARDWARE AND SOFTWARE

Hybrid Computing Facility



The computing equipment in the Engineering Systems Simulation Laboratory of the Cullen College of Engineering, University of Houston, consists of an IBM Model 44 digital computer which has 128K bytes of core memory, two high-speed multiplexer channels, a low-speed multiplexer channel and the 32-level priority interrupt feature. Three 2311 disc drives are switchable between the two high-speed multiplexer channels. The single-disc storage drive in the model 44 processing unit is attached to a subchannel on one high-speed multiplex channel. There are two 9-track tape drives, a card read punch, a line printer, and a console typewriter attached to the multiplexer channel. The digital computer configuration is shown in Figure 1.

Communication between the 360/44 and the analog computer is through a Hybrid Systems Model 1044 Hybrid Linkage unit. Data transfer between the linkage and the digital computer is split between the two high-speed multiplexer channel. Digital data for conversion at the interface is transmitted from the 360 to the linkage over a subchannel of the second high-speed multiplex channel. The "split configuration" of the linkage, insofar as A-to-D and D-to-A activity is concerned is probably unique to this installation. The purpose of such a split is that much higher effective data rates are possible with the input and output activity operating asynchronously over two separate channels.

ENGINEERING SYSTEMS SIMULATION LABORATORY

IBM System 360/44
Digital Computing System

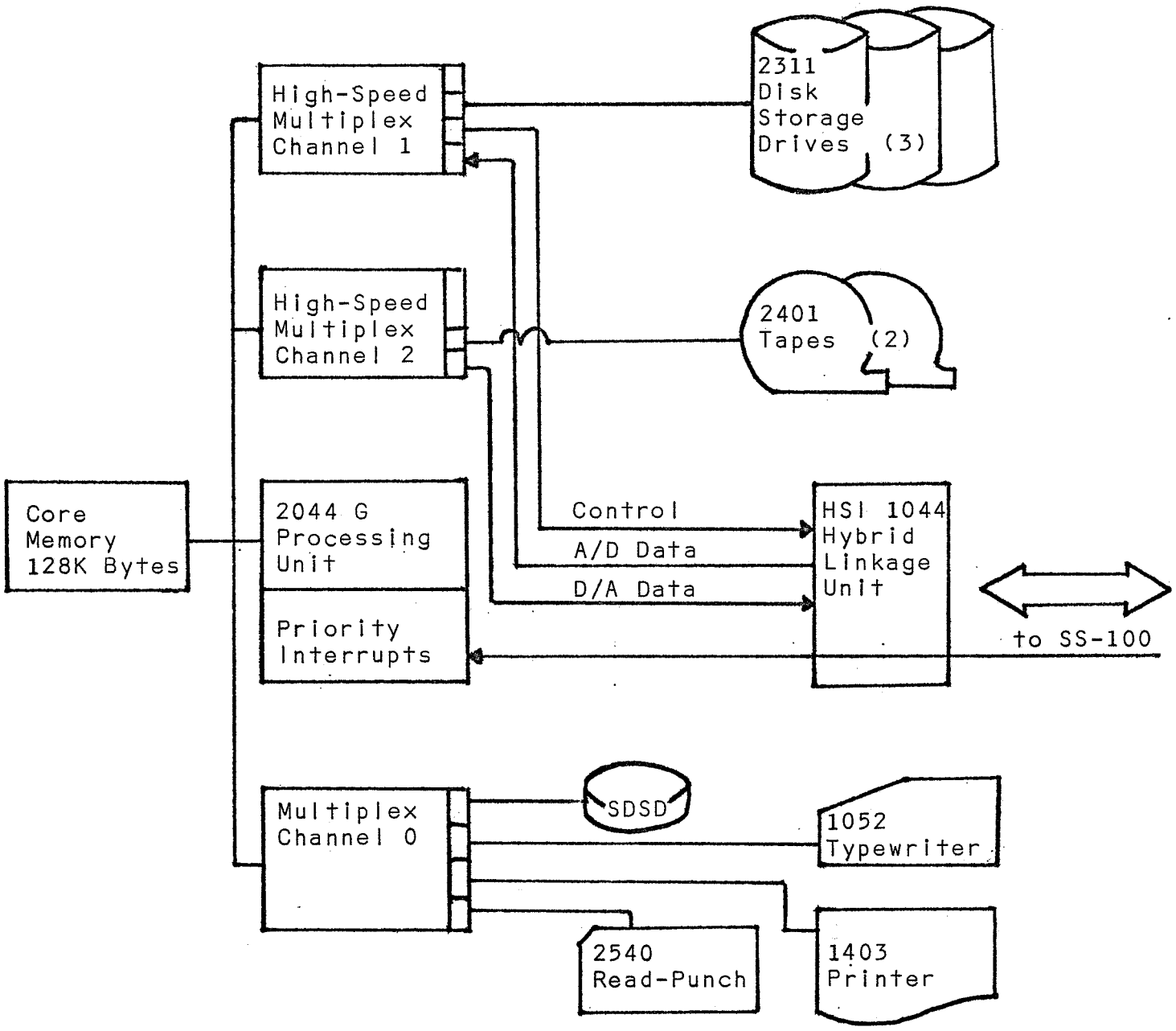


FIGURE 1

The analog computer is a Hybrid Systems Incorporated, Model SS100. Components of the analog computer summarized in Figure 2 include 54 summer/inverter amplifiers, 36 summer/integrator amplifiers, 80 servo-set potentiometers, manual potentiometers, D-to-A switches, sample-and-hold amplifiers, comparators, multipliers, sine-cosine generators and most of the lesser analog and logic components associated with an analog computer of this size and of recent manufacture. Presently Analog output is being handled by photo-recording from an oscilloscope or by digitizing and line printing.

Major software elements for the system are shown in Figure 3. The operating system for the 360/44 used in this hybrid environment is the Data Acquisition Multiprogramming System (DAMPS-II). This system is a rather elaborate extension of the programming system for the 360/44, 44PS and is furnished by IBM for the 360/44 in real-time applications. The functions of DAMPS are supplemented and extended by a system of programs, run under DAMPS, known as the Hybrid Executive. These extensions were developed by Hybrid Systems, and serve to expand the functions of DAMPS in a manner appropriate to hybrid computation.

Two additional software packages are also available. The first of these is called ANASET. Facilities are provided in ANASET for setup, static check, dynamic check, and execution monitoring of programs for the hybrid computer, utilizing a

Hybrid Systems SS-100 Analog Computing System

54	Summer-inverter amplifiers
36	Summer-integrator amplifiers
6	Dual inverter amplifiers
80	Servo-set potentiometers (with slew control)
12	Manual potentiometers
16	Digital-analog switches
16	Sample-and-hold amplifiers
12	Comparators
2	Sine-cosine generators
5	Feedback limiters
8	Quarter-square multipliers
20	Log(x) diode function generators
5	Bridge limiters
12	SPDT function relays
8	Shift registers (32 flip-flops)
50	AND/NAND gates
2	Three-mode timers
2	Single-shot multivibrators
8	Logic pushbuttons
16	Logic-level indicators
16	Analog to digital conversion multiplexer channels (13 bits + sign)
16	Multiplying digital to analog converters
16	Sense lines
32	Control lines
16	Priority interrupts
	Mode and timer control features
	Analog address and readout system
	6 Channel recorder
	x-y Plotter 8 1/2" x 11"
	Variable persistence oscilloscope

FIGURE 2

HYBRID PROGRAMMING SUPPORT SOFTWARE

DAMPS - II

Hybrid Executive

ANASET

AutoTrak

FIGURE 3

specifically restricted FORTRAN-IV statement structure for reference to analog computer components. The second subsidiary software package is called AutoTrak. AutoTrak permits the display, monitoring and modification of variables at execution-time in FORTRAN IV-coded programs at the console typewriter. Reference to these FORTRAN variables may be made either by variable name, absolute address, or by relative address. Combination of ANASET with AutoTrak allows the modification of potentiometer settings, scale factors, and other variables pertinent to the analog computer from the digital computer at the time the ANASET program is being executed.

The memory of the 360/44 under DAMPS is divided into two user partitions, a background partition and a real-time partition. DAMPS functions are summarized in Figure 4. Processing in the background partition is in conventional batch mode. 44PS jobs coded in FORTRAN-IV or assembler language can be executed in the roughly 49,000 byte area provided in the background partition. All the facilities of 44PS including overlay are available to programs being processed in the background under DAMPS. Activity in the real-time partition, the execution of real-time jobs (RTJs), falls essentially in two categories. The processing associated directly with priority interrupts is handled by so-called

DAMPS Functions

(Hybrid Programming Support)

Identify RTJs

Queue FGTs

Dequeue FGTs

Terminate FGTs

Attach RTTs to PILs

Terminate RTTs

Enable PILs

Disable PILs

Terminate RTJs

Real-Time Input-Output

- Set up controls for RTIO
- Perform (overlapped) RTIO
- Check RTIO

FIGURE 4

real-time tasks, (RTTs) one RTT being attached to each priority interrupt level (PIL). RTTs may be coded either in FORTRAN-IV or in assembler language, but in either case are routines which execute very rapidly in response to interrupts. More lengthy computational tasks, such as non-real-time input/output, etc. may be queued by RTTs for execution in the real-time partition upon completion of all pending priority-interrupt processing. These foreground tasks (FGT's) have all facilities of the DAMPS supervisor available. DAMPS-II provides a real-time partition of approximately 37,000 bytes of core storage.

The batch jobs constituting the background job stream are input through the card reader. In this two-partition mode, the loading of RTJs is control-led from the console typewriter, the programs making up the RTJ being loaded from a previously compiled and linkage-edited program library. The initiation of an RTJ consists essentially of entry of appropriate job control statements from the typewriter. In the event that a real-time job requires more storage than that available in the real-time partition, DAMPS-II may be operated in a single-partition mode, providing both the space of the real-time partition and that of the background partition to the real-time job. Control statements for the real-time job executed in this one-partition mode are then entered through the card reader.

DAMPS provides a number of functions to the real-time problem programmer in the form of subroutines which, as are appropriate, may be called from RTTs or from FGTs. Activities falling into this category include the identification of RTJs, the queueing, dequeueing, and termination of FGTs, the enabling and disabling of PILs, the attachment of RTTs to PILs, the termination of RTTs, non-real-time I/O from FGTs, real-time I/O from RTTs, (both overlapped and not overlapped with other processing), and the termination of RTJs. Functions provided in a similar manner by the Hybrid Executive shown in Figure 5 include the dynamic allocation of RTTs to PILs, display of the FGT queues, the construction of so-called uncontrolled RTTs, overlapped non-real-time I/O, manipulation of the interval timer, programmable delay, internal triggering of PILs, a set of pseudo-sense switches, and execution-time accounting. Time and space do not permit a detailed treatment here of features and operation of DAMPS and the Hybrid Executive, nor for that matter, of ANASET and AutoTrak.

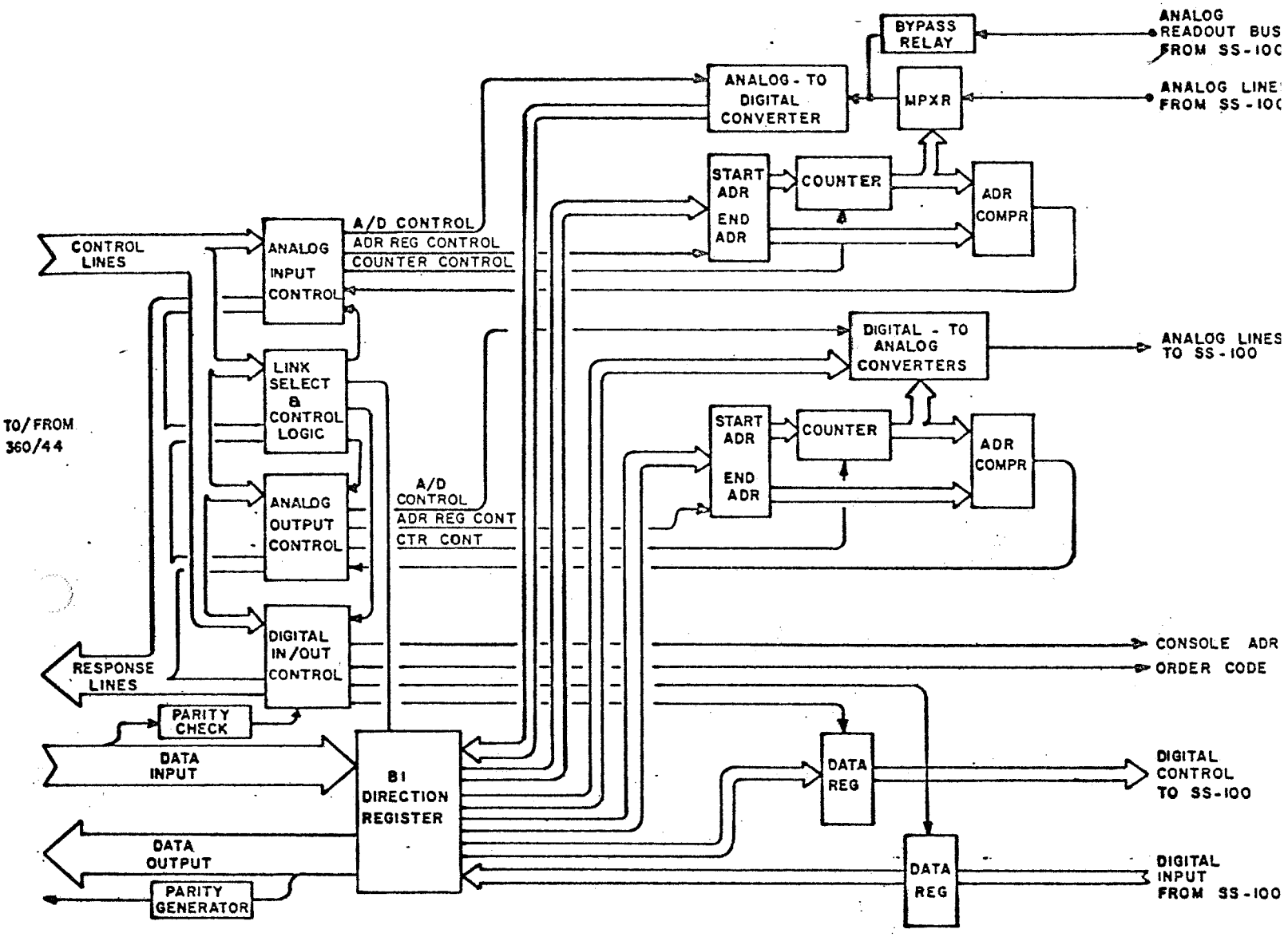
The linkage interface unit, HSI 1044, is described in Figure 6. As mentioned earlier the linkage for this particular system includes dual control logic and dual bi-directional registers to allow independent, asynchronous operation of analog input (A/D) and analog output (D/A). The control functions shown in Figure 6 are actually duplicated in the interface.

HYBRID EXECUTIVE FUNCTIONS

(Supplementing DAMPS)

Dynamic PIL - RTT allocation
Display FGT Queues
Trigger PILs internally
Attach uncontrolled RTTs
Overlapped non-RTIO
Interval timer handling
Pseudo-sense switches
Programmable delay
Execution-time accounting
Data conversion functions and control
Analog mode functions
Single bit communication
Addressing and read out of analog components
Potentiometer setting
Timer Control

FIGURE 5



BLOCK DIAGRAM OF 1044

FIGURE 6

The interface includes the control features for sequential and random addressing of A/D multiplexer channels and the multiplying D/A converters. In addition, these data transmission activities can be operated with the CPU channel in either the burst or multiplex mode. When the channel is in the multiplex mode (essential for random addressing) the transmission of A/D multiplexer or D/A converter addresses is interleaved with the input or output of analog data. The interface operation can also be controlled in either mode or direction by external synchronization where the conversion of each sample is under the control of a logic signal from a clock, gate, comparator or timer at the analog console. In the sequential, internally synchronized mode the data transfer rate approaches 100,000 samples per second concurrently on A/D and D/A channels. This is equivalent to a transfer rate of 400,000 bytes per second.

Single bit data transfers for analog control (control lines) or status sensing at the analog (sense lines) can also be controlled through the interface. All other modes of analog control, addressing and readout are available to the digital program. Thus, all analog computer elements may be addressed and read out over the A/D converter, the various modes such as IC, OP, HOLD, POTSET, RATE TEST, STATIS TEST can be controlled from the digital computer. Four decade timer intervals, clock rate and mode such as 2-mode repetitive

operation (IC, OP, HOLD) can also be initialized from the IBM 360. Analog read-out is done through the A/D converter automatically without patching at the analog console in preference to reading the digital voltmeter which would slow down the process considerably.

The analog elements are designed with a band width of roughly 100 KH for diode function generators (multipliers, log and sine-cosine functions) and 300 to 500 KH for the linear equipment (amplifiers). Switching times are on the order of one half micro-second or less for logic elements, D/A switches and mode control switches. Noise levels, unfiltered, are in the 1-5 multivolt range depending on component class. All switching is electronic with liberal use of FET switches. Two or three-mode repetitive operation at rates approaching 1000 problems solutions per second are available. Comparators have latching controls, sample and hold amplifiers may be used either as an interface between the A/D multiplexer or in stand-alone operation, D/A converters may be used as digital altenators. Sample and hold amplifiers may also be used as high speed integrators.

Four capacitor selected integration time scales are available; 1.0, .1, .01 and .001 microfarad.

3. TRAINING ACTIVITIES

A formal credit course in hybrid computation (EGR 630) has been offered twice with a total of 41 students for credit and some auditors. Two National Science Foundation Advanced Science Seminars in 1969 and 1970 with a total of 45 faculty, recruited nation-wide and several from the University of Houston, included a two-week course in analog/hybrid computation. Typical laboratory/homework exercises in EGR 630 are included in Appendix A.

4. RESEARCH IN PROGRESS

Principal investigators whose projects are using the facility include:

1. Dr. C. F. Chen, Department of Electrical Engineering, "Multivariate control systems;" M.S. Thesis of Terry Edgerton enclosed with this report.

2. Dr. S. B. Childs, Department of Mechanical Engineering, "Hybrid computer techniques for the solution of multi-nodal networks."

3. Dr. A. E. Dukler, Department of Chemical Engineering, "Surface Renewal theories of turbulent diffusion, Turbulent diffusion by Monte Carlo Methods, Stochastic models of two-phase gas-liquid flow."

4. Dr. C. J. Huang, Assistant Dean of Faculties and Department of Chemical Engineering, "System Identification in Complex Chemical Kinetic Systems."

5. Dr. R. D. Finch, Department of Mechanical Engineering, "Acoustic excitation of railroad wheels" (Department of Transportation).

6. Dr. N. H. C. Hwang, Department of Civil Engineering, "Modeling of cardiovascular system and circulation in the brain." Technical paper with J. C. Chao enclosed with this report.

7. Dr. D. Luss, Department of Chemical Engineering, "Dynamic poisoning phenomena in catalytic synthesis."

8. Dr. R. L. Motard, Department of Chemical Engineering, "Coupled distributed parameter phenomena in chemical processes, Solution of partial differential equations." Ph.D. dissertation of H. M. Lee available on request.

9. Dr. J. M. Nash, Department of Civil Engineering, "Graphic consoles in the control, programming and operation of hybrid computers."

10. Dr. W. P. Schneider, Department of Electrical Engineering, "Optimal and adaptive control of a dynamic ship positioning system." M. S. Thesis of R. St. John enclosed with this report.

5. RELATED RESEARCH

Another research contract which uses and supports the digital component of the Hybrid System is a Project THEMIS, ONR Contract N00014-68-A-0151, activity in Information Processing Systems. Reports published under our THEMIS project are listed in Appendix B.

6. FUTURE EXPANSION

Plans for future expansion except for minor items are necessarily contingent on additional funding for which proposals are now being prepared.

Analog Computer

The analog computer was not fully expanded when purchased with a National Science Foundation grant. Additional plug in components are needed among which the more critical items are quarter-square multipliers, relays and switches and digital coefficient units. Another item in short supply is patch cords for the 13 patch panels now in use. On the larger hybrid projects we are finding that 16 digital-to-analog converters are not quite sufficient.

Another area for improvement is in arbitrary function generators of which none were included in the original machine. It now appears that the optimal solution to function generation is to use a minicomputer with continuous A/D tracking of the independent variables and D/A conversion of dependent variables. Adding such a device opens up further opportunities for parallel digital processing in the system which have not been completely assessed. For instance, it would be possible to greatly expand the digital logic capability with minimal loss in performance. The main digital computer would still be responsible for major program control and data management.

For classroom use, a remote control interface which would allow variation of parameters and display selection would greatly improve the visual content in both graduate and undergraduate courses. Such a facility will most easily be

implemented when digital coefficient units are available on the analog computer.

Digital Computer

Additions to the digital computer which would enhance the usefulness of the system include: expansion of core memory from 128K bytes to 256K bytes, expansion of disc memory from 21.75 million bytes to 116 million bytes (3 IBM 2311's to 4 IBM 2314's). The addition of a communication interface (IBM 270x) will allow network operation of the hybrid computer with the University Computing Center UNIVAC 1108 as well as data acquisition links to various laboratories, and time-sharing on the IBM 360.

Hybrid System Graphics

A high priority item for the extension of hybrid problem solving capability is a full graphics console terminal. This console would serve as a hybrid user interface to display problem data in real time, to program and control the operation of the hybrid system, including such functions as set-up, static test, coefficient setting and program initialization. With the speed of a modern hybrid system, the traditional man-in-the-loop concept of analog operation is no longer viable without slowing down the process considerably. User control must

be moved to a higher supervisory level in the system. A graphic interface can display system status and data much faster and more intelligibly than conventional I/O devices thereby improving communication speed and accelerating response to human interaction.

7. SYSTEM DOCUMENTATION

Aside from maintenance and specification manuals for the system, basic user documentation enclosed with this report include:

1. Data Acquisition Multiprogramming System-DAMPS-Version 2, H20-0537, IBM Corporation.
2. SS-100 Analog/Hybrid Computer Patching Reference Manual, HS/40001.0, Hybrid Systems, Inc.
3. 1044 Hybrid Linkage Unit-Programming Reference Manual, 6030.2, Hybrid Systems, Inc.
4. Hybrid Executive - Programming Guide, 6033.2, Hybrid Systems, Inc.
5. ANASET - Analog Set-up and Static Check Program; revised at University of Houston from material prepared by Bell Helicopter Co., Fort Worth.
6. AUTOTRAK - Execution Time Interactive Facility, Bell Helicopter Co., Fort Worth.
7. DAMPS Routines for Hybrid Programming; extensions to DAMPS prepared by Bell Helicopter Co., Fort Worth.
8. DAMPS and FORTRAN - Programming Notes, Bell Helicopter Co., Fort Worth.
9. Job Control Language for Hybrid Programmers, Memorandum, University of Houston, Engineering Systems Simulation Lab.

10. 'X' Series Hybrid Executive Routines - a set of simplified immediate execution hybrid I/O routines for beginning users and simple hybrid programs, University of Houston, Engineering Systems Simulation Lab.

The programming language for the user is PS/44 FORTRAN IV for the IBM 360/44 basically equivalent to OS/360 FORTRAN IV G.

APPENDIX A

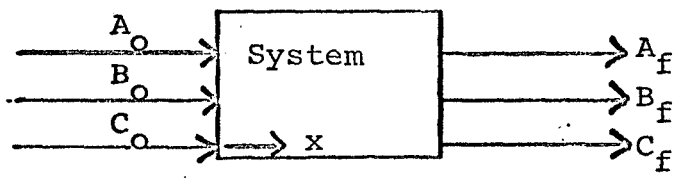
Typical laboratory/homework assignments
EGR 360 Hybrid Computation

UNIVERSITY OF HOUSTON

EGR 630 Hybrid Computation

System Identification Problem

A distributed parameter system receives as input three variables A_o, B_o, C_o and produces as output A_f, B_f, C_f. The governing differential equations for the system are assumed to be:



$x_{max} = 1$

$$\frac{dA}{dx} = -k_1 A + k_2 B$$

$$\frac{dB}{dx} = k_1 A - k_2 B + k_4 C - k_3 B$$

$$\frac{dC}{dx} = k_3 B - k_4 C$$

where x is the scaled independent space variable.

The maximum values of A, B, C are all 1.0, but scaling should be carefully considered anyway.

The unknown parameters k_1, k_2, k_3, k_4 are to be determined from a set of ten experimental runs on the physical system.

OBSERVED DATA

Run	A_o	B_o	C_o	A_f	B_f	C_f
1	.80	.10	.10	.0286	.1333	.8392
2	.75	.15	.10	.0277	.1317	.8415
3	1.0	0	0	.0323	.1410	.8282
4	.10	0	.90	.0111	.1032	.8860
5	0	.50	.50	.0111	.1042	.8850
6	.25	.50	.25	.0176	.1150	.8685
7	.95	0	.05	.0317	.1390	.8300
8	.40	.20	.40	.0200	.1192	.8610
9	0	0	1.	.0094	.0988	.8923
10	.50	0	.50	.0211	.1200	.8600

Develop a hybrid program which iteratively generates the ten experimental conditions and computes a global error criterion ($y_i = A_f$, B_f or C_f)

$$\Sigma(y_i \text{ obs} - y_i \text{ calc})^2$$

This error is transmitted through a DAC to the analog computer digital voltmeter. DVM mode on PP and DAC output is wired to DVM IN. Initial conditions A_o, B_o, C_o are reset by the digital between every repetitive operation cycle. The "best" set of k_i can be found by manually adjusting the pots which represent them and minimizing the error.

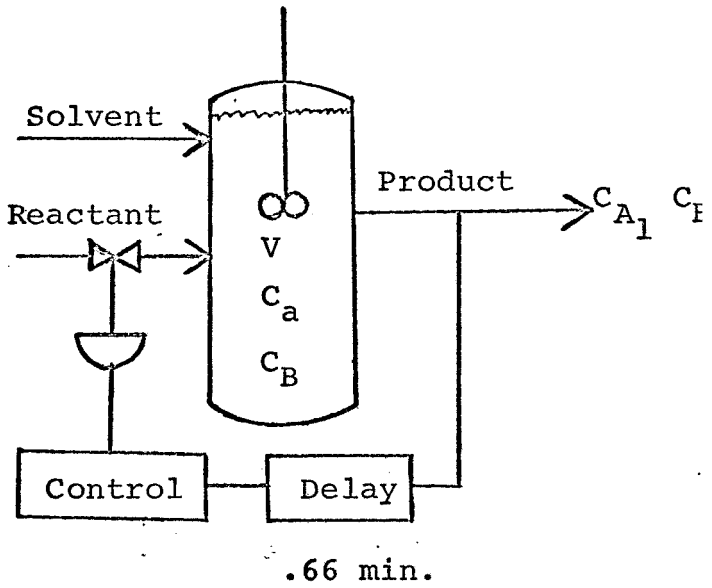
Develop a hybrid program which automatically searches for the best set of k_i.

University of Houston

EGR 630 HYBRID COMPUTATION

Problem 2

A chemical reactor receives a controlled input of a solvent, and the flow of reactant is controlled by the concentration of product, B which must not exceed 1.3 lb/gal.



Dynamic material balances on A and B are as follows:

$$Q_S C_{A_o} + \rho_A Q_A - C_A (Q_S + Q_A) - kC_A^2 V = V \frac{dC_A}{dt} - C_B (Q_S + Q_A)$$

$$+ kC_A^2 V = V \frac{dC_B}{dt}$$

where:

Q_S = Flow of solvent = 50 gal/min

Q_A = flow of reactant = 10 gal/min

C_{A_o} = solvent concentration of A = 0.4 lb/gal

ρ_A = reactant density = 9.04 lb/gal

k = reaction rate constant = 0.1418 gal/(lb - min)

V = reactor volume = 600 gal

Steady State Values:	Maximum Values:
Q_S = 50 gal/min	Q_S = 70 gal/min
Q_A = 10 gal/min	Q_A = 20 gal/min
C_A = 0.84 lb/gal	C_A = 1.5 lb/gal
C_B = 1.0 lb/gal	C_B = 1.3 lb/gal
C_{A_o} = 0.4 lb/gal	C_{A_o} = 2.0 lb/gal

The controller is a derivative-integral controller where

$$\frac{M(s)}{E(s)} = \frac{K_c(T_D s + 1)}{(T_I s + 1)}$$

and

$$M(s) = L\ \{m(t)$$
$$E(s) = L\ \ e(t)$$
$$m(t) = Q_A - A_A \text{ (steady state)}$$
$$e(t) = C_B(\text{set}) - C_B$$

$K_c = 100$; $T_I = 15.$; $T_o = 3.3$

The delay due to sampling may be approximated by a second order Padé form (p. 227 EAI Handbook of Analog Computation).

Plot the system response, C_B , on a strip chart recorder for a step change in C_B(set) from 1.0 to 1.1 . Also for a change in solvent flow from 50. to 70 gal/min.

APPENDIX B

Related Research in Information Processing Systems

ONR Contract N00014-68-A-0151

INFORMATION PROCESSING SYSTEMS

Office of Naval Research

Contract: N00014-68-A-0151

R. L. Motard, Project Manager
Cullen College of Engineering
University of Houston
Houston, Texas 77004

TECHNICAL REPORTS PUBLISHED

Report Number	*Author-Title*	*DDC AD Number*
RE 1-68	Luckinbill, Dennis L. and S. Bart Childs *Inverse Problems in Partial Differential Equations* Mechanical Engr. Department (97 pages, Aug., 1968)	691390
RE 2-68	Childs, S. Bart *Identification of Systems* Mechanical Engr. Department (34 pages, Aug., 1968)	691391
RE 1-69	Ingels, Don M. *A Digital Computer Program to Determine the Expected Mean Squares For An Analysis of Variance* Chemical Engr. Department (37 pages, Mar., 1969)	691392
RS 1-69	Friedman, Daniel P., David C. Dickson, Terrence W. Pratt, John J. Fraser *GRASPE 1.5, A Graph Processor and Its Application* Computer Science Dept. (143 pages, Aug., 1969)	696673
RE 2-69	Hough, H. V. and R. L. Motard *List Structures in Engineering Design* Chemical Engr. Department (203 pages, Mar., 1969)	691393
RS 2-69	Henderson, Helen and Elliott I. Organick *Considerations In The Design of An XDS Sigma 7 Multics* Computer Science Dept. (52 pages, Sept., 1969)	
RE 3-69	Newhouse, Albert and Robert A. Sibley, Jr. *On The Use Of Very Low Cost Terminals* Computer Science Dept. (23 pages, Mar., 1969)	691398
RE 4-69	Bryan, John L. and S. Bart Childs *Numerical Solutions of Diffusion-Type Equations* Mechanical Engr. Department (61 pages, Jun., 1969)	691399
RE 5-69	Baker, Billy E. and S. Bart Childs *Identification of Material Properties of A Nonlinearly Elastic Material by Quasilinearization* Mechanical Engr. Department (41 pages, Jun., 1969)	691401

Report Number	Author-Title	DDC AD Number
RE 6-69	Motard, R. L. *Selection of A Hybrid Computing Facility* Chemical Engr. Department (50 pages, May, 1969)	--
RE 7-69	Childs, S. Bart, et al. *QUASI, Solution of Multipoint Boundary Value Problems of Quasilinear Differential Equations* Mechanical Engr. Department (111 pages, Sept., 1969)	
RE 8-69	Ozkul, Osman S. and Charles E. Donaghey *STIL Users Manual* Indus. & Systems Engr. (29 pages, Aug., 1969)	693521
RE 9-69	Motard, R. L., H. M. Lee and R. W. Barkley *CHESS User's Guide, Second Edition* Chemical Engr. Department (123 pages, Aug., 1969)	693522
RE 10-69	Morefield, Charles L. and Bart Childs *A Problem In Optimal Reorientation of Asymmetric Space Vehicles* Mechanical Engr. Department (36 pages, Aug., 1969)	710788
RE 11-69	Singh, Darshan, Charles E. Donaghey and Prem B. Dewan *UHELP Users Manual* Indus. & Systems Engr. (22 pages, Oct., 1969)	698189
RE 12-69	Donaghey, Charles E. and Osman S. Ozkul *STIL Systems Manual* Indus. & Systems Engr. (114 pages, Sept., 1969)	712517
RE 13-69	Williams, Ray A. and Ernest J. Henley *Multicomponent Vapor-Liquid Equilibria Computation* Chemical Engr. Department (277 pages, May, 1970)	714826
RE 1-70	Paskusz, G. F. *UHCAP Users Manual* Electrical Engr. Department (61 pages, May, 1970)	708051
RS 1-70	Foster, Derrell V. and Robert A. Sibley, Jr. *Touchtone Teleprocessing On The Interdata 4* Computer Science Dept. (266 pages, Jan., 1970)	713241
RE 2-70	Cannon, Don L. *Semi-Conductor Integrated Circuit Component Models* Electrical Engr. Department (36 pages, Jul., 1970)	713475

Report Number	*Author-Title*	*DDC AD Number*
RS 2-70	Lenahan, John J. *System-State Model Theory and Implementation* Computer Science Dept. (148 pages, Oct., 1970)	715111
RE 3-70	Cannon, Don L. *Dielectric Field Computations In Semiconductor Oxide Integrated Circuits* Electrical Engr. Department (79 pages, Jul., 1970)	713476
RS 3-70	Hsu, Ju-Tung and Albert Newhouse *Strachey's General Purpose Macrogenerator In Fortran* Computer Science Dept. (48 pages, Sept., 1970)	
RE 4-70	Ingels, Don M. and R. L. Motard *PRODYC, A Simulation Program For Chemical Process Dynamics and Control* Chemical Engr. Department (259 pages, Sept., 1970)	714250
RE 5-70	Sumner, Maurice N. and E. J. Henley *A Translator for Dynamo To CSMP in PL/I* Chemical Engr. Department (75 pages, Sept., 1970)	
RE 6-70	Tarbet, Denton and George S. Dawkins *A Comparison Of Techniques To Determine The Mode Of A Unimodal Function* Indus. & Systems Engr. (26 pages, Oct., 1970)	
RE 7-70	Mears, Will and George S. Dawkins *Comparison of Integer Programming Codes* Indus. & Systems Engr. (29 pages, Oct., 1970)	714827
RE 8-70	St. John, Richard Jr. and William P. Schneider *A Hybrid Computer Study Of A Dynamic Ship Positioning System* Electrical Engr. Department (101 pages, Oct., 1970)	

IBM

Application Program

H20-0537-1

Quinn

Data Acquisition Multiprogramming System/360 Model 44 (DAMPS) (360A-CX-20X) Version 2

Program Description Manual

This publication contains detailed information for preparing programs to be executed under DAMPS Version 2. It discusses the system's requirements, its capabilities, and the facilities that are available as a result of the DAMPS extensions to the System/360 Model 44 Programming System.

Second Edition (March 1969)

This edition is a major revision obsoleting H20-0537-0. It applies to Version 2, Modification Level 0, of Data Acquisition Multiprogramming System/360 Model 44 (360A-CX-20X) and to all subsequent versions and modifications until otherwise indicated in new editions or Technical Newsletters.

Changes are continually made to the specifications herein. Therefore, before using this publication, consult the latest System/360 SRL Newsletter (N20-0360) for the editions that are applicable and current.

Copies of this and other IBM publications can be obtained through IBM branch offices. Address comments concerning the contents of this publication to: IBM, Technical Publications Department, 112 East Post Road, White Plains, N.Y. 10601.

© Copyright International Business Machines Corporation 1968, 1969

CONTENTS

1. Introduction 1

1.1 Objectives. 1
1.2 Machine-Oriented Concepts 1
1.3 Programming-Oriented Concepts . . 1
1.3.1 Background Job (BGJ) 2
1.3.2 Real-Time Task (RTT) . . . 2
1.3.3 Foreground Task (FGT). . . 2
1.3.4 Real-Time Job (RTJ) 3
1.4 System-Oriented Concepts 3

2. Language 7

2.1 Supervisor Calls 7
2.1.1 IDENTRTJ (SVC 49)--Identify an RTJ 7
2.1.2 SPABRTN (SVC 48)--Specify Abort Routine 7
2.1.3 MQUEUE (SVC 50)--Enter Request for FGT into Foreground Queue . . . 7
2.1.4 MDEQUE (SVC 51)--Delete Previous Request for FGT from Foreground Queue 8
2.1.5 DONE (SVC 39)--Terminate an FGT 9
2.1.6 ATTACH (SVC 32)--Assign an RTT to a PIL . . 9
2.1.7 CONVERT (SVC 36)--Convert Uncontrolled RTT to Controlled Mode . . 9
2.1.8 ALLOW (Subroutine)--Allow an RTT to Issue a CONVERT 10
2.1.9 RETURN (SVC 35)--Terminate an RTT 10
2.1.10 DISPILS (SVC 34)--Disable Priority Interrupt Levels 10
2.1.11 ENPILS (SVC 33)--Enable Priority Interrupt Levels 10
2.1.12 RTCANCEL (SVC 47)--Cancel an RTJ. 11
2.1.13 EORTJ (SVC 46)--End of an RTJ 11
2.1.14 RTIO (SVC 43)--Execute Real-Time I/O . . 11
2.1.15 MASK/UNMASK (SVC 44/45) Control I/O Interrupts 11
2.1.16 RDDWORD (SVC 40)--Read Direct Word 11
2.1.17 WTDWORD (SVC 41)--Write Direct Word 12
2.1.18 INHIBIT BGJ (SVC 42). . . . 12
2.1.19 RELEASE BGJ (SVC 52) . . 12
2.1.20 QUEUE (SVC 37)--Enter Request for FGT into Foreground Queue 0 12
2.1.21 UNQUEUE (SVC 38)--Delete Request for FGT from Foreground Queue 0 . . 13

2.2 FORTRAN Subroutines 13
2.2.1 FIDRTJ--Identify an RTJ. . 13
2.2.2 MQUEUE--Enter Request for FGT into Foreground Queue 13
2.2.3 MDEQUE--Delete Request for FGT from Foreground Queue 14
2.2.4 FDONE--Terminate an FGT. 14
2.2.5 FATACH--Assign an RTT to a PIL 15
2.2.6 FRETRN--Terminate an RTT. 15
2.2.7 FDSPIL--Disable Priority Interrupt Levels 15
2.2.8 FENPIL--Enable Priority Interrupt Levels 15
2.2.9 FEORTJ--End of an RTJ. . 16
2.2.10 FCANCL--Cancel an RTJ. . 16
2.2.11 FRCBSU--Request Control Block Set Up 16
2.2.12 FRTIO--Execute Real-Time I/O 17
2.2.13 FMSK--Mask I/O Interrupts. 17
2.2.14 FUNMSK--Unmask I/O Interrupts. 18
2.2.15 FCHECK--Check Progress of I/O Operations 18
2.2.16 FTREAD--Read Real-Time Magnetic Tape. 18
2.2.17 FTWRIT--Write Real-Time Magnetic Tape 19
2.2.18 FTWTM--Write End-of-File on Real-Time Magnetic Tape 19
2.2.19 FTREW--Rewind Real-Time Magnetic Tape 19
2.2.20 FTRUN--Rewind and Unload Real-Time Magnetic Tape 19
2.2.21 FDIG--Digital Input on 1827 DCU 19

2.2.22 FAIRAN--Random Analog Input on 1827 DCU 20
2.2.23 FAISEQ--Sequential Analog Input on 1827 DCU 21
2.2.24 FDAO--Digital/Analog Input on 1827 DCU 21
2.3 Assembler Language Subroutines .. 22
2.3.1 FRCBSU--Request Control Block Set Up 22
2.3.2 FRTIO--Execute Real-Time I/O 23
2.3.3 FCHECK--Check Progress of I/O Operations 23
2.3.4 FUNMSK--Unmask I/O Interrupts 24
2.3.5 FMSK--Mask I/O Interrupts 24
2.3.6 FTREAD--Read Real-Time Magnetic Tape 24
2.3.7 FTWRIT--Write Real-Time Magnetic Tape 24
2.3.8 FTWTM--Write End-of-File on Real-Time Magnetic Tape 24
2.3.9 FTREW--Rewind Real-Time Magnetic Tape 24
2.3.10 FTRUN--Rewind and Unload Real-Time Magnetic Tape 24
2.3.11 FDIG--Digital Input on 1827 DCU 25
2.3.12 FAIRAN--Random Analog Input on 1827 DCU 25
2.3.13 FAISEQ--Sequential Analog Input on 1827 DCU 26
2.3.14 FDAO--Digital Analog Input on 1827 DCU 26

3. Programming Techniques 27

3.1 Assembler Language Programming 27
3.1.1 General Register Usage .. 27
3.1.2 Real-Time Job 27
3.1.3 Foreground Tasks (FGT).. 28
3.1.4 Controlled Real-Time Task 29
3.1.5 Uncontrolled Real-Time Task 30
3.1.6 Timer Interrupt Routine 32
3.2 Input/Output 32
3.2.1 Requesting I/O 32
3.2.2 Real-Time Task Input/Output 32
3.2.3 Channel Queuing 33
3.2.4 1827 Data Control Unit--Programming Notes 33
3.2.5 Tapes as a Real-Time Device 34
3.2.6 Request Control Block ... 34
3.3 FORTRAN 36
3.3.1 MQUEUE 38
3.3.2 FQUEUE 38
3.3.3 FORTRAN Input/Output... 38
3.3.4 Overlay 39

4. System Operation 40

4.1 Concepts 40
4.2 Preparing Real-Time Jobs 40
4.2.1 RTLOAD--Operator Command 40
4.2.2 RTCNCL--Operator Command 40
4.2.3 RTJ--Job Control Statement 40
4.2.4 RTJOB--Job Control Statement 40
4.2.5 R--Linkage Editor Parameter 40
4.2.6 ROOR--Linkage Editor Parameter 40
4.3 Operation of Two-Partition System 40
4.4 Operation of Single-Partition System 41
4.5 Allocation of System Units 41
4.6 Device Assignment 41

5. Error Handling 42

5.1 Machine Check 42
5.2 Program Interruptions 42
5.3 Input/Output Errors 43
5.4 System Diagnostics 43

6. Minimum System Configuration 44

6.1 Notes on Configuration 44
6.2 Programming System 44

Appendix A: Special Error Codes 45

Appendix B: Bootstrap Words 51

Appendix C: DAMPS SVC Table 52

Appendix D: DAMPS Version 2 Storage Allocation 53

Appendix E: FORTRAN Data Set/ Symbolic Unit Table 54

Appendix F: Sample Programs 55

1. Real-Time Tasks, Foreground Tasks, and Timer Interrupt Routines--Assembler Language . . . 55
2. Real-Time Input/Output--Assembler Language 61
3. Real-Time Input/Output Using Subroutines--Assembler Language 63
4. Reentrant Computation--FORTRAN Language 66
5. System Test--FORTRAN Language . 69
6. Priority Interrupt Test--FORTRAN Language 72
7. Analog Input Sample Program--FORTRAN Language 73
8. Double Buffer Sample Program--FORTRAN Language 74
9. Digital Input Sample Program--FORTRAN Language 75
10. Digital/Analog Output Sample Program--FORTRAN Language . . . 76
11. Plot Mathematical Curves--FORTRAN Language 76
12. Plot a Four-Leaf Rose--FORTRAN Language 78

Bibliography 79

1. INTRODUCTION

This document assumes that the user is familiar with the related System/360 Model 44 Programming System and equipment. For a complete outline of the prerequisite manuals, please refer to the bibliography.

1.1 OBJECTIVES

DAMPS is an operating system designed to support the System/360 Model 44 in a variety of scientific and engineering applications. By partitioning storage, DAMPS is able to support two completely independent operations. The background partition supports a normal Model 44 P/S job stream containing routine engineering and data processing applications, while the foreground partition supports the real-time online applications. It allows the user to take advantage of the System/360 Model 44's computing power while also using its capability for rapid response to external events. A high degree of machine independence in the system helps the user to take advantage of the throughput and response without close attention to the computer's characteristics.

The system has been developed as an extension of the System/360 Model 44 P/S. In addition to using the special DAMPS features, the real-time programmer can take advantage of all of the facilities in the System/360 Model 44 P/S. The extensions to the Model 44 P/S provide the foreground partition with the capability to:

- Identify priority interrupt routines to operate in immediate response to priority interrupts
- Accomplish a broad range of processing and input/output at the priority interrupt level
- Schedule foreground routines to operate in priority sequence while no priority routines are active

At the same time, the background partition is capable of executing any job compatible with the Model 44 P/S.

From an external point of view, the extensions in the foreground partition appear as new supervisor calls (SVC's) in the System/360 Model 44 P/S. To facilitate FORTRAN programming, a companion set of FORTRAN subroutines has been added to the library. Jobs in both partitions are initiated and terminated independently.

1.2 MACHINE-ORIENTED CONCEPTS

The performance of DAMPS is based on two programming techniques involving special equipment features. The first, which is based on the Priority Interrupt special feature, establishes program tasks to operate in immediate response to external events. In this way, the system is always processing the highest level of active priority. The second, which is based on the high-resolution interval timer, assures that foreground processing is properly rescheduled after periods of priority interrupt processing.

Special action must be taken by the supervisor to support priority interrupt processing. If the task in progress at the time of a priority interrupt is to be resumed later, the status of the computer must be preserved. In order to accomplish this, the status must be saved at interrupt time and restored at interrupt completion. The DAMPS system normally accomplishes this preservation by storing and restoring the status of the computer for all priority interrupts. Because DAMPS assumes that all priority interrupt routines will compute and execute input/output, all of the related registers and words are preserved. The basic data involved are:

General registers (16)
Floating point registers
Channel status word
Channel address word
Old input/output program status word (PSW)

When the priority interrupt routine is set up, several options are available that give the user a choice as to the amount of data to be preserved.

1.3 PROGRAMMING-ORIENTED CONCEPTS

DAMPS runs two jobs simultaneously--one background job (BGJ) and one real-time job (RTJ). Each job is provided a fixed partition--the BGJ runs in the background partition, and the RTJ runs in the foreground partition. The background partition is compatible with the Model 44 P/S. Any job that will run on the Model 44 P/S supervisor will also run in the background, provided that enough space has been allocated for the partition. The foreground partition is reserved for RTJ's. Each RTJ is composed of real-time tasks (RTT's) and foreground tasks (FGT's).

Two modes of operation are available under DAMPS. The partitions run one RTJ and one BGJ simultaneously, or a single large RTJ. The latter mode is comparable to running an RTJ under DAMPS Version 1.

These four concepts--the BGJ, the RTT, the FGT, and the RTJ--are the fundamental concepts of DAMPS and are discussed in detail in the following paragraphs.

1.3.1 Background Job (BGJ)

The background partition functions just as if Model 44 P/S were in control. However, it is multiprogrammed at the lowest priority. It executes only while there is no demand for the required facilities from the foreground partition. Any standard Model 44 P/S job can be executed provided that:

1. It is not time dependent.
2. It fits in the allocated background partition.

The time-dependency restriction follows from multiprogramming--the length of time a given set of code will take to execute cannot be predicted. The partition allocations of the distributed system can be modified as described in the DAMPS System Manual.

Any type of data processing or scientific application program can be executed in the background partition. However, one of the key uses of the background normally is preparation of RTJ's. All RTJ's are fetched from the phase library, where they were placed by a Model 44 P/S job from the background. All jobs are assembled or compiled, link-edited, and placed in the system's single phase library by a BGJ. The phase library is the only library that both partitions can access. Care should be taken to avoid changing a particular entry from the background partition while the foreground partition is attempting to fetch it.

1.3.2 Real-Time Task (RTT)

An RTT is a user-written subroutine that receives control as a direct result of a priority interrupt. The RTT must subsequently relinquish control by executing a special supervisor call. While an RTT is in control, it can be interrupted only by a higher level priority. When the priority interrupt occurs, its associated new PSW becomes the current PSW, and a hardware latch is set precluding other than higher level interrupts. This latch is reset by a Load PSW special instruction, normally executed by the supervisor, after which the next lower level of processing resumes. In this manner, the highest active level of priority is always in control of the computer.

The RTT concept is designed for processing that must be accomplished on a priority interrupt basis in direct and immediate response to external events. The task associated with the highest level of priority will receive immediate attention at all times. The precise level of priority selected for a given RTT is a function of the application in which the RTT takes part. Any function performed that does not require direct response should be relegated to an FGT.

Because RTT's are executed on a priority interrupt basis, they are not allowed to use the supervisor facilities of the Model 44 P/S--these facilities are not reentrant. Therefore, a programmer coding in Assembler Language cannot use the Model 44 P/S SVC routines, and the FORTRAN programmer cannot use FORTRAN input/output statements. A complete set of new DAMPS supervisor calls and FORTRAN subroutines has been added to provide a full data processing capability at the priority interrupt level. The new facilities are described in detail in section 2.

1.3.3 Foreground Task (FGT)

An FGT is a user-written subroutine that is scheduled for execution at the user's request and initiated by the supervisor. The programmer places the request and the desired priority in one of four queues by a special supervisor call in either an RTT or an FGT.

FGT's are processed while no priority interrupts are active. Each queue has an inherent priority. The system tends to process FGT's from the highest level queue in priority sequence. The current FGT must relinquish control by executing a special supervisor call before the next FGT in the same queue can be initiated. Any time a "wait" is encountered, the supervisor processes FGT's from the next lower queue. When the wait is satisfied, control is returned to the higher priority queue. While no FGT's are active, control is given to the background partition.

The FGT is designed for processing that need not be executed in immediate response to an external event. A typical application of multiple queues involves only two queues. The higher level queue contains requests for execution of I/O-bound FGT's, while the lower contains requests for compute-bound FGT's. In this example, maximum efficiency of FGT processing is achieved. A variation of this application could contain a single, nonterminating FGT in the higher queue and a series of compute-bound FGT's in the lower queue. Another variation could involve a single nonterminating compute-bound FGT in the lower queue. The latter variation is the equivalent of straight multiprogramming between tasks, or multitasking. The variations are endless. However, remember that DAMPS is a priority-based system, always tending toward processing the highest priority work. If the demand for high-priority work is continuous, the lower level work may never get done. For instance, if a queue containing requests for compute-bound FGT's never empties, lower queues and background will never

be serviced. If this condition is reached, work should be redistributed. If it cannot be redistributed, the capacity for the job probably is insufficient.

FGT's are allowed to use all of the Model 44 P/S facilities, in addition to the new DAMPS supervisor calls and FORTRAN subroutines. The FGT's within a queue can freely share subroutines. Subroutines shared by FGT's in different queues must be reentrant. This restriction applies to the IBCOM routines in FORTRAN and implies that the standard FORTRAN input/output statements are generally restricted to a single queue. This restriction does not apply to the DAMPS FORTRAN subroutines.

1.3.4 Real-Time Job (RTJ)

An RTJ is a collection of RTT's and FGT's that a user wishes to handle as an independent program. No system constraint exists on the number of FGT's in an RTJ; however, no more than 32 RTT's are allowed in any RTJ. This constraint is imposed by the hardware.

An RTJ can be executed in the foreground partition, where it is multiprogrammed with a BGJ, or it can be run by itself as a single, large RTJ using both partitions. In either case, the RTJ can contain only one job step. RTJ's are prepared using the assembler, compiler, and linkage editor of the Model 44 P/S, which execute in the DAMPS background partition.

An RTJ is initiated in the foreground partition from the console typewriter. It is multiprogrammed with a BGJ if one is currently running or subsequently initiated. The initial phase of the job must be in the phase library at load time, and subsequent phases must be there when they are fetched by the initial or subsequent phases. Phases are placed in the phase library by a BGJ (that is, the linkage editor), the only restriction being that a phase should not be undergoing modification while it is being fetched.

An RTJ to be run in both partitions is initiated from the normal system input unit just as if it were a Model 44 P/S job, with two distinctions: it can consist of only one job step, and an RTJOB job control statement is used in place of JOB. Multiple-step JOB's cannot be run in this mode--only single-step RTJOB's.

An RTJ in either mode will act like a normal Model 44 P/S job until it executes a special supervisor call allowing it to use the DAMPS facilities. These facilities are described in detail in section 2. RTJ's may be coded in either FORTRAN or Assembler Language. The appropriate techniques are described in detail in section 3.

1.4 SYSTEM-ORIENTED CONCEPTS

When DAMPS is functioning in the two-partition mode, it functions on three levels: the priority interrupt level, the foreground level, and the background level.

When no priority interrupt levels are active, the system is functioning at the foreground level or the background level, that is, executing an FGT or the BGJ. When there is no demand for FGT processing, the system will execute in the background. This condition arises when the task queues are empty or all the active FGT's are waiting. As soon as a wait condition is cleared for an FGT, control is returned to the foreground level. The supervisor automatically stores and restores the appropriate registers during this status switching.

When a priority interrupt is taken, control is transferred to the associated RTT, and the status of the computer is saved. At the same time a hardware latch is set that holds off interrupts from the same or lower levels while allowing those from a higher level to be taken. When the RTT completes its processing, the associated hardware latch is reset, and control is given to the next lower level of priority requiring service. When the next lower level is a priority interrupt that was held off in the interim, the interrupt is taken. When the next lower level was a formally interrupted RTT, FGT, or BGJ, the computer status is restored, and control is generally returned to the point of interruption. If the BGJ was interrupted, and in the interim a demand for foreground processing was generated, control is returned to the foreground immediately. In this manner, the system is always executing the RTT associated with the highest level of priority requiring service.

Two examples are provided to illustrate the principles of DAMPS operation.

Example 1: Assume that the current RTJ contains two FGT's and five RTT's. At the opening of the example, FGT A has started executing at

priority 1, and an additional FGT request is in foreground task queue 1 as follows:

FGT	Priority
B	2

The RTT's are associated with the PIL's as follows:

RTT	PIL
A	30
B	29
C	3
D	2
E	1

Assume that the following sequence of interrupts occurs:

30, 3, 1, 2, 29

and that there is an interval of one time unit between each interrupt. Assume that each FGT and RTT requires two units of execution time, and that the first interrupt occurs after one unit of problem time.

The sequence of execution is depicted in Figure 1. Execution time of the problem is 14 units of time. The key events are outlined in the following table.

Units of Time	Events
0	FGT A is initiated.
1	PIL 30 is taken; FGT A status is saved; RTT A is initiated.
2	PIL 3 is taken; RTT A status is saved; RTT C is initiated.
3	PIL 1 is taken; RTT C status is saved; RTT E is initiated.
4	PIL 2 occurs.
5	PIL 1 resets; PIL 2 is taken; RTT D is initiated; PIL 29 occurs.
7	PIL 2 releases; RTT C is restored; RTT C continues.
8	PIL 29 is taken; RTT B is initiated.
10	PIL 29 releases; RTT A is restored; RTT A continues.
11	PIL 30 releases; FGT A is restored; FGT A continues.
12	FGT A completes; FGT B initiates.
14	FGT B completes; problem ends.

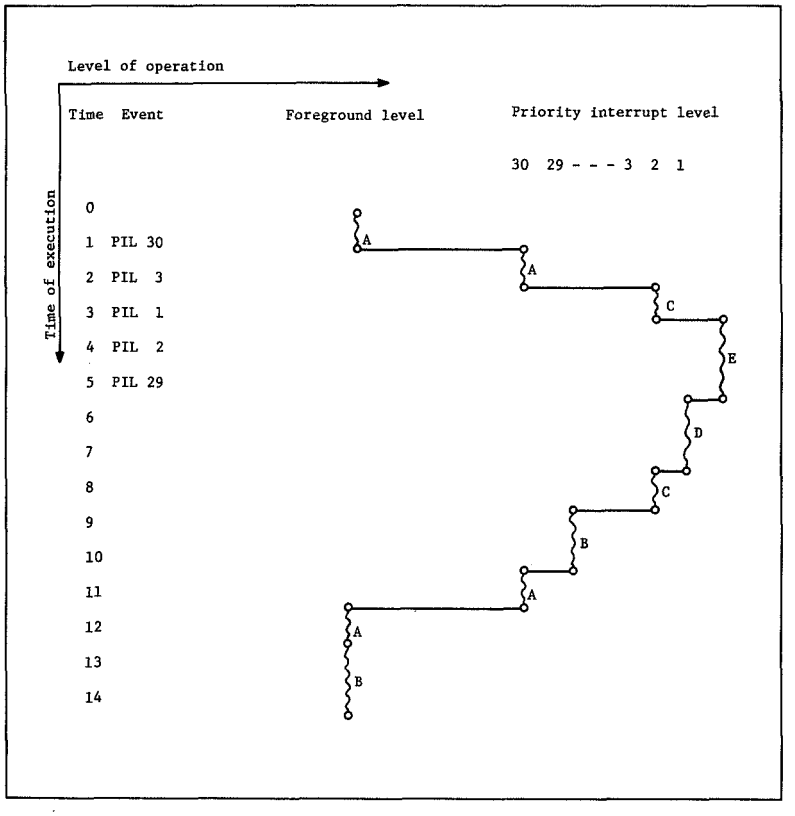


Figure 1. Sequence of execution, Example 1

Example 2: Assume that the RTT's of Example 1 had additional requests for FGT's A, B, and C into foreground task queues 0 and 2 as follows:

Queue 0		Queue 1 (per Example 1)		Queue 2	
FGT	Priority	FGT	Priority	FGT	Priority
A	1	A	1	B	1
B	2	C	2	C	2

Assume that each FGT requires two units of execution time.

Assume also that each FGT issues a wait after one unit of execution time, and that each wait lasts five units.

The sequence of execution is depicted in Figure 2. The key events are outlined in the following table.

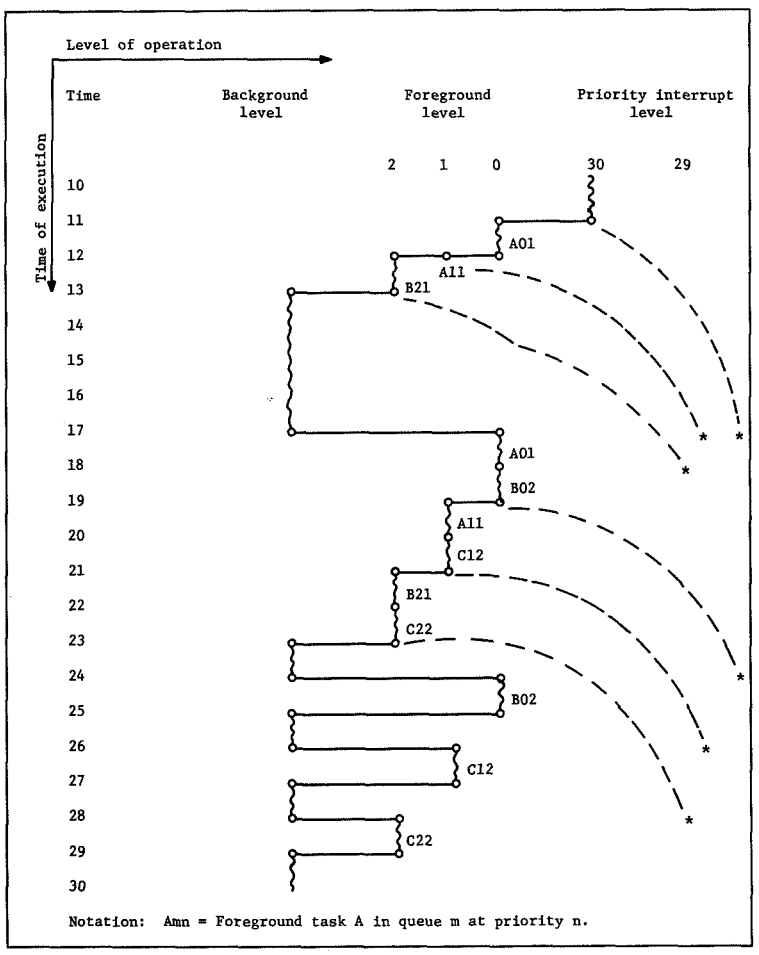


Figure 2. Sequence of execution, Example 2

Time	Events
11	PIL 30 releases; FGT A01 is initiated.
12	A01 status is saved; FGT A11 is restored; FGT A11 status is saved; FGT B21 is initiated (All had executed for 1 time unit during Example 1).
13	FGT B21 status is saved; BGJ is restored.
17	BGJ status is saved; FGT A01 is restored; A01 wait is satisfied; A11 wait is satisfied.
18	FGT A01 terminates; FGT B02 is initiated; B21 wait is satisfied.
19	FGT B02 status is saved; FGT A11 is restored.
20	FGT A11 is terminated; FGT C12 is initiated.
21	FGT C12 status is saved; FGT B21 is restored.
22	FGT B21 is terminated; FGT C22 is initiated.
23	FGT C22 status is saved; BGJ is restored.
24	BFJ status is saved; FGT B02 is restored; B02 wait is satisfied.
25	FGT B02 is completed; BGJ is restored.
26	BGJ status is saved; FGT C12 is restored; C12 wait is satisfied.
27	FGT C12 is terminated; BGJ is restored.
28	BGJ is restored; C22 is restored; C22 wait is satisfied.
29	FGJ C22 is terminated; BGJ is restored.
30	BGJ continues until some RTT reschedules foreground.

2. LANGUAGE

In addition to the features of Model 44 P/S, the following extensions are available for use in RTJ's. They were designed to meet the real-time, online requirements of DAMPS applications. If any of these special calls are executed in a BGJ, the job will be canceled.

The extensions to the System/360 Model 44 P/S that comprise the additional features for RTJ's are based on additional SVC's. The new SVC's have a companion set of FORTRAN subroutines that make the extensions available to the FORTRAN programmer. The additional FORTRAN subroutines also have a companion set of calls that the Assembler Language programmer may use. These SVC's and FORTRAN/Assembler Language subroutines comprise the language of DAMPS and are outlined in the following three sections.

2.1 SUPERVISOR CALLS

The following paragraphs describe the additional SVC's provided by DAMPS as extensions to the Model 44 P/S supervisor.

2.1.1 IDENTRTJ (SVC 49)--Identify an RTJ

The IDENTRTJ supervisor call is used to identify a real-time job. Before this supervisor call is executed, none of the DAMPS features can be used. The job appears as a standard System/360 Model 44 P/S application. Any attempt to use a DAMPS supervisor call before execution of IDENTRTJ will cause the job to be canceled.

No parameter list is associated with the IDENTRTJ supervisor call.

2.1.2 SPABRTN (SVC 48)--Specify Abort Routine

The SPABRTN supervisor call is used to specify an abort routine to the supervisor. If an abort routine has been specified, an attempt will be made to give control to it during any cancellation resulting from machine checks, program checks, and the RTCANCEL supervisor call. Terminations resulting from the EORTJ supervisor call will not result in execution of the abort routine. Only one abort routine may be specified in each real-time job.

The abort routine receives control with the machine in the supervisor state. All channels and interrupts are masked. The system may be unable to provide any information about the nature of the problem, and the results of any supervisor calls within the abort routine are difficult to predict. The abort routine can return to the supervisor to complete the cancellation procedure by executing a BR14 instruction. At this point, however, both the RTJ and BGJ will be canceled. When SPABRTN is executed, general register 1 must contain the absolute address of the abort routine. The following hexadecimal codes are returned in the low-order byte of general register 15 after execution of the SPABRTN supervisor call.

Code	Meaning
'00'	The supervisor call has been executed successfully, and an abort routine has been established.
'04'	An invalid entry point has been specified, and an abort routine has not been established. The absolute address specified was not on a halfword boundary or was outside the limits of the application program.

2.1.3 MQUEUE (SVC 50)--Enter Request for FGT into Foreground Queue

The Multiple Queuing supervisor call is used to enter a request for execution of a foreground task, with an associated task priority, into the specified foreground queue. The requests in any one foreground queue are honored in task-priority sequence. Multiprogramming among the four foreground queues is based on queue priority.

When MQUEUE is executed, general register 1 must contain the address of a parameter list. The list contains three fullwords and is aligned on a fullword boundary. Word zero of the list contains the priority of the request and the absolute address of the entry point of the requested foreground task. Task priorities range from zero to 127, with zero being the highest priority. Word 1 of the list may contain any bit pattern and is placed in general register 1 when the foreground task is given control. Word 1 in the parameter list is for the convenience of the applications programmer and is not used by the system. Word 2 contains a fullword value that indicates which of the four foreground queues is to

receive this request. The format of the parameter list is:

Field	Contents
Word 0	
Bits 0-7	A binary number from 0 - 127 representing the priority of the request
Bits 8-31	The absolute address of the entry point of the foreground task
Word 1	Any bit pattern. This value will be contained in register 1 when the FGT receives control.
Word 2	One fullword binary value identifying the foreground queue into which this request should be entered. There are four foreground queues, identified as 0, 1, 2, or 3. The highest priority queue is zero.

After completion of the SVC MQUEUE, control is returned to the applications program with a hexadecimal return code in the low-order byte of general register 15:

Code	Meaning
'00'	The FGT was successfully placed in the foreground queue.
'0F'	An invalid priority (not 0 - 127) was contained in bits 0 - 7 of word 1 in the parameter list.
'F0'	The entry point to the foreground task was invalid. The address either was not within the real-time job partition or was not on a half-word boundary.
'FF'	The particular foreground queue was full.
'04'	The address of the parameter list was invalid. The address either was not within the real-time job partition or was not on a fullword boundary.
'08'	An invalid queue was specified (not 0 - 3).

If any invalid conditions are indicated, the request is not honored. Only if the return code is '00' was the FGT successfully placed in the appropriate queue.

2.1.4 MDEQUE (SVC 51)--Delete Previous Request for FGT from Foreground Queue

The MDEQUE supervisor call is used to delete a previous request for execution of a foreground task. The unqueue request, which is actually placed in the specified foreground queue, is honored just before the next foreground scheduling of that queue. A queue request is identified on the basis of its entry point and associated task priority, in addition to its particular foreground queue.

When MDEQUE is executed, general register 1 must contain the address of a parameter list. The list contains three fullwords aligned on a fullword boundary. The first word contains the task priority and entry point address of the request to be deleted. The second word may contain anything. It must be provided for alignment only. The third word specifies the foreground queue. Just before scheduling the next foreground task, all queue requests in the appropriate foreground queue that exactly match both the priority and entry point of the unqueue request are deleted. The format of the parameter list is:

Field	Contents
Word 0	
Bits 0-7	A binary number from 0 - 127 representing the priority of the queue request that is to be deleted
Bits 8-31	The absolute address of the entry point of the queue request that is to be deleted
Word 1	Used for alignment of the third word; may contain anything
Word 2	A binary number from 0 - 3 indicating which of the foreground queues is to be affected

After completion of the SVC, control is returned to the application program with a hexadecimal return code in the low-order byte of general register 15:

Code	Meaning
'00'	The SVC MDEQUE request was successfully placed in the appropriate foreground queue.
'0F'	An invalid task priority was contained in bits 0 - 7 of the first word in the parameter list.
'F0'	The address of the entry point to the foreground task was invalid. The address either was not contained in the real-time job partition or was not on a halfword boundary.
'FF'	The specified foreground queue was full.
'04'	The address for the parameter list was invalid. The address either was not contained in the real-time job partition or did not indicate a fullword boundary.
'08'	An invalid foreground queue was indicated (not 0 - 3).

If any invalid conditions are indicated, the request is not placed in the queue. Only if the return code is '00' was the entry made in the queue.

2.1.5 DONE (SVC 39)--Terminate an FGT

The DONE supervisor call is used to terminate a foreground task. Control is returned to the supervisor, and the foreground task request with the highest priority in the same foreground queue is initiated. If multiple requests of the same priority are found, the oldest entry is honored first.

If a DONE supervisor call is executed in any routine other than an FGT, the job will be canceled. If no DONE supervisor call is executed, no subsequent FGT requests within the same queue will be rescheduled. Also, unless a wait is executed, none of the FGT requests in lower queues and the BGJ will be executed.

2.1.6 ATTACH (SVC 32)--Assign an RTT to a PIL

The ATTACH supervisor call is used to assign a specific real-time task to a specific priority interrupt level. Subsequent priority interrupts at the given level will cause control to be transferred to the associated real-time task. The ATTACH supervisor call does not enable the priority interrupt level--this is the function of the ENPILS supervisor call.

Three options of the supervisor call are available, providing for various levels of supervisor control at priority interrupt time:

1. The user may elect to have the supervisor accomplish all the necessary housekeeping at interrupt time (a "controlled RTT").

2. The user may elect to have the real-time task assume the responsibility for housekeeping (an "uncontrolled RTT").

3. The user may elect to convert a real-time task from uncontrolled to controlled mode via the CONVERT supervisor call during the RTT execution (a "convertible RTT").

Controlled real-time tasks receive control in problem state with all necessary register saving accomplished. Uncontrolled real-time tasks receive control directly as a result of the associated priority interrupt with no registers saved. The uncontrolled real-time task is totally responsible for preserving the status of the system and is not allowed to use any supervisor calls. However, the uncontrolled RTT can convert to controlled during execution by issuing the CONVERT supervisor call. The techniques for conversion are explained in more detail in section 3.1.5.

After the CONVERT supervisor call is completed, control is returned to the real-time task in problem state, and all the system supervisor calls may be used. Subsequent priority interrupts will initiate the real-time task in the uncontrolled mode.

When ATTACH is executed, general register 1 must contain the address of a parameter list. The list must contain four fullwords and must be aligned on a fullword boundary. Word 0 contains the absolute address of the entry point of the real-time task. Word 1 of the list contains the absolute address of a 61-word save area, which must be aligned on a doubleword boundary. The first halfword of word 2 in the list contains the hexadecimal representation of the priority interrupt level. The third byte of word 2 contains an indication of the selected option, and the fourth byte is zero. Word 3 of the list contains the absolute address of a four-doubleword floating point save area aligned on a doubleword boundary. The floating point save area is optional. The format of the list is:

Field	Contents
Word 0	The absolute address of the entry point of the real-time task
Word 1	The absolute address of a 61-word save area aligned on a doubleword boundary
Word 2	
Bytes 1-2	The hexadecimal equivalent of 0 - 31 representing the priority interrupt level to be attached. Zero is the highest level of priority.
Byte 3	A hexadecimal indicator for the selected option:
	'00'-- The RTT is to be uncontrolled. The 61-word save area is not required if this option is selected.
	'0F'-- The RTT is to be initially uncontrolled but may be converted to a controlled task during execution (a convertible RTT).
	'FF'-- The real-time task is to be controlled.
Byte 4	'00'
Word 3	The address of a four-doubleword floating point save area on a doubleword boundary, or zero if floating point registers need not be saved.

Inconsistencies in the request, such as a priority level greater than 31 or an entry point outside the problem program area, will cause the job to be canceled.

2.1.7 CONVERT (SVC 36)--Convert Uncontrolled RTT to Controlled Mode

The CONVERT supervisor call is used to convert a convertible real-time task to the controlled mode. The supervisor call is used during execution of the real-time task and allows the remainder of the task

to use the facilities of DAMPS. The option for converting must have been selected in the ATTACH supervisor call at the time the task was created. Termination of the RTT will be via the RETURN supervisor call. CONVERT must be executed before other SVC's each time the RTT is activated.

The CONVERT supervisor call has no parameter list. Three procedures must be observed before execution of the supervisor call:

1. The first instruction in the real-time task must save general register 15 in the LOWWORD for a particular priority interrupt level. The absolute address set aside by DAMPS for each LOWWORD is given in Appendix B.

2. The ALLOW subroutine must be executed to tell the system which priority interrupt level is being converted. The program branches directly to ALLOW, with the return address to the RTT in general register 14 and the priority interrupt level in general register 13. The absolute address of ALLOW is given in Appendix B.

3. General registers 1 - 14 must contain the same data as they did when the priority interrupt occurred so that the system can assume responsibility for the proper restoration of these registers.

Any irregularities recognized during the CONVERT supervisor call will cause the job to be canceled. However, because uncontrolled and convertible RTT's run in supervisor state, the DAMPS supervisor is not aware that these routines are in control. Consequently, the diagnostic protection is weak.

2.1.8 ALLOW (Subroutine)--Allow an RTT to Issue a CONVERT

The ALLOW subroutine is used to allow a real-time task to execute the CONVERT supervisor call. It is required to identify the priority interrupt level of the associated real-time task to the DAMPS supervisor.

ALLOW may be executed by loading a value equal to the priority interrupt level into general register 13 and branching directly to ALLOW, with general register 14 containing the return address to the RTT. The absolute address of ALLOW is given in Appendix B.

Any irregularities recognized during ALLOW will cause the job to be canceled.

2.1.9 RETURN (SVC 35)--Terminate an RTT

The RETURN supervisor call is used to terminate a real-time task. Control is returned to the next lower real-time task, when one is active. If no RTT's are active, control is returned to the foreground or background.

If a RETURN supervisor call is executed in any routine other than an RTT, the job will be canceled. If the RETURN supervisor call is omitted, control will never be returned to the point of interruption.

2.1.10 DISPILS (SVC 34)--Disable Priority Interrupt Levels

The DISPILS supervisor call is used to disable priority interrupt levels. All of the levels indicated in an individual call are disabled simultaneously.

When the DISPILS supervisor call is executed, general register 1 must contain the address of a parameter list. The parameter list contains from 3 to 34 bytes, depending on the number of priority interrupt levels to be disabled. Byte 0 contains either a hexadecimal '00' or a hexadecimal 'FF', and byte 1 indicates the number of referenced priority interrupt levels. Each of the remaining bytes contains the hexadecimal equivalent of a decimal number from 0 - 31, indicating a distinct level to be disabled. The format of the parameter list is:

Byte	Contents
0	A hexadecimal '00' or 'FF'. Any other combination is invalid.
1	The number of priority interrupt levels to be disabled by this supervisor call in hexadecimal. Any number greater than 32 or less than 1 is invalid.
2 : 33	One byte, containing the hexadecimal equivalent of 0 - 31, for each priority interrupt level to be disabled. Any number greater than 31 is invalid.

Any invalid conditions, such as disabling a PIL without an RTT attached, will cause the job to be canceled.

2.1.11 ENPILS (SVC 33)--Enable Priority Interrupt Levels

The ENPILS supervisor call is used to enable priority interrupt levels. All of the levels indicated in an individual call are enabled simultaneously. Pending interrupts on the indicated levels will be cleared if the reset option is exercised.

When the ENPILS supervisor call is executed, general register 1 must contain the address of a parameter list. The parameter list contains from 3 to 34 bytes, depending on the number of priority interrupt levels to be enabled. Byte 0 indicates whether the reset option is selected, and byte 1 indicates the number of referenced priority interrupt levels. Each of the remaining bytes contains the hexadecimal equivalent of a decimal number

from 0 - 31, indicating a distinct level to be enabled. The format of the parameter list is:

Byte	Contents
0	An indicator for the reset option. A hexadecimal 'FF' indicates that pending interrupts are to be reset, and a hexadecimal '00' indicates the converse. All other settings are invalid.
1	The number of priority interrupt levels to be enabled with this supervisor call in hexadecimal. Any number greater than 32 or less than 1 is invalid.
2 : 33	One byte, containing the hexadecimal equivalent of 0 - 31, for each priority interrupt level to be enabled. Any number greater than 31 is invalid.

Any invalid conditions, such as enabling a PIL without an RTT attached, will cause the job to be canceled.

2.1.12 RTCANCEL (SVC 47)--Cancel an RTJ

The RTCANCEL supervisor call is used for abnormal termination of an RTJ. If an abort routine is specified, it is given control immediately after execution of RTCANCEL. If the abort routine returns control to the supervisor, a system dump is provided, and both the RTJ and BGJ are canceled. If the abort routine does not return, DAMPS is no longer in control. If the RTCANCEL occurs in an RTT, both the RTJ and BGJ are canceled, and a system dump is provided. If the RTCANCEL occurs in an FGT, the BGJ is not canceled, and a system dump is not provided unless the DUMP option was specified as a job control parameter. In either case, an error message is printed on the console typewriter.

2.1.13 EORTJ (SVC 46)--End of an RTJ

The EORTJ supervisor call is used for normal termination of an RTJ. RTCANCEL is used for abnormal terminations. After EORTJ has been executed, control is returned to the supervisor, which initiates loading the next job. Failure to end an RTJ in this manner may result in interference with later jobs.

The EORTJ supervisor call should be executed in an FGT. If it is executed in an RTT, it is treated as an RTCANCEL executed in an RTT; that is, both the BGJ and RTJ are canceled, a system dump is provided, and a message is printed on the console typewriter.

No parameter list is associated with the EORTJ supervisor call.

2.1.14 RTIO (SVC 43)--Execute Real-Time I/O

The RTIO supervisor call is used to execute input/output operations via the real-time channel scheduler.

When the RTIO supervisor call is executed, general register 1 must contain the address of a request control block (RCB) that defines the desired operation. For a description of the RCB, see 3.2.6.

An invalid or unassigned SYSUNIT index value will result in job cancellation.

2.1.15 MASK/UNMASK (SVC 44/45)--Control I/O Interrupts

The MASK and UNMASK supervisor calls are used to enable and disable I/O interrupts.

When the MASK or UNMASK supervisor call is executed, general register 1 must contain the address of a byte that specifies, in bit positions 0 - 2, the desired setting of the channel mask portion of the system mask. If interruptions from the non-real-time devices are to be accepted during RTT's, the high-resolution interval timer is required.

2.1.16 RDDWORD (SVC 40)--Read Direct Word

The RDDWORD supervisor call is used in conjunction with the Direct Word feature. Execution of this supervisor call causes acceptance of 32 bits of data from the direct-in lines of the Direct Word feature, in conjunction with outputting eight timing pulses and a readout pulse.

General register 1 contains the absolute address of a fullword for the 32 static signals on the direct-in lines. The word must be on a fullword boundary. General register 1 also indicates polarities for the eight timing pulses. The readout pulse is provided automatically by the supervisor call. General register 15 will contain a completion code when control is returned after the supervisor call.

The format of general register 1 is:

Field	Meaning
Bits 0 - 7	A representation of the polarity for the eight timing pulses
Bits 8 - 31	The absolute address of a fullword for receiving the data on the 32 static direct-in lines

The hexadecimal return codes in general register 15 are:

Code	Meaning
'00'	The operation was successfully initiated.
'04'	The input area was not within the bounds of the application program or was not on a fullword boundary. The operation was not initiated.

2.1.17 WTDWORD (SVC 41)--Write Direct Word

The WTDWORD supervisor call is used in conjunction with the Direct Word feature. Execution of this supervisor call provides 32 static signals on the direct-out lines of the Direct Word feature, in conjunction with outputting eight timing pulses and a writeout pulse.

General Register 1 contains the absolute address of a fullword containing the 32 static signals for the direct-out lines. The word must be on a fullword boundary. General register 1 also indicates polarities for the eight timing pulses. The writeout phase is provided automatically by the supervisor. General register 15 will contain a completion code when control is returned after the supervisor call.

The format for general register 1 is:

Field	Contents
Bits 0 - 7	A representation of the polarities for the eight timing pulses
Bits 8 - 31	The absolute address of a fullword containing a representation of the polarities for the 32 static signals for the direct-out line

The hexadecimal return codes in general register 15 are:

Code	Meaning
'00'	The operation was successfully initiated.
'04'	The output area was not within the bounds of the application program or was not on a fullword boundary. The operation was not initiated.

2.1.18 INHIBIT BGJ (SVC 42)

This supervisor call is used to inhibit background job processing. The INHIBIT SVC may be issued only from an FGT. No parameter list is associated with the INHIBIT supervisor call. Following successful execution of INHIBIT, the BGJ will not be given control until the BGJ is released. The BGJ is released when one of the following occurs:

1. The RTJ issues a RELEASE BGJ supervisor call.
2. The RTJ terminates for any reason.

A return code is contained in the low-order byte of register 15:

Code	Meaning
'00'	The operation was successfully completed; the BGJ is inhibited.
'04'	The INHIBIT SVC was not issued from an FGT and therefore was ignored.

2.1.19 RELEASE BGJ (SVC 52)

This SVC is used when the BGJ is to be released after having been inhibited. The RELEASE BGJ supervisor call may be issued only from a foreground task. No parameter list exists for the RELEASE BGJ supervisor call. A return code is provided in the low-order byte of register 15:

Code	Meaning
'00'	The RELEASE BGJ SVC was successfully computed
'04'	The RELEASE BGJ was ignored because the SVC was not issued from an FGT.

2.1.20 QUEUE (SVC 37)--Enter Request for FGT into Foreground Queue 0

The QUEUE supervisor call is used to enter a request for execution of a foreground task, with an associated priority, into foreground queue 0, the highest priority queue. The requests in any foreground queue are honored in priority sequence.

When QUEUE is executed, general register 1 must contain the address of a parameter list. The list contains two fullwords and is aligned on a fullword boundary. Word 0 of the list contains the priority of the request and the absolute address of the entry point of the requested foreground task. Foreground tasks may have multiple entry points. Priorities range from zero to 127, with zero being the highest priority. Word 1 of the list may contain any bit pattern and is placed in general register 1 when the foreground task is given control. Word 1 in the parameter list is for the convenience of the applications programmer and is not used by the system. The format of the parameter list is:

Field	Contents
Word 0	
Bits 0 - 7	A binary number from 0 - 127 representing the priority of the request
Bits 8 - 31	The absolute address of the entry point of the foreground task
Word 1	
Bits 0 - 31	Any bit pattern for the transmittal to foreground task via general register 1

After completion of the SVC QUEUE, control is returned to the applications program with a hexadecimal return code in the low-order byte of general register 15:

Code	Meaning
'00'	The foreground task was successfully placed in foreground queue 0.
'0F'	An invalid priority was contained in bits 0 - 7 of word 0 in the parameter list.
'F0'	The entry point to the foreground task was invalid. The address either was not contained within the RTJ partition or was not on a halfword boundary.
'FF'	The foreground queue was full. DAMPS is distributed with 20 spaces in the queue. If this is insufficient, refer to the DAMPS Operations Manual.
'04'	The address for the parameter list was invalid. The address either was not contained within the RTJ partition or did not indicate a fullword boundary.

If any invalid conditions are indicated, the request is not honored. Only if the return code is '00' was the entry successfully made in the foreground queue.

2.1.21 UNQUEUE (SVC 38) -- Delete Request for FGT from Foreground Queue 0

The UNQUEUE supervisor call is used to delete from foreground queue 0 a previous request for execution of a foreground task. The unqueue request, which is actually placed in foreground queue 0, is honored just before the next foreground scheduling. A queue request is identified on the basis of its entry point and associated priority.

When UNQUEUE is executed, general register 1 must contain the address of a parameter list. The list consists of one fullword, aligned on a fullword boundary, containing the priority and entry point address of the request to be deleted. Just by scheduling the next foreground task, all queue requests in the foreground queue that exactly match both priority and entry point in the unqueue request are deleted. The format of the single word in the parameter list is:

Field	Contents
Bits 0 - 7	A binary number from 0 - 127 representing the priority of the queue request to be deleted
Bits 8 - 31	The absolute address of the entry point of the queue request to be deleted

After completion of the SVC UNQUEUE, control is returned to the application program with hexadecimal return code in the low-order byte of general register 15:

Code	Meaning
'00'	The SVC UNQUEUE request was successfully placed in the foreground queue.
'0F'	An invalid priority was contained in bits 0 - 7 of the word in the parameter list.
'F0'	The address of the entry point to the foreground task was invalid. The address either was not contained within the RTJ partition or was not on a halfword boundary.
'FF'	The foreground queue was full.
'04'	The address for the parameter list was invalid. The address either was not contained within the RTJ partition or did not indicate a fullword boundary.

If any invalid conditions are indicated, the request is not placed in the queue. Only if the return code is '00' was the entry made in the queue.

2.2 FORTRAN SUBROUTINES

This section describes the FORTRAN subroutines that can be used by the FORTRAN programmer to accomplish the special DAMPS functions in an RTJ. Unless otherwise specified, the arguments in a DAMPS FORTRAN subroutine refer to standard-length variables. These calls may not be used in a BGJ.

2.2.1 FIDRTJ -- Identify an RTJ

The FIDRTJ FORTRAN subroutine is used by the FORTRAN programmer to identify a real-time job. Before the execution of this subroutine, standard Model 44 P/S features can be used, but none of the DAMPS special features can be used. Any attempt to use the extended DAMPS features will result in cancellation of the job. The subroutine is executed by the FORTRAN call statement:

```
CALL FIDRTJ
```

No parameter list is associated with the call.

2.2.2 MQUEUE -- Enter Request for FGT into Foreground Queue

The MQUEUE FORTRAN subroutine is used to enter a request for execution of a foreground task, with an associated task priority, into a foreground queue. The requests within each foreground queue are honored in task priority sequence. Multiprogramming is performed among tasks in different foreground queues and the background job.

The MQUEUE FORTRAN subroutine is executed by a FORTRAN call statement with the format:

CALL MQUEUE (IP1, NAME, IRC, IP2, QN)

The names in the parameter list have the following meanings:

IP1 -- An integer from 0 - 127 representing the task priority of the foreground task. All other values are invalid.

NAME -- The name of the foreground task being queued. The name must also appear in an EXTERNAL statement, and the foreground task must have been compiled as a FORTRAN subroutine.

IP2 -- An integer variable that can be used to pass a variable to the queued subroutine

QN -- An integer from 0-3 indicating which of the four queues is to be affected

IRC -- An integer variable that will contain the completion code returned after completion of the subroutine:

IRC Code	Meaning
0	The queue request was successfully placed in the appropriate foreground queue.
4	The address of the parameter list was invalid.
8	The queue specification was invalid-(not 0 - 3).
15	An invalid task priority was contained in IP1.
240	The name of the foreground task was invalid. The name may not have appeared in an EXTERNAL statement.
255	The foreground queue was full.

Only if IRC is zero was the foreground task queued. Any invalid conditions not indicated above will cause the job to be canceled. Also, if more or less than five parameters are contained in the parameter list, the return code is unpredictable.

2.2.3 MDEQUE -- Delete Request for FGT from Foreground Queue

The MDEQUE FORTRAN subroutine is used to delete a previous request for execution of a foreground task from a particular foreground queue. The unqueue request, which is actually placed in the specified foreground queue, is honored just before the next foreground scheduling of that queue. A queue request within any particular queue is identified on the basis of the name of the foreground task and the task priority of the request. When the unqueue request is honored, all requests in the specified queue with the same name and task priority will be deleted.

The MDEQUE FORTRAN subroutine is executed by a FORTRAN call statement with the format:

CALL MDEQUE (IP1, NAME, IRC, QN)

The names in the parameter list have the following meanings:

IP1 -- An integer from 0 - 127 representing the queue priority of the foreground task. All other values are invalid.

NAME -- The name of the foreground task to be deleted from the queue. The name must also appear in an EXTERNAL statement.

QN -- An integer from 0-3 indicating the queue

IRC -- An integer variable containing the completion code returned after completion of the subroutine:

IRC Code	Meaning
0	The dequeue request was successfully placed in the queue.
4	The address of the parameter list was invalid.
8	An invalid queue number was specified--must be in range 0 — 3.
15	An invalid task priority was contained in the parameter list.
240	The name of the foreground task was invalid. The name may not have appeared in an EXTERNAL statement.
255	The specified foreground queue was full.

Only if IRC is zero was the request queued. Any invalid condition other than those outlined above will cause the job to be canceled. If the parameter list contains more or fewer than four entries, the return code is unpredictable.

2.2.4 FDONE -- Terminate an FGT

The FDONE FORTRAN subroutine is used to terminate a FORTRAN foreground task. Control is returned to the supervisor, and the foreground task with the highest priority in the foreground queue is initiated. If no more requests are found in the same queue, the supervisor gives control to a lower level queue or to the background partition. If multiple requests of identical priorities within the same queue are found, the oldest entry is honored first. Use of the FDONE subroutine does not preclude use of the END statement in a FORTRAN subroutine that is to be a foreground task.

The FDONE subroutine is executed by a FORTRAN call statement with the format:

CALL FDONE

The subroutine has no parameter list.

If the FDONE FORTRAN subroutine is used in any routine other than an FGT, the job will be cancelled. If the FDONE subroutine is omitted, subsequent foreground tasks will not be scheduled.

2.2.5 FATACH -- Assign an RTT to a PIL

The FATACH FORTRAN subroutine is used to associate a real-time task with a specific priority interrupt level. Subsequent priority interrupts at that level will cause control to be transferred to the associated real-time task. The real-time task is terminated by the FRETRN FORTRAN subroutine. The FATACH subroutine does not enable the priority interrupt level -- this is the function of the FENPIL FORTRAN subroutine.

The FATACH FORTRAN subroutine is executed by a FORTRAN call statement with the format:

CALL FATACH (RTT, RTSAVE, N, FPSAVE)

The names in the parameter list have the following meanings:

RTT -- The name of the real-time task to be attached. The real-time task must have been compiled as a separate FORTRAN subroutine. The name must appear in a FORTRAN EXTERNAL statement in the subroutine where the attach is executed.

RTSAVE -- The name of an array containing a save area of 61 fullwords on a doubleword boundary. A different save area must be established for each real-time task; this area may not be used for any other purpose.

N -- An integer from 0 - 31 indicating the priority interrupt level to be associated with the real-time task. Zero is the highest priority.

FPSAVE -- The name of an array containing four doublewords on a doubleword boundary. A different floating point save area must be established for each real-time task; this area may not be used for any other purpose.

Any irregularities recognized during the FATACH subroutine will result in cancellation of the job.

2.2.6 FRETRN -- Terminate an RTT

The FRETRN FORTRAN subroutine is used to terminate a FORTRAN real-time task. Control is returned to the next lower real-time task when one is active. If no RTT's are active, control is returned to foreground or background processing. Use of the FRETRN subroutine does not preclude the use of the END statement in a FORTRAN subroutine that is to be a real-time task.

The FRETRN FORTRAN subroutine is executed by a FORTRAN call statement with the format:

CALL FRETRN

The subroutine has no parameter list.

If the FRETRN FORTRAN subroutine is used in any routine other than an RTT, the job will be canceled. If the FRETRN subroutine is omitted, control will never be returned to the point of interruption.

2.2.7 FDSPIL--Disable Priority Interrupt Levels

The FDSPIL FORTRAN subroutine is used to disable priority interrupt levels. All of the levels indicated in an individual call are disabled simultaneously.

The FDSPIL subroutine is executed by a FORTRAN call statement with the format:

CALL FDSPIL (I, J_1, J_2, . . . J_n)

The names of the parameter list have the following meanings:

I -- An integer 0. All other values are invalid.

J_1 - J_n -- Integers ranging from 0 - 31 indicating specific priority interrupt levels to be disabled. From 1 - 32 integers may be specified by a single call.

Any invalid conditions, such as disabling a PIL with no RTT attached, will cause the job to be canceled.

2.2.8 FENPIL--Enable Priority Interrupt Levels

The FENPIL FORTRAN subroutine is used to enable priority interrupt levels. All of the levels indicated in an individual call are enabled simultaneously. Pending interrupts on the indicated levels will be cleared if the reset option is exercised.

The FENPIL FORTRAN subroutine is executed by a FORTRAN call statement with the format:

CALL FENPIL (I, J_1, J_2, . . . , J_n)

The names in the parameter list have the following meanings:

I -- An integer indicating the selection for the reset option. If I = 255, the selected priority interrupts will have pending interrupts reset at the time of enabling. If I = 0, the pending interrupts will not be reset. All other values are invalid.

J_1 - J_n -- Integers ranging from 0 - 31 indicating specific priority interrupt levels to be enabled. From 1 - 32 integers may be specified by a single call.

Any invalid conditions, such as enabling a PIL with no RTT attached, will cause the job to be canceled.

2.2.9 FEORTJ--End of an RTJ

The FEORTJ FORTRAN subroutine is used by the FORTRAN programmer for normal termination of an RTJ. After FEORTJ has been executed, control is returned to the supervisor, and another RTJ may be initiated. CALL FEORTJ should be executed in an FGT. Execution of CALL FEORTJ in an RTT causes the supervisor to cancel both the BGJ and the RTJ and to provide a system dump. The subroutine is executed by the FORTRAN call statement:

CALL FEORTJ

No parameter list is associated with the call.

2.2.10 FCANCL--Cancel an RTJ

The FCANCL FORTRAN subroutine is used by the FORTRAN programmer for abnormal termination of a real-time job. If the DUMP option was indicated in the job card for the run, a storage dump is provided, and the BGJ is canceled. Execution of CALL FCANCL in an RTT causes the supervisor to cancel both the BGJ and the RTJ and to provide a system dump. The subroutine is executed by the FORTRAN call statement:

CALL FCANCL

No parameter list is associated with the call.

2.2.11 FRCBSU--Request Control Block Set Up

The FRCBSU FORTRAN subroutine is used by the FORTRAN programmer to build a request control block. A request control block is required for each request for real-time input or output and serves as the communication link between the program and the channel scheduler. The request control block can be used by only one input/output request at a time. The contents of the block must not be changed during execution of the transaction. If the parameters of the FRCBSU call and subsequent input or output transactions are identical, however, the request control block can be reused.

Several options for input/output transactions may be designated in the request control block. The associated transaction may be queued if the device is busy, or the user may wish to have it canceled. The user may wish to treat the associated request as a burst operation. A request may also be initiated for the queueing of a foreground task upon successful completion of the input/output transaction.

Multiple request control blocks can be constructed with a single FRCBSU call. This option is designed for input/output operations requiring multiple devices to be initiated simultaneously. In particular, it must be used for analog input random operations on the 1827 Data Control Unit (see section 2.2.22 on FAIRAN). In all cases where this option is used, a separate channel program must be provided for each request control block. When this multiple request control block is referenced, DAMPS will treat the requests simultaneously. When executing input/output, DAMPS will execute the operations simultaneously (see section 2.2.12 on FRTIO). When checking completion, DAMPS will check only for completion of all operations (see section 2.2.15 on FCHECK).

The FRCBSU subroutine is executed by a FORTRAN call statement with the format:

```
CALL FRCBSU (RCB, IUNIT1, CCWA1,
IOPT1, IFGTA, NAME, IP1, IP2,
IUNIT2, CCWA2, IOPT2, IUNIT3,
CCWA3, IOPT3)
```

The entries in the parameter list are defined as follows:

RCB -- The name of area where the request control block (RCB) will be constructed and maintained. The area must be eight fullwords for each RCB requested with this call.

IUNIT1 -- An integer code for the symbolic unit. The codes are:

16 = SYS000
17 = SYS001
18 = SYS002
Etc.

(See Appendix E for a complete list.)

CCWA1 -- The name of the area where the channel program will be located. This area will also be referenced in a call to the appropriate subroutine for construction of the channel program.

IOPT1 -- An integer indicating one or more of the following options. If more than one option is being described, the value is the sum of the applicable codes. If none is applicable, this parameter and all the following parameters should be omitted.

IOPT1

Code	Meaning
16	The I/O operation is to be random analog input with comparator.
8	The input/output request is not to be queued by the channel scheduler if the facility is busy.
4	A second request control block is requested, for which additional

parameters are required as described below.

2 The input/output request is to be treated as a burst operation.

1 A foreground task is to be queued in queue 0 when the input/output transaction is completed. Additional parameters are required as described below.

Note: The next four parameters are included only when IOPT is an odd integer, indicating FGT queuing.

IFGTA -- The name of a three-fullword area where a parameter list will be constructed for queuing the foreground task.

NAME -- The name of the entry point of the foreground task.

IP1 -- An integer from 0 - 127 representing the priority of the foreground task. All other values are invalid.

IP2 -- An integer variable that can be used to pass a variable to the queued foreground task.

Note: The next two parameters are required if the multiple request control block option is indicated in IOPT1. The third parameter is optional unless a third request control block is requested.

IUNIT2 -- An integer code for the symbolic unit

CCWA2 -- The name of the area where the channel program will be located. This area will also be referenced in a call to the appropriate subroutine for construction of the channel program.

IOPT2 -- An integer code indicating the selected options. If none of the options is applicable, this parameter should be omitted.

IOPT2 Code	Meaning
4	Additional request control blocks are requested, for which additional parameters are required as described below.
2	The input/output request is to be treated as a burst operation.
6	Both of the above

IUNIT3 -- An integer code for the symbolic unit

CCWA3 -- The name of the area where the channel program will be located. This area will also be referenced in a call to the appropriate subroutine for construction of the channel program.

IOPT3 -- An integer code indicating the burst option. If IOPT3 = 2, the input/output request is to be trested as a burst operation. If the option is not applicable, this parameter should be omitted.

If any irregularities are recognized during execution of the FRCBSU subroutine, the job will be canceled.

2.2.12 FRTIO--Execute Real-Time I/O

The FRTIO subroutine is used by the FORTAN programmer to execute real-time input/output. The FRTIO subroutine is executed by a FORTRAN call statement in the format:

CALL FRTIO (RCB, IRET)

The entries in the parameter list are defined as follows:

RCB -- The name of the request control block that was established by a previous FRCBSU call statement. If multiple request control blocks were established by the FRCBSU call, the system will attempt to initiate all the operations simultaneously. If any busy conditions are encountered, one or more of the operations will be queued.

IRET -- An integer variable where one of the following two codes will be placed by the FRTIO subroutine. This parameter is required only if parameter IOPT1 in the CALL FRCBSU statement indicated that the I/O request was not to be queued.

IRET Code	Meaning
0	The input/output operation was initiated.
8	The input/output operation was not initiated because the unit or channel was busy and the user specified the "no queuing" option when the request control block was established.

In the case of multiple request control blocks, IRET is a collective indicator. For further analysis, it is necessary to examine the individual request control blocks.

If any irregularities are recognized during execution of the FRTIO subroutine, the job will be canceled.

2.2.13 FMSK--Mask I/O Interrupts

The FMSK FORTRAN subroutine is used to mask one or more channels (disable I/O interrupts) during execution of a real-time task.

The FMSK subroutine is executed by a FORTRAN call statement with the format:

CALL FMSK (RCB1, RCB2, RCB3)

The entries in the parameter list reference previously established request control blocks. The channels to which the referenced devices are attached will be disabled. From one to three RCB's may be specified.

If any irregularities are recognized during execution of the FMSK subroutine, the job will be canceled.

2.2.14 FUNMSK--Unmask I/O Interrupts

The FUNMSK FORTRAN subroutine is used to unmask a channel (enable I/O interrupts) during execution of a real-time task.

The FUNMSK subroutine is executed by a FORTRAN call statement with the format:

CALL FUNMSK (RCB1, RCB2, RCB3)

The entries in the parameter list reference previously established request control blocks. The channels to which the indicated devices are attached will be enabled. From one to three RCB's may be specified.

If any irregularities are recognized during execution of the FUNMSK subroutine, the job will be canceled. If interruptions from non-real-time devices are to be accepted during RTT's, the high-resolution interval timer is required.

2.2.15 FCHECK--Check Progress of I/O Operations

The FCHECK subroutine is used by the FORTRAN programmer to check on the progress of a real-time input/output operation. The FCHECK subroutine is executed by a FORTRAN call statement in the format:

CALL FCHECK (RCB, IRET, IOPT)

The entries in the parameter list are defined as follows:

RCB -- The name of the area containing the request control block established with an FRCBSU subroutine call.

IRET -- An integer variable where one of the following return codes will be placed by the FCHECK subroutine:

IRET Code	Meaning
0	The input/output operation came to completion.
1	A unit exception was indicated in the ending status of the input/output operation.
2	An error was indicated in the ending status.
4	The input/output operation has not been completed.
8	The input/output operation was not started. A busy condition was encountered when the operation was requested, and parameter IOPT1 in the FRCBSU call indicated that the request was not to be queued.

If multiple request control blocks were established by the FRCBSU call, the IRET code is a collective reference.

If the user wishes to do error analysis, all the individual request control blocks must be examined to see which devices caused errors.

IOPT -- An integer code of 1 designating that the FCHECK subroutine is to wait for completion of the operation before returning control to the calling program. This parameter is invalid in a call issued from an RTT.

If any irregularities are recognized during execution of the FCHECK subroutine, the job will be canceled.

2.2.16 FTREAD--Read Real-Time Magnetic Tape

The FTREAD subroutine is used by the FORTRAN programmer to build a channel program for the execution of a tape input operation. The channel program, the address of which is maintained in the request control block, is referenced by the real-time channel scheduler when initiating the operation. The actual execution of the input request is accomplished by use of the FRTIO subroutine.

The FTREAD subroutine is executed by a FORTRAN call statement with the format:

CALL FTREAD (CCWA, INA, ICNT, IOPT)

The entries in the parameter list are defined as follows:

CCWA -- The name of an area, two double-words in length, where the channel program is to be constructed

INA -- The name of an area where the input data is to be placed

ICNT -- An integer equal to the number of bytes to be read

IOPT -- The presence of any nonzero integer indicates that records with a length unequal to that specified by ICNT should not be flagged as an error.

Detection of invalid parameters during execution of the FTREAD subroutine will result in job cancellation.

2.2.17 FTWRIT--Write Real-Time Magnetic Tape

The FTWRIT subroutine is used by the FORTRAN programmer to build a channel program for the execution of a tape output operation. The channel program, the address of which is maintained in the request control block, is referenced by the real-time channel scheduler when initiating the operation. The actual execution of the output request is accomplished by the FRTIO subroutine.

The FTWRIT subroutine is executed by a FORTRAN call statement with the format:

CALL FTWRIT (CCWA, OUTA, ICNT)

The entries in the parameter list are defined as follows:

CCWA	--	The name of an area, two doublewords in length, where the channel program is to be constructed
OUTA	--	The name of an area where the output data will be located
ICNT	--	An integer equal to the number of bytes to be written

Detection of invalid parameters during execution of the FTWRIT subroutine will result in job cancellation.

2.2.18 FTWTM--Write End-of-File on Real-Time Magnetic Tape

The FTWTM subroutine is used by the FORTRAN programmer to build a channel program for writing an end-of-file mark on an output tape. The channel program, the address of which is maintained in the request control block, is referenced by the real-time channel scheduler when initiating the operation. The actual execution of the I/O operation is accomplished by the FRTIO subroutine.

The FTWTM subroutine is executed by a FORTRAN call statement with the format:

CALL FTWTM (CCWA)

The CCWA parameter defines an area, two doublewords in length, where the channel program will be constructed. Any irregularities recognized during execution of the FTWTM subroutine will cause the job to be canceled.

2.2.19 FTREW--Rewind Real-Time Magnetic Tape

The FTREW subroutine is used by the FORTRAN programmer to build a channel program for rewinding a tape. The channel program, the address of which is maintained in the request control block, is referenced by the real-time channel scheduler when initiating the operation. The actual execution of the I/O operation is accomplished by the FRTIO subroutine.

The FTREW subroutine is executed by a FORTRAN call statement with the format:

CALL FTREW (CCWA)

The CCWA parameter defines an area, two doublewords in length, where the channel program will be constructed. Any irregularities recognized during execution of the FTREW subroutine will cause the job to be canceled.

2.2.20 FTRUN--Rewind and Unload Real-Time Magnetic Tape

The FTRUN subroutine is used by the FORTRAN programmer to build a channel program for rewinding and unloading a tape. The channel program, the address of which is maintained in the request control block, is referenced by the real-time channel scheduler when initiating the operation. The actual execution of the I/O operation is accomplished by the FRTIO subroutine.

The FTRUN subroutine is executed by a FORTRAN call statement with the format:

CALL FTRUN (CCWA)

The CCWA parameter defines an area, two doublewords in length, where the channel program will be constructed. Any irregularities recognized during execution of the FTRUN subroutine will cause the job to be canceled.

2.2.21 FDIG--Digital Input on 1827 DCU

The FDIG subroutine is used by the FORTRAN programmer to build a channel program for input of digital data from the 1827 DCU. The channel program, the address of which is maintained in the request control block, is referenced by the real-time channel scheduler when initiating the operation. The actual execution of the I/O operation is accomplished by the FRTIO subroutine.

The FDIG subroutine is executed by a FORTRAN call with the format:

CALL FDIG (CCWA, INA, ICNT, IPTADD, IOPT)

The entries in the parameter list are defined as follows:

CCWA	--	The name of the area where the channel program will be constructed. For sequential or single addressing, the area must be two doublewords in length. For a random operation, the length of the area must be 2n doublewords, where n is the number of groups read.
INA	--	The name of the input area. The length of the area must be n halfwords, where n is the number of groups to be read.

ICNT -- An integer equal to the number of digital groups to be read

IPTADD -- The name of the area containing the addresses of the digital input groups to be read. For a sequential operation, the area contains the halfword address of the first group to be read. For a random operation, the area contains n halfword group addresses, where n is equal to the number of groups read.

IOPT -- An integer code indicating one of the following options:

IOPT Code	Meaning
00	Sequential, multiplex mode, no external sync
01	Sequential, forced burst, no external sync
04	Single, multiplex mode, no external sync
05	Single, forced burst, no external sync
08	Sequential, multiplex mode, external sync
09	Sequential, forced burst, external sync
12	Single, multiplex mode, external sync
13	Single, forced burst, external sync
16	Random, multiplex mode, no external sync
17	Random, forced burst, no external sync
24	Random, multiplex mode, external sync
25	Random, forced burst, external sync

If IOPT is omitted, a code of 00 is assumed. If any irregularities occur during execution of the FDIG subroutine, the job will be canceled.

2.2.22 FAIRAN--Random Analog Input on 1827 DCU

The FAIRAN subroutine is used by the FORTRAN programmer to build a channel program for input of random analog data from the 1827 DCU. The channel program, the address of which is maintained in the request control block, is referenced by the real-time channel scheduler when initiating the operation. The actual execution of the I/O operation is accomplished by the FRTIO subroutine. The option for multiple request control blocks must be used in the FRCBSU call. Each request control block references a unique channel program.

The FAIRAN subroutine is executed by a FORTRAN call with the format:

```
CALL FAIRAN (CCWA1, INA1, ICNT1,
IOPT, CCWA2, INA2, ICNT2, CCWA3,
INA3, ICNT3)
```

The entries in the parameter list are defined as follows:

CCWA1 -- The name of a doubleword area where the Analog-to-Digital Converter channel program is to be constructed

INA1 -- The name of the input area. The length of the area must be n halfwords, where n is the number of points to be read.

ICNT1 -- An integer equal to the number of points to be read

IOPT -- An integer code indicating one of the following options:

IOPT Code	Meaning
00	11-bit resolution, no external sync
01	14-bit resolution, no external sync
02	8-bit resolution no external sync
16	11-bit resolution, external sync
17	14-bit resolution, external sync
18	8-bit resolution, external sync

CCWA2 -- The name of a doublword area where the Multiplexor Control channel program is to be constructed.

INA2 -- The name of the area containing the halfword multiplexor addresses of the points to be scanned. If the operation includes the comparator, the area also contains the interspersed limit words. Use 4096 plus the address of the point when addressing the solid-state multiplexors.

Note: If the comparator is to be used during the operation, the next four parameters must be included. If the remaining parameters are omitted, the operation will not involve the comparator, and the number of multiplexor addresses written will be made equal to the number of points read.

ICNT2 -- An integer equal to the number of points to be scanned plus the number of input values to be limit-checked

CCWA3 -- The name of a doubleword area where the comparator channel program is to be constructed

INA3 -- The name of the area where the out-of-limits addresses will be read. For a normal mode of operation, the length of the area should be equal to n halfwords, where n is the number of input values to be limit-checked. For external interrupt mode, the area should be one halfword in length.

INCT3 -- An integer equal to the number of out-of-limits addresses that can be transmitted by a single read command

to the comparator. For a normal mode of operation, the value should be equal to the number of input values to be checked. If the comparator is wired for external interrupt, the value should be 1.

If any irregularities are recognized during execution of the FAIRAN subroutine, the job will be canceled.

2.2.23 FAISEQ--Sequential Analog Input on 1827 DCU

The FAISEQ subroutine is used by the FORTRAN programmer to build a channel program for input of sequential analog data from the 1827 DCU. The channel program, the address of which is maintained in the request control block, is referenced by the real-time channel scheduler when initiating the operation. The actual execution of the I/O operation is accomplished by the FRTIO subroutine.

The FAISEQ subroutine is executed by a FORTRAN call statement with the format:

CALL FAISEQ (CCWA, INA, IPTADD, ICN ', IOPT)

The entries in the parameter list are defined as follows:

CCWA -- The name of an area, two doublewords in length, where the channel program is to be constructed

INA -- The name of the input area. The length of the area must be n halfwords, where n is the number of points to be read.

IPTADD -- The name of a halfword area containing the point address with which the sequential scan is to begin. When addressing the solid state multiplexors, use 4096 plus the address.

ICNT -- An integer equal to the number of points to be read

IOPT -- An integer code indicating one of the following options (if the standard mode is selected, this parameter may be omitted):

IOPT Code	Meaning
00	11-bit resolution, no external sync (standard)
01	14-bit resolution, no external sync
02	8-bit resolution, no external sync
16	11-bit resolution, external sync
17	14-bit resolution, external sync
18	8-bit resolution, external sync

If any irregularities are recognized during execution of the FAISEQ subroutine, the job will be canceled.

2.2.24 FDAO--Digital Analog Input on 1827 DCU

The FDAO subroutine is used by the FORTRAN programmer to build a channel program for digital or analog output to the 1827 DCU. The channel program, the address of which is maintained in the request control block, is referenced by the real-time channel scheduler when initiating the operation. The actual execution of the I/O operation is accomplished by the FRTIO subroutine.

The FDAO subroutine is executed by a FORTRAN call statement with the format:

CALL FDAO (CCWA, OUTA, ICNT, IOPT, CTLA)

The entries in the parameter list are defined as follows:

CCWA -- The name of a doubleword area where the channel program is to be constructed. If the control option is selected, the area must be two doublewords in length on a doubleword boundary.

OUTA -- The name of the output area that contains the output point addresses and data.
For sequential or single address output, the area contains the starting point halfword address followed by the data halfwords.
For random output, the area contains 2n halfwords. Each pair of halfwords contains the point address followed by the data.

ICNT -- An integer equal to the number of points to be written

IOPT -- An integer code indicating one of the following options (if the standard mode is selected, this parameter may be omitted):

IOPT Code	Meaning
00	Sequential, multiplex mode, no external sync (standard)
01	Sequential, forced burst mode, no external sync
02	Random, multiplex mode, no external sync
03	Random, forced burst mode, no external sync
04	Single, multiplex mode, no external sync
05	Single, forced burst mode, no external sync
08	Sequential, external sync, multiplex mode
09	Sequential, external sync, forced burst mode

IOPT Code	Meaning
10	Random, external sync, multiplex mode
11	Random, external sync, forced burst mode
12	Single, external sync, multiplex mode
13	Single, external sync, forced burst mode

To indicate the control option, add 16 to the appropriate option code. If this option is selected, the next parameter is required.

CTLA -- The name of a halfword containing the control data

This option is required only if the control option is selected. The word will contain a 64 for analog buffer and a 128 for pulse output.

2.3 ASSEMBLER LANGUAGE SUBROUTINES

The subroutines described on the following pages were designed for inclusion in FORTRAN programs, but they are available to Assembler Language programmers and can be executed in an RTT or FGT. The routines reside in the relocatable library and are incorporated in the user's program by linkage editor. The following rules govern the use of these subroutines.

1. General registers must contain the following:
   Reg 1 -- Address of the parameter list
   Reg 13 -- Address of an 18-word save area. The same save area may be used by more than one subroutine but only at one priority level.
   Reg 14 -- Return address
   Reg 15 -- Address of the subroutine
2. The parameter list consists of one or more fullword addresses as defined for the subroutine. The first byte of the last entry in the list must contain the hexadecimal code 80.

The following example illustrates the required coding.

```
                EXTRN       FRTIO
                .
                .
                .
                LA          1,PARLIST
                LA          13,SAVE
                L           15,=A(FRTIO)
                BALR        14,15
                .
                .
                .
SAVE            DS          18F
PARLIST         DC          x'80'
                DC          AL3(RCB)
```

2.3.1 FRCBSU--Request Control Block Set Up

The FRCBSU subroutine is used by an Assembler Language programmer to construct a request control block. When the subroutine is called, general register 1 must contain the address of a parameter list defined as follows:

Word 0 -- Address of the area where the RCB's will be constructed. The length of the area must be eight fullwords for each RCB requested with this call.

Word 1 -- Address of a fullword containing the symbolic unit index value

Word 2 -- Address of the channel program

Word 3 -- Address of a fullword containing one or more of the option codes listed below. If more than one option apply, the value should be the sum of the codes. If none of the options applies, this parameter should be omitted.

OP Code	Meaning
16	The I/O operation is to be random analog input with comparator (normal mode).
8	The input/output request is not to be queued by the channel scheduler if the facility is busy.
4	A second request control block is requested, for which additional parameters are required as described in word 7.
2	The input/output request is to be treated as a burst operation.
1	A foreground task is to be queued when the input/output operation is completed. Additional parameters are required as described below.

Note: The next four parameters are required only when a foreground task is to be queued.

Word 4 -- Address of an area, three fullwords in length, where the parameter list for the foreground task will be constructed

Word 5 -- Address of a fullword containing the address of the entry point of the foreground task

Word 6 -- Address of a fullword containing the FGT priority code

Word 7 -- A fullword used to pass variable information to the foreground task that is queued when the input/output operation is completed. The address of this word is passed to the FGT in register 1.

Note: The next two parameters are required if multiple request control blocks are to be constructed. The third is operational unless a third RCB is requested.

Word 8 -- Address of a fullword containing the symbolic unit index value

Word 9 -- Address of the channel program

Word 10 -- Address of a fullword containing one of the following option codes:

OP Code	Meaning
4	A third request control block is requested, for which additional parameters are required as described below.
2	The input/output request is to be treated as a burst operation.
6	Both of the above

Word 11 -- Address of a fullword containing the symbolic unit index value

Word 12 -- Address of the channel program

Word 13 -- Address of a fullword containing the value 2 if the I/O is to be treated as a burst operation; otherwise, this parameter should be omitted.

Any irregularities recognized during execution of this subroutine will result in cancellation of the job.

2.3.2 FRTIO--Execute Real-Time I/O

The FRTIO subroutine is used by an Assembler Language programmer to execute a real-time input/output operation. When the subroutine is called, general register 1 will point to a parameter list defines as follows:

Word 0 -- Address of the request control block that defines the operation

Word 1 -- Address of a fullword where a return code will be posted. This parameter is required only if the option code in the RCB indicates that the request is not to be queued if facilities are busy.

Return Code	Meaning
0	The input/output operation was initiated.
8	The input/output operation was not initiated because the unit or channel was busy and the user specified the "no queuing" option when the request control block was established.

If any irregularities are recognized during execution of the FRTIO subroutine, the job will be canceled.

2.3.3 FCHECK--Check Progress of I/O Operation

The FCHECK subroutine is used by the Assembler Language programmer to check on the progress or results of a real-time input/output operation. When the subroutine is called, general register 1 will point to a parameter list defined as follows:

Word 0 -- Address of the request control block

Word 1 -- Address of a fullword where one of the following return codes will be placed by the FCHECK subroutine:

Return Code	Meaning
0	The operation came to a normal completion.
1	Unit exception was indicated in the ending status .
2	An error was indicated in the ending status.
4	The operation has not been completed.
8	The input/output operation was not started. A busy condition was encountered when the operation was requested, and the option code in the RCB indicated that the request was not to be queued.

Word 2 -- Address of a fullword containing a code that may indicate one or more of the following options. If more than one of the options apply, the value should equal the sum of the codes; if none applies, this parameter should be omitted.

OP Code	Meaning
1	The program is to wait for completion of the operation. This option is invalid when exercised at priority interrupt level.
2	Return to the end-of-file routine specified by word 3 when a unit exeption is indicated in the ending status.
4	Return to the error routine specified by word 4 (or word 3 if no EOF routine is provided) when an error is indicated in the ending status.

Word 3 -- If an EOF routine is provided (indicated by option code 2) word 3 contains the address to which control will be passed. If no EOF routine is provided but an error routine is indicated by option code 4, word 3 contains the address of the error routine, and word 4 is omitted.

Word 4 -- Address to which control will be passed when an error is indicated in the ending status.

Note: When FCHECK passes control to an end-of-file or an error routine, register 15 contains the address of that routine. All other registers are

restored as for a normal return, including register 14, which contains the return address.
If any irregularities are recognized during execution of the FCHECK subroutine, the job will be canceled.

2.3.4 FUNMSK--Unmask I/O Interrupts

The FUNMSK subroutine is used to unmask one or more channels (enable I/O interrupts) during execution of a real-time task. When the subroutine is called, general register 1 must contain the address of a parameter list that includes the addresses of from one to three request control blocks. The channels to which the referenced devices are attached will be enabled, and unreferenced channels will be disabled.

If any errors are detected during execution of the FUNMSK subroutine, the job will be canceled. If interruptions from non-real-time devices are to be accepted during RTT's, the high-resolution interval timer is required.

2.3.5 FMSK--Mask I/O Interrupts

The FMSK subroutine is used to mask one or more channels (disable I/O interrupts) during execution of a real-time task. When the subroutine is called, general register 1 must contain the address of a parameter list that includes the addresses of from one to three request control blocks. The channels to which the referenced devices are attached will be disabled, and unreferenced channels will be enabled.

If any errors are detected during execution of the FMSK subroutine, the job will be canceled.

2.3.6 FTREAD--Read Real-Time Magnetic Tape

The FTREAD subroutine is used by the Assembler Language programmer to construct a channel program to read magnetic tape. When the subroutine is called, general register 1 will contain the address of a parameter list defined as follows:

- Word 0--Address of an area, two doublewords in length, where the channel program will be constructed
- Word 1--Address of the input area
- Word 2--Address of a fullword containing the number of bytes to be read
- Word 3--The presence of word 4 in the parameter list indicates that a wrong-length record will not be flagged as an error. The contents of this word are insignificant.

Any irregularities recognized during execution of this subroutine will cause the job to be canceled.

2.3.7 FTWRIT--Write Real-Time Magnetic Tape

The FTWRIT subroutine is used by the Assembler Language programmer to construct a channel program to write a tape. When the subroutine is called, general register 1 will contain the address of a parameter list defined as follows:

- Word 0--Address of an area, two doublewords in length, where the channel program will be constructed
- Word 1--Address of the output area
- Word 2--Address of a fullword containing the number of bytes to be written

Any irregularities recognized during execution of this subroutine will cause the job to be canceled.

2.3.8 FTWTM--Write End-of-File on Real-Time Magnetic Tape

The FTWTM subroutine is called by the Assembler Language programmer to build a channel program for writing a tape mark. When the subroutine is called, general register 1 will contain the address of a parameter list consisting of the address of an area, two doublewords in length, where the channel program will be constructed. Any irregularities recognized during execution of this subroutine will cause the job to be canceled.

2.3.9 FTREW--Rewind Real-Time Magnetic Tape

The FTREW subroutine is used by the Assembler Language programmer to construct a channel program for rewinding a tape. When the subroutine is called, general register 1 will contain the address of a parameter list consisting of the address of an area, two doublewords in length, where the channel program will be constructed. Any irregularities recognized during execution of this subroutine will cause the job to be canceled.

2.3.10 FTRUN--Rewind and Unload Real-Time Magnetic Tape

The FTRUN subroutine is used by the Assembler Language programmer to construct a channel program for rewinding and unloading a tape. When the subroutine is called, general register 1 will contain the address of a parameter list consisting of the address of an area, two doublewords in length, where the channel program will be constructed. Any irregularities recognized during execution of this subroutine will cause the job to be canceled.

2.3.11 FDIG--Digital Input on 1827 DCU

The FDIG subroutine is used by the Assembler Language programmer to build a channel program for input of digital data from the 1827 DCU. When the subroutine is called, general register 1 will contain the address of a parameter list defined as follows:

Word 0 -- Address of the area where the channel program is to be built. For sequential or single addressing, the area must be two doublewords in length. For a random operation, the length of the area must be 2n doublewords, where n is the number of points to be read.

Word 1 -- Address of the input area. The area must be n halfwords in length, where n is the number of groups read.

Word 2 -- Address of a fullword containing the number of points to be read.

Word 3 -- Address of the area containing the digital input group addresses. For a sequential operation, the area contains the starting address. For a random operation, the area contains the halfword address of each group to be read.

Word 4 -- Address of a fullword containing a code indicating one of the following options (if the standard mode is desired, this parameter may be omitted):

Op Code	Meaning
00	Sequential, multiplex mode, no external sync (standard)
01	Sequential, forced burst, no external sync
04	Single, multiplex mode, no external sync
05	Single, forced burst, no external sync
08	Sequential, multiplex mode, external sync
09	Sequential, forced burst, external sync
12	Single, multiplex mode, external sync
13	Single, forced burst, external sync
16	Random, multiplex mode, no external sync
17	Random, forced burst, no external sync
24	Random, multiplex mode, external sync
25	Random, forced burst, external sync

If any irregularities occur during execution of the FDIG subroutine, the job will be canceled.

2.3.12 FAIRAN--Random Analog Input on 1827 DCU

The FAIRAN subroutine is used by the Assembler Language programmer to build a channel program for input of random analog data from the 1827 DCU. When the subroutine is called, general register 1 will contain the address of a parameter list defined as follows:

Word 0 -- Address of a doubleword area where the ADC channel program is to be built

Word 1 -- Address of the input area. The area must be n halfwords in length, where n is the number of points to be read.

Word 2 -- Address of a fullword containing the number of points to be read.

Word 3 -- Address of a fullword containing a code indicating one of the following options:

Op Code	Meaning
00	11-bit resolution, no external sync
01	14-bit resolution, no external sync
02	8-bit resolution, no external sync
16	11-bit resolution, external sync
17	14-bit resolution, external sync
18	8-bit resolution, external sync

Word 4-- Address of a doubleword area where the multiplexor control channel program is to be built

Word 5-- Address of the area containing the halfword multiplexor addresses of the points to be scanned. If the operation includes the comparator, the area also contains the interspersed limit words.

If the comparator is to be used during the operation, the next four parameters must be included. If the remaining parameters are omitted, the operation will not involve the comparator, and the number of multiplexor addresses written will be made equal to the number of points read.

Word 6-- Address of a fullword containing a value equal to the number of points to be scanned plus the number of input values to be limit-checked

Word 7-- Address of a doubleword area where the comparator channel program is to be built

Word 8-- Address of the area where the out-of-limits addresses will be read. For normal mode of operation, the length of the area should be equal to n halfwords, where n is the number of input values to be limit-checked. For external interrupt mode, the area should be one halfword in length.

Word 9-- Address of a fullword containing the number of out-of-limits addresses that can be transmitted by a single read command to the comparator. For

normal mode of operation, the value should be equal to the number of input values to be checked. If the comparator is wired for external interrupt, the value should be 1.

If any irregularities are recognized during execution of the FAIRAN subroutine, the job will be canceled.

2.3.13 FAISEQ--Sequential Analog Input on 1827 DCU

The FAISEQ subroutine is used by the Assembler Language programmer to build a channel program for input of sequential analog data from the 1827 DCU. When the subroutine is called, general register 1 will contain the address of a parameter list defined as follows:

Word 0 -- Address of the area, two doublewords in length, where the channel program is to be built

Word 1 -- Address of the input area, 2n halfwords in length, where n is the number of points to be read

Word 2 -- Address of the halfword area containing the point address with which the sequential scan is to begin.

Word 3 -- Address of a fullword containing the number of points to be read

Word 4 -- A fullword containing a code indicating one of the following options (if this parameter is omitted, the default option 00 will be used):

Op Code	Meaning
00	11-bit resolution, no external sync
01	14-bit resolution, no external sync
02	8-bit resolution, no external sync
16	11-bit resolution, external sync
17	14-bit resolution, external sync
18	8-bit resolution, external sync

If any irregularities are recognized during execution of the FAISEQ subroutine, the job will be canceled.

2.3.14 FDAO--Digital Analog Input on 1827 DCU

The FDAO subroutine is used by the Assembler Language programmer to construct a channel program for digital or analog output to the 1827 DCU. When the subroutine is called, general register 1 will contain the address of a parameter list defined as follows:

Word 0 -- Address of a doubleword area where the channel program will be built. If the control option is selected, the area must be two doublewords in length.

Word 1 -- Address of the output area containing the point addresses and data
For sequential output, the area contains the starting point address followed by the data half words.
For random output, the area will contain 2n halfwords. Each pair of halfwords contains the point address followed by the data.

Word 2 -- Address of a fullword containing the number of points to be written

Word 3 -- Address of a fullword containing a code indicating one of the following options (if this parameter is omitted, the default option 00 will be used):

Op Code	Meaning
0	Sequential, multiplex mode, no external sync
1	Sequential, forced burst mode, no external sync
2	Random multiplex mode, no external sync
3	Random, forced burst mode, no external sync
4	Single, multiplex mode, no external sync
5	Single, forced burst mode, no external sync
8	Sequential, external sync, multiplex mode
9	Sequential, external sync, forced burst mode
10	Random, external sync, multiplex mode
11	Random, external sync, forced burst mode
12	Single, external sync, multiplex mode
13	Single, external sync, forced burst mode

To indicate the control option, add 16 to the appropriate option code. If this option is selected, the next parameter is required.

Word 4 -- Address of a halfword containing the control data

If any irregularities are recognized during execution of this subroutine, the job will be canceled.

3. PROGRAMMING TECHNIQUES

The following paragraphs outline the recommended techniques for application programming under DAMPS.

3.1 ASSEMBLER LANGUAGE PROGRAMMING

3.1.1 General Register Usage

Each FGT, RTT, and timer interrupt routine must set up its own base registers. When any one of the three receives control, the contents of all registers are unpredictable with one exception--a fullword value may be passed to an FGT in general register 1 when the FGT is given control. However, the system saves all registers and the system status before giving control to each controlled RTT, FGT, or timer interrupt routine; therefore, all registers and DAMPS facilities are available immediately.

The following registers should not be used for base registers:

1--Used for passing parameters to the supervisor call routines

14--Can be used for a base register. However, use of system subroutines requires a return address in register 14.

15--Will contain a return code from supervisor call routines

3.1.2 Real-Time Job

A real-time job (RTJ) can be considered as a normal Model 44 P/S job until the SVC IDENTRTJ instruction is issued. After the SVC IDENTRTJ (identify real-time job) is issued, all DAMPS executed facilities can be used. All real-time tasks (RTT), foreground tasks (FGT), and timer routines together constitute the RTJ. Normal termination of an RTJ is accomplished with the EORTJ SVC issued from an FGT. Abnormal termination may result from a variety of causes, such as a program check, invalid SVC parameters, etc. If an RTJ detects conditions that necessitate abnormal termination, the RTCANCEL SVC can be used to invoke cancellation. Termination can be invoked by an RTT, an FGT, or a timer routine; however, if by other than an FGT, a dump will be taken and the BGJ terminated.

At object time, all routines will have been link-edited together as a real-time job in the phase library to form the RTJ. This RTJ can be initiated from the console as follows:

1. Press request button.
2. Type: RTLOAD
3. Type: // RTJ
4. Type: any SYSUNITS, for example -- //SYS012 ACCESS IN, 280=
5. Type: name of phase, for example-- // EXEC COMP

The routine whose entry point was specified when the RTJ was link-edited becomes the first FGT executing in queue 0.

Normally, the first FGT attaches the RTT's and enables various priority interrupt levels (PIL's). It may also queue additional FGT's. However, until this first FGT has executed an SVC DONE, other FGT's in queue 0 cannot begin processing, since FGT execution within any one of the four queues is serial. However, if this or any subsequent FGT in queue 0 issues an SVC WAIT or CHECK, control is given to any FGT in the next lower queue until the WAIT condition is satisfied.

If no FGT requests are found in queue 1, or if queue 1 issues an SVC WAIT or CHECK, control passes to queue 2. If queue 2 does not require processing, control is passed to queue 3. If queue 3 does not require processing, the BGJ is given control. The BGJ continues until it no longer needs the machine. At that time the system enters the wait state. However, whenever a wait condition is satisfied (either by I/O completion of a WAIT or CHECK, or by an entry being placed into an empty queue) for a queue whose priority is higher than the queue currently processing, the higher level queue is given control immediately. Only if the current queue is executing an SVC will the transfer not be immediate, and in that case it is deferred only until the SVC is completed.

An RTT gains control whenever an interrupt is accepted from the PIL to which it is attached. On this basis, the RTT's are multiprogrammed by priority. Only one RTT can be attached to a given priority interrupt level, and this attachment continues until the RTJ terminates. If the ATTACH SVC is issued a second time for a given PIL, the RTJ will be terminated. Thus a maximum of 32 RTT's can be included in an RTJ.

The distinction between ATTACH and ENABLE is important. Until the PIL has been enabled, an RTT will not receive control even if the priority interrupt is being generated. The ENABLE and DISABLE supervisor call routines can be repeated any number of times. In fact, multiple ENABLE or DISABLE supervisor calls to the same PIL will not create a problem. However, only the PIL's that have RTT's attached can be referenced by the ENABLE and DISABLE SVC routines. The selective enabling or disabling can be performed for one or more specific PIL's without changing the enable/disable status of the other PIL's. The ENABLE and

DISABLE SVC's can be issued from any FGT, RTT, or timer routine. However, if the enabling and disabling of PIL's is handled in RTT's, care should be taken to assure that the correct enable/disable status is maintained. An RTT enabling or disabling a PIL can be interrupted by a higher level PIL before the ENABLE or DISABLE SVC is actually executed. Also, the RTT can return to a lower level PIL, which can also issue an ENABLE/DISABLE SVC and completely negate the effect of the current RTT.

Whenever an RTT disables a lower priority PIL, the RTT attached to the lower PIL is not necessarily prevented from further execution. If the higher level RTT had interrupted while the lower RTT was in a state of execution, when the OLD PSW is loaded, control will return to the RTJ attached to the disabled PIL level.

3.1.3 Foreground Task (FGT)

Foreground tasks execute instructions in problem program state. The entry point to an FGT is defined when the FGT is queued. The FGT is completed when the SVC DONE is executed. If a particular FGT is to be executed more than once, it should be serially reusable and must be queued each time that it is to be executed.

Foreground tasks can be interrupted by any RTT or timer interrupt routine. However, since the status and registers are saved and restored by the system, the interrupt has no effect on the FGT except that time has elapsed. Only one FGT will be processing at any time within one queue. To give control to an FGT, a request must be placed in the foreground queue via the QUEUE SVC. A QUEUE SVC can be executed from an FGT (in any of the four queues), an RTT, or a timer interrupt routine. A task priority, a parameter value (consisting of one fullword), and an indication of which queue is to receive the request is specified when the task is queued. The parameter value is loaded into general register 1 when the FGT is given control. The priority and the parameter value, together with the entry point, are placed in the specified foreground queue.

If the specified foreground queue is empty when an RTT queues an FGT, the priority of the queues is reordered immediately after the system returns to the foreground level.

If that queue is of higher priority level than the queue currently in control, the current queue is suspended, and the new FGT is given control.

Example:

QUEUE 1

FGT1					

QUEUE 2

FGT2	FGT3				

Assume that an RTT interrupts the execution of FGT 2 and places the request for FGT 1 in a previously empty queue. When the RTT completes, control goes to FGT 1--not to FGT 2.

If the queue is not empty, each entry is placed in the queue, and all processing continues unchanged. If a particular queued FGT is no longer desired, it can be unqueued by execution of the SVC UNQUEUE. However, the UNQUEUE SVC will affect all entries in the specified queue that exactly match the entry point and priority level specified. Entries are not deleted directly by the SVC UNQUEUE. The UNQUEUE SVC invokes the QUEUE routine and results in an "unqueue" entry being placed in the specified foreground queue. Thus, when a particular queue is full, an attempt to unqueue an FGT cannot be successfully completed. The size of each of the four foreground queues may be varied only during system creation. The queues need not be the same size.

When an FGT issues an SVC DONE, that entire queue is examined. If an "unqueue" entry is found, all matching entries that were executed before the UNQUEUE are deleted. After that entire queue is processed, the request with the highest priority is given control. If two or more FGT's are found with equal priority, the oldest entry is given precedence. Once an FGT is given control, it continues until it issues an SVC DONE or a request for an I/O. When an FGT issues an I/O instruction, the controls are given to another FGT in another queue; upon completion of the I/O, the controls are returned to the FGT that requested the I/O. When the DONE is executed, the entire queue is again examined to account for any additional FGT's that may have been queued, and control is given to the FGT with the highest priority.

As described in an earlier section, whenever one of the higher levels cannot use the machine (either because it is waiting for I/O completion or because it is empty), the next lower queue will be given control. If one of the queues can use the machine, the BGJ is given control. When the BGJ can no longer use the machine, the system enters the wait state.

Thus, if RTT's, FGT's, or both are able to use all of the available time, the background job never receives control.

3.1.4 Controlled Real-Time Task

The entry point of a controlled real-time task is specified as a parameter to the ATTACH SVC routine, when the RTT is attached to a specific priority interrupt level. Completion of an RTT is accomplished by the RETURN supervisor call, which returns control to the point of interruption. Once the PIL has been enabled, the RTT is given control every time the PIL interrupts. The RTT runs to completion unless it is interrupted by a higher PIL. In this case, the interrupting RTT completes and returns to the point of interruption as though the interrupt had never occurred. A controlled RTT executes instructions in problem state and cannot use the Model 44 P/S supervisor services. All of the DAMPS SVC routines except INHIBIT/BGJ and RELEASE/BGJ may be used by a controlled RTT. However, execution of EORTJ in an RTT will cause both the RTJ and BGJ to be terminated.

Only one RTT should be attached to a particular PIL. Any attempt to attach more than one RTT to the same PIL will result in RTJ termination. A unique 61-word save area must be provided for each RTT that is attached. This save area must not be used by the problem program. Also, each RTT that intends to use floating point registers must provide an additional save area consisting of four doublewords. When priority interrupts are accepted, these areas are used to save the status of the system. An example is shown in Figure 3.

When the SVC RETURN is executed, registers and status are restored from that save area. Since PIL's can interrupt at any time, obviously any attempt to make use of the save areas in an application program would have catastrophic results.

When the controlled RTT is given control, all registers and DAMPS facilities are immediately available to the RTT. The RTT establishes the required base registers and performs the necessary processing. Then, when the SVC RETURN is executed, control returns to the exact point of PIL interruption. Every time a priority interrupt is accepted, the appropriate RTT is given control. Thus, the same RTT will be executed as many times as interrupts are accepted and therefore should be serially reusable. However, RTT's are not required to be reentrant.

```
      LA      1, ATTPIL1            *ATTACH RTTL
      SVC     32
      LA      1, ATTPIL4            *ATTACH RTT4
      SVC     32
      LA      1, ENLIST14           *ENABLE PILS 1 and 4
      SVC     33




ATTPIL1       DS 0F
              DC A(REALTIM1)        *ENTRY POINT TO RTT
              DC A(PSAVE1)          *ADDR OF 61 WORDS PIL SAVE AREA
              DC H'1'               *PIL NUMBER - 1
              DC X'FF00'            *CTL BYTES
              DC F'0'               *NO FLOATING POINT AREA

ATTPIL4       DC A(REALTIM4)
              DC A(PSAVE4)
              DC H'4'               same comments as above
              DC X'FF00'
              DC F'0'




   PSAVE1     DC 61 F'0'  SAVE     AREA     FOR     PIL     1
   PSAVE4     DC 61 F'0'   "        "        "       "      4
```

Figure 3. Example of a controlled RTT

3.1.5 Uncontrolled Real-Time Task

There are two types of uncontrolled RTT's:

1. The totally uncontrolled RTT, which never uses the DAMPS supervisor and runs to completion in supervisor state with a storage protect key of zero. Control is returned to the point of interruption by execution of the LPSX instruction.

2. The uncontrolled but convertible RTT, which receives direct control in supervisor state and converts to a controlled RTT in problem state in order to use the DAMPS supervisor. Since this routine completes as a controlled RTT, it issues the SVC RETURN like a normal controlled RTT to return to the point of interruption.

A parameter of the ATTACH SVC specifies whether the RTT is to be controlled, totally uncontrolled, or uncontrolled but convertible. For both the controlled and the convertible RTT, a 61-word save area must be provided. The save area is not required for the totally uncontrolled RTT. The single advantage of the uncontrolled RTT is that it receives immediate, direct control. However, due to the design of the priority interrupt feature, immediate control is guaranteed only for the highest PIL enabled. Each lower PIL cannot interrupt until every PIL of higher priority is inactive. For example, if an uncontrolled RTT is attached to PIL 3, and an interrupt is received by PIL 3 when a controlled RTT attached to PIL 2 is processing, the interrupt at PIL 3 will not be honored until PIL 2 has completed. Because uncontrolled RTT's receive control directly, the system status and all registers are vulnerable. Before the uncontrolled RTT can change the contents of any register, it must save the contents of that register. Since all register contents are unpredictable, the RTT must use the appropriate bootstrap word in lower core (addressable without a base register) to store (bootstrap) the first register. This register can then be used as the base register to store any other registers that may be changed by the uncontrolled RTT. For the totally uncontrolled RTT, any register can be saved in the low-core location. A convertible RTT, however, must store register 15 and follow a precisely defined series of instructions in order to convert to a controlled PIL. For each PIL, a unique word exists in lower core that can be used to bootstrap the first register (see Appendix B). If other than the specific word is used, the resulting damage to the system is unpredictable. Also, the low-core word must not be used for any other purpose.

Although uncontrolled RTT's execute in the supervisor state with a zero protection key 8, execution of any instructions that alter the status of the system (such as SIO, SVC, SSM, or LPSW) is prohibited. When the uncontrolled RTT is completed, it must restore the system status and all registers to the exact contents at the time of interruption and then execute the LPSX instruction for the appropriate old PIL PSW (see Appendix B). This procedure resets the priority latch and returns control to the point of interruption. However, system status and register contents are the responsibility of the uncontrolled RTT. For this reason, and because the uncontrolled RTT's execute in supervisor state, they are considered an extension of the DAMPS supervisor. If at all possible, controlled RTT's should only be used.

If an uncontrolled RTT is to be converted, the first instruction to receive control must be:

```
(ENTRY POINT)    ST    15, LOWWORD
```

LOWWORD is the absolute address of the appropriate bootstrap word as outlined in Appendix B.

The uncontrolled RTT must save all registers that will be changed before conversion. Registers to be used after conversion need not be saved. The uncontrolled part of the RTT can now be executed.

When the RTT is ready to be converted, it must save registers 13 and 14 (unless it has already saved those registers), load the PIL number (a binary value from 0-31) and BAL 14, ALLOW. ALLOW is a routine located in lower memory that will save the system status normally saved by the DAMPS supervisor. The absolute address of ALLOW is given in Appendix B.

When ALLOW returns control, registers 13 and 14 must be restored to the contents they had when the RTT received control. Any other registers that were changed, with the exception of register 15, must also be restored. The associated LOWWORD must remain unchanged from the start of the RTT. At this point, the contents of registers 0 through 14 should be the same as they were immediately following execution of the first instruction by the controlled RTT. Next, the uncontrolled RTT issues the CONVERT supervisor call, which will convert the RTT from uncontrolled to controlled. The SVC CONVERT routine saves the registers and returns control to the RTT in problem state. At this point, the converted RTT may use any of the system functions that are available to a controlled RTT. The RTT will again be uncontrolled for subsequent entries. The same series of conversion instructions must be executed for each interrupt.

An example of an uncontrolled convertible RTT is shown in Figure 4. Assume that this RTT is attached to PIL 5. Since registers 13 and 14 must always be saved if an RTT is to be converted, these registers should be used as work registers before conversion.

```
SOURCE STATEMENT


CONVERT    EQU    36
RETURN     EQU    35
LOWWORD5   EQU    1320
ALLOW      EQU    1920

* ENTRY POINT TO THE RTT
ENTRYP     ST     15,LOWWORD 5      ** SAVE REG 15
           BALR   15,0              SET UP BASE REGISTER
           USING  *,15
           ST     1,REG1            SAVE REGISTERS 1 AND 2
           ST     2,REG2            WHICH WILL BE USED BY UNCONTROLLED
*                                   PART OF THE RTT
           ST     13,REG13          ** SAVE 13 and 14
           ST     14,REG14          **

************************************
*  PERFORM IMMEDIATE RESPONSE
*  UNCONTROLLED PART OF RTT.
*
*
*  USE ONLY REGISTERS 1,2,13, AND 14
************************************

           L      13,PILNUMBR       **  LOAD PIL NUMBER INTO 13
           BAL    14,ALLOW          **  CALL ALLOW ROUTINE
           L      13,REG13          **  RESTORE REGISTER 13
           L      14,REG14          **                AND 14
           L      1,REG1            RESTORE REGISTERS 1 AND 2
           L      2,REG2
           SVC    CONVERT           **  CONVERT TO CONTROLLED RTT
*
*  LAST INSTRUCTION TO BE EXECUTED IN SUPERVISOR STATE

*  AT THIS POINT ALL REGISTERS BECOME AVAILABLE AND ALL
*  DAMPS SVCS MAY BE USED.  THE RTT HAS BEEN CONVERTED TO
*  A CONTROLLED RTT IN PROBLEM PROGRAM STATE.
*
*  WHEN READY TO RETURN TO THE POINT OF INTERRUPTION, THE
*  FOLLOWING SVC IS EXECUTED.

           SVC    RETURN

PILNUMBR   DC     F'5'              PIL NUMBER 5
REG1       DS     F
REG2       DS     F
REG13      DS     F
REG14      DS     F
*
*  END OF RTT
```

Figure 4. Example of a convertible, uncontrolled RTT

3.1.6 Timer Interrupt Routine

A timer interrupt routine is established in exactly the same manner as a normal Model 44 P/S timer interrupt routine, except that:

1. An address in register 13 is not required.
2. A routine can be established for each of the four queues.

A timer routine entry point can be changed by execution of another STXITC SVC from the same foreground queue. The time interval for a timer routine is set and changed exactly as for the Model 44 P/S. The routine to be affected is determined by which queue issues the SETIME SVC. The timer routine must issue an SVC RTXITC in order to return to point of interruption.

When the timer interval has expired, an external interrupt is generated. The timer interrupt handler saves all registers and system status and gives control to the appropriate timer interrupt routine in problem program state with all channels and external interrupts masked. All registers and most DAMPS SVC routines are immediately available to the timer routine. However, the BGJ timer routine may not issue any SVC's. A timer routine must establish whatever base registers it requires. When a timer routine is completed, it issues the SVC RTXITC, which returns control to the point of interruption. The resulting effect on the interrupt program is as though the interrupt had never occurred. A timer routine cannot use any other Model 44 P/S supervisor services. Thus, a timer interrupt routine cannot use the timer facilities (GETIME, SETIME, and STXITC). One timer routine cannot interrupt an RTT, another timer routine, or any of the supervisor routines.

However, an FGT from any one of the four queues and the BGJ can be interrupted by any user's timer routine. Since the BGJ timer routine is capable of interruption, thus suspending the foreground, caution should be used when an RTT is multiprogrammed with a BGJ that uses a timer routine.

A timer routine may be interrupted by an RTT, but the RTT will return control to the exact point of interruption. Therefore, the interruption will have no effect on the timer routine.

3.2 INPUT/OUTPUT

The DAMPS extended channel scheduler provides a means to schedule and initiate input/output (I/O) when reentrancy or low overhead is required. Reentrancy and lack of overhead, however, impose certain restrictions and burdens on the user that do not exist under other I/O routines. Some of these additional responsibilities are relieved by the use of system subroutines provided for constructing I/O control blocks and channel programs and for executing and checking I/O operations. A call to one of these routines incurs some overhead; however, careful planning will minimize the effect of this overhead, and the editing performed by the subroutine may reduce debugging time. These modules reside in the relocatable library and are described in section 2.

The System/360 Model 44 P/S channel scheduler is not replaced by the DAMPS I/O module. The two routines coexist; therefore, I/O requests issued to the System/360 Model 44 P/S scheduler are valid if they are issued at foreground level. Use of the DAMPS channel scheduler is valid at either foreground or priority level. Devices that are supported by the System/360 Model 44 P/S but that will be accessed through DAMPS may be referenced by OPEN and CLOSE statements.

3.2.1 Requesting I/O

As in the System/360 Model 44 P/S, the request control block (RCB) is used for the information exchange between the problem program and the channel scheduler. Before execution of a request for I/O, the programmer must have established an RCB that defines the operation in terms of device, channel program, and processing options. The channel program referenced in the RCB must also have been constructed before issuing the I/O request. The actual request for I/O (SVC 43) is executed with register 1 containing the address of the RCB. Neither the channel program nor the RCB should be altered until the completion bit has been posted in the RCB by the channel scheduler. At this time, ending status detail is available in the RCB in the form of the channel status word and sense data. Indicators that summarize the ending status are also set in the RCB communication byte.

3.2.2 Real-Time Task Input/Output

When an RTT receives control, all channels are masked, that is, no I/O interrupts can occur. If the programmer wishes to allow I/O interrupts in the RTT, he must unmask the channel or channels with which he is concerned. Bits 0 - 2 of the referenced byte replace system masks bits 0 - 2, and the new mask remains in effect until another mask change is requested or until the RTT releases control.

No capability exists for selectively enabling I/O interrupts at the device or subchannel level; thus, when the channel is enabled, the program is subject to interruption by any device operating on that

channel. Each interruption caused by a real-time device increases the execution time of the RTT by a minimum of 1.2 milliseconds. The processing of priority level interruptions caused by non-real-time devices is deferred until control returns to the foreground level. The processing time required for this type of interruption, however, includes execution of routines that stack and unstack the interrupt data. If interruptions from non-real-time devices are to be accepted during RTT's, the high-resolution interval timer is required.

When an I/O interrupt is taken at a priority level other than the one at which the operation was initiated, or when the same I/O facilities are being accessed at more than one priority level, an RTT may have to give up control without being able to satisfy a completion test on a previously requested operation. The following examples illustrate this point.

Example A. PIL 1 requests an input operation and releases control.

When the channel is enabled at the foreground level, the interrupt from the input operation is accepted by the CPU, and control is given to the I/O interrupt handler. At this point the operation is physically complete, but, before execution of the routine that sets the completion indicator in the RCB, PIL 1 interrupts and tests the RCB for completion of the previous operation. The completion indicator will not be set ON until PIL 1 again releases control and until the channel scheduler resumes processing at the priority level at which the I/O interrupt occurred.

Example B. The digital input unit in the 1827 is being accessed at PIL 0 and PIL 1.

The RTT for PIL 1 requests an input operation, and the channel scheduler has set the device logically busy before actually initiating the operation with the SIO instruction. PIL 0 interrupts and also requests an I/O operation. This second request is placed in the channel queue and is initiated only after control has been returned to the channel scheduler at PIL 1, the first request has been started, and the resulting interrupt has been processed.

3.2.3 Channel Queuing

When a request for I/O is executed, certain tests are made by the supervisor to determine whether the required facilities are available. If the device or channel facilities are busy and the user wishes to allow deferred execution of I/O, the request is placed in a queue and will be executed in turn when the facilities become available. The queues are maintained by channel, within channel by priority level of requesting routine, and within priority level by order of request. When an I/O interruption occurs, the channel scheduler attempts to initiate I/O requests that have been queued for the interrupting channel. The location in the queue at which the initiation attempt begins is the first request for the priority level in control when the I/O interruption occurred. If control is at foreground level, queue examination begins at priority level zero. The examination progresses from high to low priority with level zero following foreground if the starting point was not level zero.

3.2.4 1827 Data Control Unit -- Programming Notes

Reading analog input in the random mode requires concurrent operation of the Analog-to-Digital Converter (ADC) and the Multiplexor Control. If the comparator is used for input limit checking (in the normal mode), it must run concurrently with the ADC and MPX Control. These "multiple operations" are defined by the user through linked RCB's and are initiated by executing one request for I/O (SVC 43 or FRTIO), which references only the first RCB in the series. The required RCB's, which are always constructed in contiguous storage locations, are formatted as follows:

Random analog scan

1st RCB -- Defines the operation for the ADC, option code bit 1 = 1

2nd RCB -- Defines the operation for the MPX Control, option code bit 1 = 0

Random analog scan with comparator (normal mode)

1st RCB -- Defines the operation for the comparator, option code bit 1 = 1. Option code bit 4 must be set to 1 to cause the channel scheduler to force ending status when the ADC has indicated "end". Omission of this bit causes the comparator to remain busy, and the RCB will not be set complete.

2nd RCB -- Defines the operation for the ADC, option code bit 1 = 1

3rd RCB -- Defines the operation for the MPX Control, option code bit 1 = 0

Additional requirements for linked RCB's are:

1. If either the no-input/output-queuing or FGT-queuing option is applicable to the operation, the indication must be in the first RCB.
2. Each of the devices involved in the operation provides its own ending status; therefore, each of the RCB's must be tested for completion before the operation can be considered complete.

3.2.5 Tapes as a Real-Time Device

Programmers planning to use the DAMPS facilities instead of the System/360 Model 44 P/S facilities for magnetic tape operations should remember that the supervisor will not provide for error recovery. If the nature of the application is such that time is available for re-try, the programmer is responsible for providing an error routine that analyzes the CSW to determine whether the error is recoverable. If chained CCW's are used in the recovery procedure (for example, backspace-erase-write), and if the tape control unit is Model 2404/2804--Simultaneous Tape Control, the programmer should be thoroughly familiar with the functional characteristics of this unit and be prepared to handle the situation when a busy condition is encountered during execution of the channel program. An alternative, although more costly in terms of execution time, is to execute the recovery as three unchained CCW's.

Careful consideration should be taken when magnetic tapes are to be used in a real-time environment. The software support of magnetic tapes in the area of error recovery must be considered as an integral part of the device when determining reliability and data integrity for the magnetic tape unit.

The use of DAMPS (nonrecovery) versus the Model 44 P/S (recovery) for tapes will have a direct impact on reliability and data integrity.

Users of magnetic tape units in a real-time (nonrecovery) environment should place more emphasis on the following:

1. Testing and use of tapes that are as nearly perfect as possible
2. More frequent cleaning of the tape transport
3. Extreme care in handling of tapes

The System/360 Model 44 P/S SVC's OPEN and CLOSE can be used if they are issued at foreground level. NOTE and POINT will produce invalid results, since DAMPS does not maintain file position pointers.

3.2.6 Request Control Block

The format of the RCB is:

WORD 0	SYSUNIT	OPTION	COMMUNICATION
WORD 1	ADDRESS OF CHANNEL PROGRAM		
WORD 2	ADDRESS OF FGT PARAM LIST		
WORD 3	CHANNEL STATUS WORD		
WORD 4	CHANNEL STATUS WORD		
WORD 5	SENSE DATA		
WORD 6	CHANNEL QUEUE POINTER		
WORD 7	CHANNEL QUEUE POINTER		

The fields of the RCB are described in the following table:

Word	Byte	Description
0	0	Symbolic unit index (user-defined). Codes are the same as those defined for Model 44 P/S I/O.
0	1	Option code (user-defined): Hex 80 = Request not to be queued if required facilities are busy. Applies to all RCB's that are part of a multiple operation. Must be set in the first RCB. Hex 40 = Multiple operation. Indicates that the RCB located in the eight words following this RCB is to be processed as part of this operation. This option is designed for I/O requiring concurrent operation of two or three devices (for example, random analog input from the 1827 DCU). Hex 20 = Burst mode operation. Indicates that the channel program will force burst mode on a device that does not normally operate in burst mode. Hex 10 = Queue foreground task when this operation is complete. Word 2 of the RCB contains the address of the parameter list required by the system queuing routine. The FGT will not be queued if an inoperative channel or device prevents starting the operation (bit 2 of communication byte = 1, and channel-end in CSW = 0) or if the operation is not started due to a busy condition (indicated by bit 3 of the communication byte). Hex 08 = Indicator for comparator operation in normal mode
0	2	Communication byte (set by channel scheduler): Hex 80 = Operation complete. May be accompanied by other bits indicating abnormal conditions. Hex 40 = Abnormal end. Summary indication that ending status was other than normal. Condition is further defined by the CSW in bytes 12 - 19 of the RCB. Hex 20 = Nonoperational channel or device. Set whenever (1) an error is indicated when SIO instruction is executed, or (2) an error prevented completion of a sense operation. Hex 10 = Operation not initiated. Channel or device was busy and "no queuing" is indicated in the option code.

Word	Byte	Description
0	3	Reserved for system use
1	4-7	Address of channel program (user-defined)
2	8	Return code from FGT queuing (set when operation is completed): 00 = FGT successfully queued 01 = FGT not queued. Device or channel not operational or operation not initiated due to busy condition when "no queuing" is specified in option code. 04 = Invalid parameter list address (FGT not queued) 0F = Invalid FGT priority (FGT not queued) F0 = Invalid FGT entry point (FGT not queued) FF = FGT queue full (FGT not queued)
2	9-11	Address of foreground task parameter list when bit 3 of option code = 1. The parameter list format is: Word 0, byte 0 = FGT, priority Word 0, bytes 1-3 = FGT entry point address Word 1 = address or value that is to be passed to the FGT
3-4	12-19	CSW at completion of I/O operation
5	20-23	Sense data. Present if "unit check" is indicated in CSW, and bit 2 of communication byte is off.
6-7	24-31	Reserved for system use

3.3 FORTRAN

A DAMPS 2 FORTRAN real-time job (RTJ) consists of one main program containing the setup calls and definitions, and several subroutines, each representing a real-time task (RTT) or a foreground task (FGT). The RTT's and FGT's actually are the computational programs. The main program contains mostly the definition. The rules for writing DAMPS RTT's are the same as for writing ordinary FORTRAN programs, with the exception of the conventions outlined in the following paragraphs.

The FORTRAN RTJ begins processing in the main program, and remains a normal FORTRAN program until a CALL FIDRTJ is executed. After the CALL FIDRTJ, all of the DAMPS-executed facilities can be used via the FORTRAN subroutines described in section 2.2. The programmer in the main program attaches the PIL's to certain programs, sets up request control blocks for reading and writing data, and enables the PIL's of those RTT's that must begin execution immediately. The main program is an FGT and must be terminated with a CALL FDONE statement. For each RTT the programmer must establish two save areas and associate the RTT with a priority interrupt level.

The first save area is 61 words in length and is used by the system for storing intermediate results and the contents of the general registers at priority interrupt time. The second save area is a four-doubleword save area. It is used to save the contents of the four floating point registers at priority interrupt time. The DIMENSION statement is used to establish these two save areas.

For example:

```
DOUBLE PRECISION SAVE1D, SAVE1
DIMENSION SAVE1D (4)
DIMENSION SAVE1 (31)
```

The association between the RTT and the PIL is established through the FATACH subroutine. For example:

```
CALL FATACH (PRG1, SAVE1, 4, SAVE1D)
```

Where PRG1 is the name of the RTT, SAVE 1 is the 61-word save area, 4 is the PIL number, and SAVE1D is the four-doubleword save area. The names of subroutines that are passed as arguments in a CALL statement must be defined by an EXTERNAL statement. For example:

```
EXTERNAL PRG1
```

Each FORTRAN RTT must be written as a FORTRAN subroutine and must be terminated by a CALL FRETRN statement. For example:

```
SUBROUTINE PRG1
DIMENSION A(10), B(10), C(10)
EXTERNAL PRINT
_____________
_____________
_____________
_____________
CALL FRETRN
END
```

While CALL FRETRN represents the last executable statement in a FORTRAN RTT, it does not preclude the necessity for an END statement. FORTRAN RTT's may not execute any FORTRAN statements requiring the use of IBCOM. In particular this convention applies to FORTRAN input/output statements but does not apply to the mathematical subroutine library.

In all RTT's reading and writing must be done with DAMPS input/output routines. However, all of the DAMPS extended facilities, including input/output, are available to the FORTRAN programmer via the subroutines outlined in section 2. 2. Channels are normally masked during the execution of an RTT, but this condition can be reversed by appropriate use of the FMSK and FUNMSK subroutines.

Each FORTRAN foreground task (FGT) is written as a FORTRAN subroutine. Each FGT, as well as the main program, must be terminated by a CALL FDONE statement.

The CALL FDONE in the main program must be executed before subsequent FGT's can be executed, but this requirement has no effect on RTT's. It is advisable to do only the setups in the main program and some preliminary computation, if needed. Then the RTT's and FGT's should be programmed for the solution of the problem.

An example of a main program is shown in Figure 5. The use of FDONE does not preclude the necessity for the END statement.

FORTRAN FGT's may use any normally executable FORTRAN statement, including input/output. The execution of an FGT is instigated by the MQUEUE and FQUEUE statements.

```
      DIMENSION ITRCB(8)
      DIMENSION X(6)
      DOUBLE PRECISION SAVE1D, SAVE2D, TPCCW
      DIMENSION SAVE1D(4), SAVE2D(4), TPCCW(2)
      DIMENSION SAVE1(61), SAVE2(61)
      EXTERNAL PRG1, PRG2
      COMMON X, ITRCB, TPCCW
C     TELLS THE SUPERVISOR THAT THE PROGRAM CONTAINS DAMPS
C     STATEMENT.
C
      CALL FIDRTJ
C
C     BUILD AN RCB AND CHANNEL PROGRAM FOR TAPE WRITING
C
      CALL FRCBSU (ITRCB, 28, TPCCW)
      CALL FTWRIT (TPCCW, X, 24)
C     ATTACH PRIORITY INTERRUPTS TO RTTS
      CALL FATACH (PRG1, SAVE1, 16, SAVE1D)
      CALL FATACH (PRG2, SAVE2, 4, SAVE2D)
      CALL FENPIL (255, 16, 4)
      CALL FDONE
      END
```

Figure 5. Example of a main program

3.3.1 MQUEUE

There are four program queuing areas --0, 1, 2, and 3. Queue 0 has the highest priority, 3 the lowest; that is, all programs queued in queue 0 are executed first, then programs in queue 1, and so on. In the same way that the FATACH statement associates an RTT with a PIL, the MQUEUE statement associates an FGT with a queue. Queuing can be done in any RTT or FGT program:

CALL MQUEUE (P, NAME, IRET, PARAM, Q)

P = Priority
NAME = Name of entry point to FGT (that is, program name)
IRET = Return code or completion code
Q = Queue indicator (0, 1, 2, or 3)
PARAM = Parameter value (can be used to pass a variable to the queued subroutine)

Example:

```
SUBROUTINE RTT1
DIMENSION A(10), B(10)
_______________
_______________
_______________
EXTERNAL FGT1
_______________
_______________
_______________
CALL MQUEUE (10, FGT1, I, 0, 0)
IF (I.EQ.255) CALL FRETRN
IF (I.NE.0) CALL FCANCL
_______________
_______________
_______________
CALL FRETRN
END
```

After the CALL MQUEUE, two statements test the completion code. The programmer may branch to any routine, depending on the condition of the return code.

3.3.2 FQUEUE

The FQUEUE is the same as MQUEUE with one exception--it has one queuing area. The queuing indicator need not be passed as a parameter in the statement. The request goes automatically into queue 0.

CALL FQUEUE (5, PRINT, IR, 0)

The print routine is queued with a priority of 5, IR contains the completion code, and zero is a parameter value.

The DAMPS system provides the programmer with several FORTRAN input/output subroutines for use in RTT's as described in section 2.2. Under no conditions may the regular FORTRAN input/output statements be used in RTT's. In general, the user is required to establish a request control block (RCB) and channel program before actually executing the input/output. These can be established once in the main program and then used in any RTT or FGT routine.

Example:

```
DIMENSION NRCBR (8), ARRAYR (2)
DIMENSION RDCCW (2)
DOUBLE PRECISION RDCCW
_______________
_______________
_______________
CALL FRCBSU (NRCBR, 28, RDCCW)
CALL FAISEQ (RDCCW, ARRAYR, NRDPT, 2)
CALL FRTIO (NRCBR)
_______________
_______________
_______________
```

The above setup will read two analog points through 1827 into the Model 44.

NRCBR is an array for the RCB, 28 is the symbolic unit number, and RDCCW is an array for the channel program. ARRAYR is an array where the data will be read in. NRDPT contains the 1827 point addresses, and 2 is the number of points to be read.

3.3.3 FORTRAN Input/Output

The actual scheduling of any input/output operations (that is, reading and writing data) is accomplished by a CALL FRTIO statement.

Actual input/output is carried on simultaneously. An FCHECK subroutine has been provided to check the progress of input/output operations.

Example:

```
 5 CALL FCHECK (NRCBR, IRET)
   IF (IRET.EQ.0) GO TO 10
   IF (IRET.EQ.4) GO TO 20
10 CALL FRTIO (NRCBR)
_______________
_______________
_______________
20
_______________
_______________
_______________
   GO TO 5
```

Before issuing the FRTIO statement, it is advisable to issue the FCHECK and then interrogate the return code about the status of the channel.

In the above example, the first test on the completion code is to see whether the previous operation has been completed normally. If it has, the program branches to statement 10, where a new read is issued. The other test is to see whether the channel is busy. If it is, the program branches to statement 20, where the programmer may wish to execute other instructions, and then returns to the FCHECK statement.

NRCBR is the address of the area where the RCB was constructed. IRET is the return code.

3.3.4 Overlay

The complete overlay capability available to the Model 44 P/S programmer is available to the DAMPS programmer. In an RTJOB, the overlay techniques are identical to those used by the Model 44 P/S programmer, except that he may not violate the rules for using Model 44 P/S facilities in RTT's. Consequently, the FORTRAN calls LINK and LOAD can not be used in an RTT. In an RTJ, the overlay phases can be prepared in real time in the background and fetched into the foreground partition, overlaying phases that are not currently needed. In an RTJ, care must be taken to avoid fetching phases that are not in the phase library or that are undergoing modification by the background job stream. For a general discussion of job preparation and related concepts, see section 4, "System Operation".

4. SYSTEM OPERATION

4.1 CONCEPTS

The two partitions in DAMPS are operated independently. There are two sources for job control input. BGJ's and RTJ's can be initiated and terminated independently. Each partition has a separate job control processor, but they do not run simultaneously because there is only one set of system tables for the entire system. However, the system automatically interlocks; therefore, the user need not concern himself with this duality. DAMPS in all cases gives priority to the foreground partition--while an RTJ is being loaded, the BGJ is held up. In effect, the real-time loading acts as if it were an FGT.

4.2 PREPARING REAL-TIME JOBS

RTJ's and BGJ's are prepared in the background partition using the standard Model 44 P/S processors. Processing of BGJ's is completely compatible with the Model 44 P/S. The procedures differ slightly, however, for the RTJ. Some new parameters have been added to the system to assist in preparing and operating RTJ's. These are outlined in the following paragraphs.

4.2.1 RTLOAD--Operator Command

The RTLOAD is an operator command for loading the real-time job control processor into the foreground partition. After the processor is loaded, the operator may proceed to enter the job control statements associated with the RTJ. The initial phase of the RTJ must be in the phase library, already prepared in a background job.

4.2.2 RTCNCL--Operator Command

The RTCNCL is an operator command used for canceling an RTJ. All activity in the foreground partition ceases, and a dump is provided if it was specified in the RTJ statement.

4.2.3 RTJ--Job Control Statement

The RTJ statement replaces the JOB job control statement when executing jobs in the foreground partition. In all other respects, the foreground and background job control statements are identical.

4.2.4 RTJOB--Job Control Statement

The RTJOB statement replaces the JOB job control statement when executing RTJ's requiring both partitions.

4.2.5 R--Linkage Editor Parameter

The R linkage editor parameter replaces the S parameter in the PHASE statement for RTJ's.

Example:

Link-edit a program for execution in the foreground partition. The program is composed of one phase called REALTIME and two modules, MAIN and SUB A, already in the module library. The required job control statements are:

```
// EXEC LNKEDT (KEEP)
   PHASE REALTIME, R
   INCLUDE MAIN, L
   INCLUDE SUBA, L
   ENTRY
/*
/&
```

The phase is now contained in the phase library ready for loading by the real-time job control processor. Notice that the keep option is required to enter the phase into the phase library.

4.2.6 ROOR--Linkage Editor Parameter

The ROOR linkage editor parameter replaces the ROOT parameter in the PHASE statement for RTJ's.

4.3 OPERATION OF TWO-PARTITION SYSTEM

The normal mode of operation for DAMPS 2 is two-partition operation. In the distributed system, the real-time partition, which resides immediately above the supervisor, is fixed at 36,623 bytes. The background partition, located from the top of the real-time partition to the end of core storage, fills the remainder of storage (48,908 bytes in a 2044 Model G). Appendix D shows a detailed core storage map for the distributed system on a 2044 Model G. The sizes of these partitions can be altered by reassembling and link-editing the system.

However, each partition must be large enough to contain the pertinent job control processor. The procedure for modifying partition size is given in the System Manual.

The background job stream, which enters via SYSRDR, may contain all Model 44 P/S facilities and console operations, with one exception. If the real-time partition is active, the background job stream may not execute a CONDENSE on the system phase library, SDSABS. If this operation is encountered in the background job stream while the real-time partition is active, the CONDENSE operation is flagged with a VOL REQ ERR, and the JOB continues. If the foreground partition is active for a significant part of the day, it is advisable to have a utility job to CONDENSE SDSABS whenever the real-time partition is quiescent.

4.4 OPERATION OF SINGLE-PARTITION SYSTEM

Sometimes a real-time program will be too large to fit within the real-time partition. To handle this problem and provide compatibility with DAMPS 1, provision has been made in DAMPS Version 2 to revert to single-partition operation. To initiate this procedure, both the real-time and background partitions must be cleared out. The real-time job is then called out of the phase library by an appropriate stream of job control cards read through the card reader. Using this mode of operation, all devices and system units are available to the job.

To identify this single-partition mode of operation to the job control processor, a new job control statement has been added:

```
// RTJOB
```

When using this statement, the following restrictions must be observed:

1. Only one job step is allowed--the program must have been previously link-edited and placed into the system phase library.
2. The only valid job control statements are ACCESS, ALLOC, LABEL, and EXEC.

At the completion of this job, the system reverts to two-partition operation and reads the card reader for the next job. This next job may be a normal background job or another RTJOB.

If an RTJOB statement appears in the background job stream after an RTJ is already operating in the foreground partition, the RTJOB will be "flushed", that is, skipped over in the job stream. If an RTLOAD operator command is recognized after an RTJOB is in execution, the operator command will be ignored.

4.5 ALLOCATION OF SYSTEM UNITS

During two-partition operation of the system, symbolic system units can be assigned to only one partition. The assignments for the distributed system are:

1. SYSAB1 through SYS010 are assigned to the background partition, or the BGJ.
2. SYS011 through SYS020 are assigned to the foreground partition, or the RTJ.

During single-partition operation, the symbolic system units are all assigned to the real-time job, or RTJOB.

The exact assignment of symbolic system units can be changed. The only fixed requirements are that SYSAB1 through SYS003 be assigned to the background partition, and that the symbolic system units assigned to either partition be contiguous.

4.6 DEVICE ASSIGNMENT

During two-partition operations, a device can generally be assigned to only one partition. Certain devices are also restricted to the background partition because of the functions they perform for the system. The specific rules are:

1. The system printer is restricted to the background partition.
2. The system card reader is restricted to the background partition.
3. The system residence unit is permanently assigned to the background partition. However, the rule against sharing devices does not apply to the system's use of the permanently assigned system input data sets such as SDSAB1.
4. The 1052 console can be used by both partitions. When using it as an input device, the following precautions must be observed:
   a. The device remains busy during an entire read operation. If an RTJ cannot afford to wait, the device must not be used for input.
   b. The RTJ must precede each read by writing a descriptive message or command without an intervening wait. In this manner, the operator will always know when he is typing a message to the RTJ.
5. All other devices can be assigned to either partition--but not to both.

5. ERROR HANDLING

DAMPS has not extended the specific error recovery features of the Model 44 P/S. In the background partition, errors are handled exactly as in the Model 44 P/S. In the foreground, the real-time user is given the opportunity to process his own errors whenever possible. If the foreground partition must be canceled due to error conditions, the real-time user is given the opportunity to execute his own abort routine. The specific logic that DAMPS uses in response to various error indications is described in the following paragraphs.

The two partitions operate under DAMPS with virtual independence. The RTJ and BGJ are initiated and terminated with virtual independence. However, the BGJ is sometimes sacrificed to protect the integrity and responsiveness of the RTJ. The converse is never true--the foreground partition is always honored as the primary service of DAMPS.

With the following two exceptions, errors occurring in the background partition should cause it to cancel without affecting the foreground partition:

1. Machine checks will affect both partitions in all cases.
2. Storage partition violations on systems without the Store and Fetch Protection special feature may affect both partitions.

In the following circumstances, cancellation of the RTJ will result in subsequent termination of the BGJ:

1. Terminations caused by RTT's or RTJ timer interrupt routines will cause subsequent termination of the BGJ.
2. When an RTJ that has specified the DUMP option is terminated by any abnormal condition where a dump is provided, the BGJ will subsequently be canceled.
3. Whenever an RTJ abort routine returns control to the supervisor, both partitions will be canceled.

In addition to the above system-initiated actions, it is possible for an uncontrolled RTT to bring down both partitions. In all cases where the system affects the cancellation of both partitions, a system dump is provided.

5.1 MACHINE CHECK

DAMPS does not attempt to recover from a machine check. Processing following such an error is limited to execution of the normal Model 44 P/S machine check routine and, following that, termination of all processing. However, the user is given the opportunity of executing his own abort routine after the machine check interrupt. The abort routine is given control in the supervisor state, with priority interrupts and channels masked. Further processing is completely at the discretion of the user.

5.2 PROGRAM INTERRUPTIONS

In the background partition, program interruptions are handled exactly as in the Model 44 P/S, except that the interruption code is extended to include support for the Store and Fetch Protection special feature. The processing of program interruptions in the foreground partition differs as described in the following paragraphs.

When a program interruption occurs, the action taken depends on the type of error (as defined by the interruption code) and the machine state at the time of the interruption. A program interruption with a code from 1 through 5 always results in cancellation of the job. A program interruption with a code from 6 through 15 results in cancellation of the job if it occurs in the supervisor state or during an RTT or a timer interrupt routine. If an interruption in the latter code class occurs during an FGT, the user is given the option of executing a routine. After taking the appropriate action, the user routine exits to the supervisor, and normal processing resumes. If a user routine is not provided, the job is canceled.

In all cases of abnormal cancellation, the user's abort routine will be given control, if an abort routine is specified. The user-provided abort routine is given control immediately before final cancellation, in supervisor state, with all channels and priority interrupts masked. The real-time status of the system is totally disabled; that is, all PIL's have been disabled, and the real-time I/O queues are cleared. When the user does not provide an abort routine, cancellation continues. The user may attempt to continue cancellation after execution of his abort routine by returning control to the supervisor. Whenever a real-time abort routine is executed, the BGJ is canceled if the supervisor has not been destroyed by the abort routine.

It cannot be too highly emphasized that the abort routine is in complete control of a vulnerable system. No part of the system, including the supervisor, should be invoked at this time. Any procedures are completely at the discretion of the user. Any program check occurring in the abort routine will probably cause the system to attempt a second execution of the cancellation procedure; however, the abort routine will not be called the second time.

5.3 INPUT/OUTPUT ERRORS

In the background partition, input/output errors are handled exactly as in the Model 44 P/S, with one exception. The Model 44 P/S does not support the Store and Fetch Protection special feature. When an input/output partition violation is recognized in the background, the following modified Model 44 P/S message is written by the DAMPS supervisor.

FAOFI CUU IO PROT CHK

The processing of input/output errors in the foreground partition differs as described in the following paragraphs.

The treatment of I/O errors resulting from non-real-time operations in the foreground partition is essentially the same as that furnished by the Model 44 P/S: operations resulting in data errors are retried the number of times standard for that device, nonoperational device/channel indications result in job cancellation, and channel/interface control checks are handled by machine-check simulation.

Errors indicated as a result of I/O operations on real-time devices are communicated to the user through the RCB. Although the DAMPS channel scheduler does not initiate cancellation for any type of I/O error, use of the FCHECK subroutine results in cancellation when nonoperational channel/device or channel/interface control checks are indicated. Under any other circumstances, the user is responsible for initiating the appropriate procedure.

I/O error statistics are accumulated by device and are printed by Job Control at the end of each job. Errors on real-time devices are categorized as permanent failures because the system does not attempt to recover from them automatically.

5.4 SYSTEM DIAGNOSTICS

The processing of diagnostic messages concerning the background partition has not been changed. The messages are the same, and they will appear on the same system unit. All foreground diagnostic messages will be printed on the console typewriter. Foreground diagnostic messages resulting from the standard Model 44 P/S have not been changed, except that they will appear on the console typewriter. They are described in detail in the literature outlined in the bibliography. If the diagnostic message is a result of an error detected by the extended DAMPS supervisor, it will have the format:

```
***CANCELED BY DAMPS SUPERVISOR***
***DAMPS ERROR CODE = III LEV NN***
***LAST USER PSW - XXXXXXXXXXXXXXXX
```

The PSW in the last line of the message indicates the user's SVC that caused the error. The PSW is not printed for errors detected by most of the DAMPS FORTRAN subroutines. The III field contains a hexadecimal code that identifies the specific nature of the error. A complete listing of the error codes is given in Appendix A. The NN field contains a representation of the priority level at which the error occurred, for instance, '00' for priority interrupt level 0, and '16' for priority interrupt level 16. Priority interrupt level 32 indicates the foreground level, and priority interrupt level 33 indicates one of the timer routines. The contents of general register 0 have been changed by the DAMPS supervisor, as will be noted if the Dump option has been selected.

The FORTRAN mathematical library subroutines have been adjusted to cancel via the DAMPS extended supervisor. The DAMPS error codes and the FORTRAN execution error codes are equivalent. A list of these equivalences is given in Appendix A.

6. MINIMUM SYSTEM CONFIGURATION

System/360 Model 2044G, one Multiplexor Channel (#5248 or 4598), Priority Interrupt Feature (#5625) High Resolution Interval Timer (#4555), one System Input Device (1442 Model N1, 2501 Model B1 or B2, 2520 Model B1, 2540 Model 1, 2401 Model 1 through 6), one Output Listing Device (1403 Model 2, 3, 7, N1, 1443 Model N1 or any of the above-listed tape units), one Output Punching Device (1442 Model N2, 2520 Model B2 or B3, or any of the above-listed read punches, or any of the above-listed tape units), one Intermediate Storage Area (Second Single Disk Storage Drive (#6415), 2311 Disk Storage Drive, or two of the above-listed tape units).

6.1 NOTES ON CONFIGURATION

1. Any configuration restrictions on the System/360 Model 44 Programming System processors are also restrictions on their use under DAMPS, Version 2.

2. External interrupt is not supported. If the 1827 Comparator is wired for external interrupt, it must use the Priority Interrupt special feature.

3. Sharing channels between real-time and non-real-time devices will degrade the performance of the real-time channel scheduler.

4. Except for console typewriter output, devices cannot be shared by both real-time and background jobs.

6.2 PROGRAMMING SYSTEM

Version 2 is an extension of the System/360 Model 44 Programming System. The extensions were programmed in the Assembler Language of the system. Application programs can be developed using the FORTRAN and Assembler Languages.

APPENDIX A: SPECIAL ERROR CODES

Table 1. DAMPS Supervisor Special Error Codes

Error Code	Meaning
021	A RETURN SVC was issued from an FGT or a timer routine.
031	A DONE SVC was issued from an RTT or a timer routine.
041	An RTX ITC SVC was issued by other than a timer routine.
070	An illegal SVC was issued from an RTJOB (a real-time job loaded from the console).
077	A special DAMPS SVC was issued from a non-real-time job, for example, before execution of an IDENTRTJ SVC.
0A1	The address of the parameter list for an ATTACH SVC was not on a fullword boundary.
0A2	The address of the parameter list for an ATTACH SVC was not in the problem storage area.
0A3	The entry point of the RTT for an ATTACH SVC was not on a halfword boundary.
0A4	The entry point of the RTT for an ATTACH SVC was not in the problem area, or there was an incorrect number of parameters in the parameter list.
0A5	An invalid control byte was contained in an ATTACH SVC parameter list.
0A6	An ATTACH SVC save area was not on a fullword boundary.
0A7	An ATTACH SVC save area was not in the problem storage area.
0A8	An ATTACH SVC floating point save area was not on a doubleword boundary.
0A9	An ATTACH SVC floating point save area was not in the problem storage area.
0AA	An ATTACH SVC priority interrupt level was invalid, that is, outside the 0 - 31 range.
0AB	An attempt was made to attach an RTT to a priority interrupt level that was already attached.
0C1	An invalid SVC code was issued.
0C2	A System/360 Model 44 P/S SVC was issued from an RTT.
0C3	A System/360 Model 44 P/S SVC was issued from a timer routine.
0D1	The address of the parameter list for a DISPILS SVC was invalid (not in user's area).
0D2	The control code contained in byte 0 of a DISPILS SVC parameter list was invalid, that is, neither '00' nor 'FF'.
0D3	The number of priority interrupt levels indicated by a DISPILS SVC was invalid, that is, outside the 1 - 32 range.
0D4	An invalid priority interrupt level was indicated in a DISPILS SVC, that is, outside the 0 - 31 range.
0D5	An attempt was made to disable a priority interrupt level that was not attached.
0E1	The address of the parameter list for an ENPILS SVC was invalid.
0E2	The control code contained in byte 0 of an ENPILS SVC parameter list was invalid, that is, neither '00' nor 'FF'.

Table 1 (cont'd)

Error Code	Meaning
0E3	The number of priority interrupt levels indicated by an ENPILS SVC was invalid, that is, outside the 1-32 range.
0E4	An invalid priority interrupt level was indicated in an ENPILS SVC, that is, outside the 0 - 31 range.
0E5	An attempt was made to enable a priority interrupt level that was not attached.
0F1	The system was unable to find the JSB, probably because of system damage.
0F2	A DAMPS SVC was issued from an uncontrolled RTT in the supervisor state.
0F3	A CONVERT SVC was issued by a routine other than an uncontrolled RTT. The CONVERT SVC can only be used in an uncontrolled RTT after execution of the ALLOW subroutine.
0F4	A DAMPS SVC has been issued by an uncontrolled RTT before execution of the ALLOW subroutine or while in supervisor state.
B0n	An error was detected in the parameter list of an FRCBSU call. *
B1n	An error was detected in the parameter list of an FDAO Call. *
B2n	An error was detected in the parameter list of an FAISEO call. *
B3n	An error was detected in the parameter list of an FAIRAN call. *
B4n	An error was detected in the parameter list of an FDIG call. *
B5n	An error was detected in a real-time tape call. *
B6n	An error was detected in the parameter list of an FRTIO call. *
B6E	The RCB referenced in an FRTIO call indicates that the previous operation is not complete. Register 2 contains the address of the RCB.
B7n	An error was detected in the parameter list of an FCHECK call. *
B7E	The RCB referenced in an FCHECK call indicates a non-operational channel/device, command reject, or any condition identified by bits 42-47 of the CSW. Register 2 contains the address of the RCB.
B8n	An error was detected in the parameter list of an FMSK or FUNMSK call. *
BA1	The channel scheduler detected an invalid or unassigned SYSUNIT index. Register 1 contains the address of the RCB.
BA2	The I/O interrupt stack area has overflowed.

*If n is not 0, it indicates the position of the invalid parameter. If n is 0, the save area addressed by general register 13 is not in the problem program area, or the number of parameters is incorrect.

Table 2. Equivalence Between DAMPS and FORTRAN Error Codes

DAMPS Error Code	Equivalent FORTRAN Error Code	In FORTRAN Subroutine
F01	OA251I	SQRT
F02	OA252I	EXP
F03	OA253I	LOG
F04	OA254I	SIN/COS
F05	OA255I	ATAN
F06	OA252I	TANH
F07	OA261I	DSQRT
F08	OA262I	DEXP
F09	OA263I	DLOG
F0A	OA264I	DSIN/DCOS
F0B	OA265I	DATAN
F0C	OA262I	DTANH
F0D	OA251I	CABS
F0E	OA251I	CSQRT
F0F	OA252I	CEXP
F10	OA254I	CEXP
F11	OA251I	CLOG
F12	OA253I	CLOG
F13	OA255I	CLOG
F14	OA273I	CLOG
F15	OA252I	CSIN/CCOS
F16	OA254I	CSIN/CCOS
F17	OA274I	CSIN/CCOS
F18	OA275I	CSIN/CCOS
F19	OA261I	CDABS
F1A	OA261I	CDSQRT
F1B	OA262I	CDEXP
F1C	OA264I	CDEXP
F1D	OA281I	CDEXP
F1E	OA282I	CDEXP
F1F	OA261I	CDLOG
F20	OA263I	CDLOG
F21	OA265I	CDLOG
F22	OA283I	CDLOG
F23	OA262I	CDSIN/CDCOS
F24	OA264I	CDSIN/CDCOS
F25	OA284I	CDSIN/CDCOS
F26	OA285I	CDSIN/CDCOS
F27	OA252I	EXP
F28	OA256I	EXP
F29	OA251I	ARSIN/ARCOS
F2A	OA257I	ARSIN/ARCOS
F2B	OA258I	TAN/COTAN
F2C	OA259I	TAN/COTAN
F2D	OA251I	DSINH/DCOSH
F2E	OA266I	DSINH/DCOSH
F2F	OA251I	DARSIN/DARCOS
F30	OA267I	DARSIN/DARCOS
F31	OA268I	DTAN/DCOTAN
F32	OA269I	DTAN/DCOTAN
F33	OA251I	EXP
F34	OA252I	GAMMA/ALGAMMA
F35	OA253I	GAMMA/ALGAMMA
F36	OA290I	GAMMA/ALGAMMA
F37	OA291I	GAMMA/ALGAMMA
F38	OA262I	DERF/DERFC/DEXP
F39	OA262I	DGAMMA/DLGAMA
F3A	OA263I	DGAMMA/DLGAMA
F3B	OA300I	DGAMMA/DLGAMA
F3C	OA301I	DGAMMA/DLGAMA
F3D	OA241I	FIXPI#
F3E	OA242I	FRXPI#
F3F	OA243I	FDXPI#
F40	OA252I	FRXPR#
F41	OA253I	FRXPR#
F42	OA244I	FRXPR#
F43	OA245I	FDXPD#
F44	OA263I	FDXPD#
F45	OA262I	FDXPD#
F46	OA246I	FXCPI#
F47	OA247I	FCDXI#

Table 3. Description of Equivalent FORTRAN Error Codes

Error Code	Explanation
OA241I	For an exponentiation operation (i**j) in the subprogram BOAFIXPI (FIXPI#) where i and j represent integer variables or integer constants, the value of j is less than or equal to zero.
OA242I	For an exponentiation operation (r**j) in the subprogram BOAFRXPI (FRXPI#), where r represents a real* 4 variable or integer constant, the value of r is zero, and the value of j is less than or equal to zero.
OA243I	For an exponentiation operation (d**j) in the subprogram BOAFDXPI (FDXPI#), where d represents a real* 8 variable or real* 8 constant and j represents an integer variable or integer constant, the value of d is zero, and the value of j is less than or equal to zero.
OA244I	For an exponentiation operation (r**s) in the subprogram BOAFRXPR (FRXPR#), where r and s represent real*4 variables or real*4 constants, the value of r is zero, and the value of s is less than or equal to zero.
OA245I	For an exponentiation operation (d**p) in the subprogram BOAFDXPD (FDXPD#), where d and p represent real*8 variables or real*8 constants, the value of d is zero, and the value of p is less than or equal to zero.
OA246I	For an exponentiation operation (z**j) in the subprogram BOAFCXPI (FXCPI#), where z represents a complex*8 variable or integer constant, the value of z is zero, and the value of j is less than or equal to zero.

Error Code	Explanation
OA247I	For an exponentiation operation (z**j) in the subprogram BOAFCDXI (FCDXI#), where z represents a complex*16 variable or complex*16 constant and j represents an integer variable or integer constant, the value of j is less than or equal to zero.
OA251I	In the subprogram BOASSQRT (SQRT), the value of the argument is less than zero.
OA252I	In the subprogram BOASEXP (EXP), the value of the argument is greater than 174.673.
OA253I	In the subprogram BOASLOG (ALOG and ALOG10), the value of the argument is less than or equal to zero. Because this subprogram is called by an exponential subprogram, this message also indicates that an attempt has been made to raise a negative base to a real power.
OA254I	In the subprogram BOASSCN (SIN and COS), the absolute value of an argument is greater than or equal to $2^{18}\pi$. ($2^{18}\pi$ = .82354966406249996D+06)
OA255I	In the subprogram BOASATN2, when entry name ATAN2 is used, the value of both arguments is zero.
OA256I	In the subprogram BOASSCNH (SINH or COSH), the value of the argument is greater than or equal to 174.673.
OA257I	In the subprogram BOASASCN (ARCSIN or ARCOS), the absolute value of the argument is greater than 1.
OA258I	In the subprogram BOASTNCT (TAN or COTAN), the absolute value of the argument is greater than or equal to $2^{18}\pi$. ($2^{18}\pi$ = .82354966406249996D+06)

Table 3. Description of Equivalent FORTRAN Error Codes (Cont)

Error Code	Explanation
OA259I	In the subprogram BOASTNCT (TAN or COTAN), the value of the argument is too close to one of the singularities ($\pi/2$, $3\pi/2$, ... for the tangent; π, 2π, ... for the cotangent).
OA261I	In the subprogram BOALSQRT (DSQRT), the value of the argument is less than zero.
OA262I	In the subprogram BOALEXP (DEXP), the value of the argument is greater than 174.673.
OA263I	In the subprogram BOALLOG (DLOG and DLOG10), the value of the argument is less than or equal to zero. Because the subprogram is called by an exponential subprogram, this message also indicates that an attempt has been made to raise a negative base to a real power.
OA264I	In the subprogram BOALSON (DSIN and DCOS), the absolute value of the argument is greater than or equal to $2^{50}\pi$. ($2^{50}\pi$ = .3537118873780223 9D+16)
OA265I	In the subprogram BOALATN2, when entry name DATAN2 is used, the value of both arguments is zero.
OA266I	In the subprogram BOALSCNH (DSINH or DCOSH), the absolute value of the argument is greater than or equal to 174.673.
OA267I	In the subprogram BOALSCN (DARSIN or DARCOS), the absolute value of the argument is greater than 1.
OA268I	In the subprogram BOALTNCT (DTAN or DCOTAN), the absolute value of the argument is greater than or equal to $2^{50}\pi$. ($2^{50}\pi$ = .3537118873702239D+16)
OA269I	In the subprogram IHCLTNCT (DTAN or DCOTAN), the value of the argument is too close to one of the singularities ($\pi/2$, $3\pi/2$, ... for the tangent, π, 2π, ... for the cotangent).
OA271I	In the subprogram BOACSEXP (CEXP), the value of the real part of the argument is greater than 174.673.
OA272I	In the subprogram BOACSEXP (CEXP), the absolute value of the imaginary part of the argument is greater than or equal to $2^{18}\pi$. ($2^{18}\pi$ = .8235496640629996D+06)
OA273I	In the subprogram BOACSLOG (CLOG), the value of both the real and imaginary parts of the argument is zero.
OA274I	In the subprogram BOACSSCN (CSIN or CCOS), the absolute value of the real part of the argument is greater than or equal to $2^{18}\pi$. ($2^{18}\pi$ = .8235496640624999 6D+06)
OA275I	In the subprogram BOACSSCN (CSIN or CCOS), the absolute value of the imaginary part of the argument is greater than 174.673.
OA281I	In the subprogram BOACLEXP (CDEXP), the value of the real part of the argument is greater than 174.673.
OA282I	In the subprogram BOACLEXP (CDEXP), the absolute value of the imaginary part of the argument is greater than or equal to $2^{50}\pi$. ($2^{50}\pi$ = .3537118873780223 9D+16)
OA283I	In the subprogram BOACLLOG (CDLOG), the value of both the real and imaginary parts of the argument is zero.

Table 3. Description of Equivalent FORTRAN Error Codes (Cont)

Error Code	Explanation
OA284I	In the subprogram BOACLSCN (CDSIN or CDCOS), the absolute value of the real part of the argument is greater than or equal to $2^{50}\pi$. ($2^{50}\pi$ = .35371188737802239D+16)
OA285I	In the subprogram BOACLSCN (CDSIN or CDCOS), the absolute value of the imaginary part of the argument is greater than 174.673.
OA290I	In the subprogram BOASGAMA (GAMMA), the value of the argument is outside the valid range. (Valid range = $2^{-252} < x < 57.5744$)
OA291I	In the subprogram BOASGAMA (ALGAMA), the value of the argument is outside the valid range. (Valid range = $0 < x < 4.2937 \times 10^{73}$)
OA300I	In the subprogram BOALGAMA (DGAMA), the value of the argument is outside the valid range. (Valid range = $2^{-252} < x < 57.5744$)
OA301I	In the subprogram BOALGAMA (DLGAMA), the value of the argument is outside the valid range. (Valid range = $0 < x < 4.2937 \times 10^{73}$)

APPENDIX B: BOOTSTRAP WORDS

The following table identifies the low-core bootstrap words that are associated with each PIL and the location of the special PSW for each PIL. The absolute address of the ALLOW subroutine is 1920 (decimal).

PIL	Bootstrap Word	Special Old PSW
0	1280	2048
1	1288	2064
2	1296	2080
3	1304	2096
4	1312	2112
5	1320	2128
6	1328	2144
7	1336	2160
8	1344	2176
9	1352	2192
10	1360	2208
11	1368	2224
12	1376	2240
13	1384	2256
14	1392	2272
15	1400	2288
16	1408	2304
17	1416	2320
18	1424	2336
19	1432	2352
20	1440	2368
21	1448	2384
22	1456	2400
23	1464	2416
24	1472	2432
25	1480	2448
26	1488	2464
27	1496	2480
28	1504	2496
29	1512	2512
30	1520	2528
31	1528	2544

APPENDIX C: DAMPS SVC TABLE

Name of SVC Routine	SVC Code	Can be Issued From							
		BG	FGT	End FGT	RTT	End RTT	Uncontrolled RTT	Timer Routine	End Timer Routine
All Model 44 Ps SVC's	0-24	X	X						
RTXITC	25								X
ATTACH	32		X		X			X	
ENPILS	33		X		X			X	
DISPILS	34		X		X			X	
RETURN	35					X			
CONVERT	36						X		
QUEUE	37		X		X			X	
UNQUEUE	38		X		X			X	
DONE	39			X					
READ DIRECT	40		X		X				
WRITE DIRECT	41		X		X				
INHIBIT BGJ	42		X						
REAL TIME I/O	43		X		X				
MASK I/O CHANNELS	44				X				
UNMASK I/O CHANNELS	45				X				
EORTJ	46		X	X					
RTCANCEL	47		X	X	X	X			
SABRTN	48		X		X				
IDENTRTJ	49	X							
MQUEUE	50		X		X			X	
MDEQUE	51		X		X			X	
RELEASE BGJ	52		X						

APPENDIX D: DAMPS VERSION 2 STORAGE ALLOCATION

Addresses

Hexadecimal	Decimal		Symbolic Name
1FFFF	131,068	Background Partition 48,908 Bytes	D2BHCORE
140F0	82,160		BKGPPORG
140A0	82,080	Bkg. Save Area	BKSAVE
		Bkg. User Comm. Region	
14000	81,920		D2BLCORE
B0F0	45,296	Real-Time Partition 36,623 Bytes	RTJPPORG
B0A0	45,216	Real-Time Save Area	RTSAVE
		RT User Comm. Region	
B000	45,056		D2RLCORE
		Supervisor	

APPENDIX E: FORTRAN DATA SET SYMBOLIC UNIT TABLE

Data Set Reference Numbers	Symbolic Unit Names	
	DEC	
1	17	SYS001
2	18	SYS002
3	19	SYS003
4	20	SYS004
5	6	SYSIPT
6	8	SYSOPT
Back 7	9	SYSPCH
8	16	SYS000
9	21	SYS005
10	22	SYS006
11	23	SYS007
12	24	SYS008
13	25	SYS009
14	26	SYS010
15	27	SYS011
FG. 16	28	SYS012
17	29	SYS013
18	30	SYS014
19	31	SYS015
20	32	SYS016
21	33	SYS017
22	34	SYS018
23	35	SYS019
24	36	SYS020

4. SYSLOG

SAMPLE PROGRAM NO. 1

```
FLGS   L.CTR    OBJCODE   ADDR    STMT      SOURCE STATEMENT

                                   2  * THIS SAMPLE PROGRAM DEMONSTRATES THE RELATIONSHIP BETWEEN REAL TIME
                                   3  * TASKS, FOREGROUND TASKS, AND TIMER INTERRUPT ROUTINES.
                                   4  * WHEN THE PROGRAM IS LOADED, CONTROL IS GIVEN TO THE FIRST
                                   5  * ROUTINE AT BACKGROUND LEVEL.  THIS ROUTINE THEN ISSUES THE SVC
                                   6  * IDENTIFY REAL TIME JOB WHICH CHANGES THE STATUS TO FOREGROUND, AND
                                   7  * THE ROUTINE BECOMES THE FIRST FOREGROUND TASK (TASK1).
                                   8  * TASK1 READS CARDS AND PRINTS THEM.  CONCURRENTLY A RTT ATTACHED TO
                                   9  * PIL 4 IS INCREMENTING A VALUE EVERY TIME PIL 4 INTERRUPTS.  PIL 4
                                  10  * IS SET TO INTERRUPT EVERY 3.3 MILLISECONDS AND PIL 1 EVERY 17
                                  11  * MILLISECONDS.  THE RTT ATTACHED TO PIL 1 OBTAINS THE VALUE GENERATED
                                  12  * BY THE RTT OF PIL 4 AND STORES IT IN A BUFFER.  WHEN 500 VALUES
                                  13  * HAVE BEEN STORED IN THE BUFFER (PIL 1 HAS INTERRUPTED 500 TIMES),
                                  14  * PILS 1 AND 4 ARE DISABLED AND A FGT IS QUEUED WHICH WILL PRINT THE
                                  15  * ENTIRE CONTENTS OF THE BUFFER.  BY DIVIDING 17 MILLISECONDS  BY 3.3
                                  16  * IT BECOMES APPARENT THAT PIL 4 WILL INTERRUPT 5 TIMES FOR EVERY
                                  17  * INTERRUPT FROM PIL 1.  ALSO, SINCE PIL 1 HAS A HIGHER PRIORITY
                                  18  * THAN PIL 4, IT IS POSSIBLE FOR PIL 1 TO INTERRUPT THE RTT ATTACHED
                                  19  * TO PIL 4 BUT PIL 4 CANNOT INTERRUPT THE RTT ATTACHED TO PIL 1.  BOTH
                                  20  * PIL 1 AND PIL 4 INTERRUPT THE FOREGROUND TASK.
                                  21  * TO ILLUSTRATE THE USE OF A TIMER ROUTINE, A TIMER INTERVAL OF 15
                                  22  * SECONDS IS SET WHEN THE FOREGROUND TASK BEGINS LISTING CARDS.
                                  23  * IF CARD LISTING CONTINUES BEYOND 15 SECONDS, A TIMER (EXTERNAL)
                                  24  *
                                  25  * INTERRUPT OCCURS WHICH GIVES CONTROL TO THE TIMER INTERRUPT ROUTINE.
                                  26  * THE TIMER INTERRUPT ROUTINE SETS ON A SWITCH AND THEN RETURNS TO THE
                                  27  * POINT OF INTERRUPTION IN THE FOREGROUND TASK.  THE SWITCH IS TESTED
                                  28  * AS EACH CARD IS LISTED, AND IF IT IS TURNED ON, THE FOREGROUND TASK
                                  29  * TERMINATES.
                                  30  *
                                  31  *
                                  32  *
                                  33  *                          FOREGROUND TASK 1
000000                            34  TASK1     CSECT
000000  05C0                      35            BALR  12,0                ** SET  UP BASE REGISTER FOR FGT1
                        00002     36            USING *,12
                        00031     37  IDENTRTJ EQU   49
                        00020     38  ATTACH   EQU   32
                        00021     39  ENPILS   EQU   33
                        00022     40  DISPILS  EQU   34
                        00032     41  MQUEUE   EQU   50
                        0002E     42  EORTJ    EQU   46
                        00023     43  RETURN   EQU   35
                        00027     44  DONE     EQU   39
                        00019     45  RTXITC   EQU   25
                        00016     46  STXITC   EQU   22
                        00017     47  SETIME   EQU   23
                        00010     48  GETIME   EQU   16
                        00004     49  READ     EQU   4
                        00005     50  WRITE    EQU   5
                        00006     51  CHECK    EQU   6
                        0002F     52  RTCANCEL EQU   47
                        00005     53  SYSRDR   EQU   5
                        00007     54  SYSLST   EQU   7
000002  0A31                      55            SVC   IDENTRTJ            * IDENTIFY REAL TIME JOB
000004  4110 C406       00408     56            LA    1,EJECT             * EJECT PRINTER
```

SAMPLE PROGRAM NO. 1

```
FLGS   L.CTR    OBJCODE   ADDR   STMT      SOURCE STATEMENT

       000008   45E0 C0A8 000AA    57            BAL   14,PRINT1
       00000C   4110 C0CA 000CC    58            LA    1,ATTPIL1          * ATTACH RTT1
       000010   0A20              59            SVC   ATTACH
       000012   4110 C0DA 000DC    60            LA    1,ATTPIL4          * ATTACH RTT4
       000016   0A20              61            SVC   ATTACH
       000018   4110 C0EA 000EC    62            LA    1,ENLIST14         * ENABLE PILS 1 AND 4
       00001C   0A21              63            SVC   ENPILS
                                  64  *
       00001E   5810 C466 00468    65            L     1,=A(TIMRTN)       * SPECIFY TIMER ROUTINE ENTRY POINT
       000022   0A16              66            SVC   STXITC
       000024   59F0 C45E 00460    67            C     15,=F'0'           * TEST RETURN CODE, IF ZERO, ENTRY
       000028   4780 C02C 0002E    68            BE    STEP01               POINT WAS ACCEPTED
       00002C   0A2F              69            SVC   RTCANCEL           * IF NOT CANCEL JOB
       00002E   5810 C3A2 003A4    70  STEP01    L     1,CON15SEC
       000032   0A17              71            SVC   SETIME             * SET TIMER INTERVAL OF 15 SECONDS
       000034   4110 C0EE 000F0    72  MAIN      LA    1,RLIST            * READ CARD INTO INCARD
       000038   0A04              73            SVC   READ
       00003A   0A06              74            SVC   CHECK
       00003C   59F0 C45E 00460    75            C     15,=F'0'
       000040   4780 C044 00046    76            BE    STEP011
       000044   0A2F              77            SVC   RTCANCEL           * IF READ ERROR - CANCEL JOB
                          00046    78  STEP011   EQU   *
       000046   9561 C0FF 00101    79            CLI   INCARD,C'/'
       00004A   4770 C054 00056    80            BNE   STEP012
       00004E   955C C100 00102    81            CLI   INCARD+1,C'*'
       000052   4780 C08C 0008E    82            BE    STEP03
                          00056    83  STEP012   EQU   *
       000056   5810 C3A6 003A8    84            L     1,LINECT           * TEST LINE COUNTER
       00005A   4A10 C46A 0046C    85            AH    1,=H'1'
       00005E   5010 C3A6 003A8    86            ST    1,LINECT
       000062   4910 C46C 0046E    87            CH    1,=H'50'           * HAVE 50 LINES BEEN PRINTED
       000066   4720 C094 00096    88            BH    SKIP
       00006A   4110 C0FE 00100    89  STEP02    LA    1,INCARD-1         * PRINT CARD JUST READ
       00006E   45E0 C0A8 000AA    90            BAL   14,PRINT1
       000072   4110 0001 00001    91            LA    1,1                * TEST TIMESTOP - IF EQUAL TO 1,
       000076   5910 C3AA 003AC    92            C     1,TIMESTOP           15 SECONDS HAS EXPIRED, THEREFORE
       00007A   4770 C032 00034    93            BNE   MAIN                   END THIS FGT
                                  94  *
       00007E   4110 C3AE 003B0    95            LA    1,TIMEXP           * PRINT TIME EXPIRED MESSAGE
       000082   45E0 C0A8 000AA    96            BAL   14,PRINT1
       000086   4110 C406 00408    97            LA    1,EJECT            * EJECT PRINTER
       00008A   45E0 C0A8 000AA    98            BAL   14,PRINT1
                                  99  *                                  * THIS IS THE LINKAGE POINT IF LAST
                                 100  *                                    CARD WAS READ, A TIMER ROUTINE
                          0008E  101  STEP03    EQU   *                      MAY STILL BE PENDING, THEREFORE
       00008E   4110 0000 00000   102            LA    1,0                * CANCEL ANY PENDING TIMER ROUTINE
       000092   0A16             103            SVC   STXITC
                                 104  *
       000094   0A27             105            SVC   DONE               * END OF FGT 1
                                 106  *                 FGT 1 SUBROUTINES
       000096   4110 C406 00408  107  SKIP      LA    1,EJECT
       00009A   45E0 C0A8 000AA  108            BAL   14,PRINT1          * EJECT PRINTER
       00009E   4110 0001 00001  109            LA    1,1
       0000A2   5010 C3A6 003A8  110            ST    1,LINECT           * RESET LINE COUNT
       0000A6   47F0 C068 0006A  111            B     STEP02
```

SAMPLE PROGRAM NO. 1

```
FLGS   L.CTR    OBJCODE    ADDR    STMT      SOURCE STATEMENT

                                   112  *
                                   113  *
       0000AA   5010 C17E  00180   114  PRINT1   ST    1,DATADR
       0000AE   4110 C17A  0017C   115           LA    1,WLIST
       0000B2   0A05               116           SVC   WRITE
       0000B4   0A06               117           SVC   CHECK
       0000B6   59F0 C45E  00460   118           C     15,=F'0'
       0000BA   4780 C0C6  000C8   119           BE    STEP04
       0000BE   59F0 C462  00464   120           C     15,=F'4'
       0000C2   4780 C0C6  000C8   121           BE    STEP04
       0000C6   0A2F               122           SVC   RTCANCEL
       0000C8   07FE               123  STEP04   BR    14
                                   124  *
                                   125  *                        PARAMETER LISTS
                                   126  *
       0000CC                      127  ATTPIL1  DS    OF
       0000CC   00000488           128           DC    A(REALTIM1)          * ENTRY POINT TO RTT
       0000D0   000001B8           129           DC    A(PSAVE1)            * ADDR OF 61 WORD PIL SAVE AREA
       0000D4   0001               130           DC    H'1'                 * PIL NUMBER - 1
       0000D6   FF00               131           DC    X'FF00'              * CTL BYTES
       0000D8   00000000           132           DC    F'0'                 * NO FLOATING POINT SAVE AREA
                                   133  *
       0000DC   00000CD0           134  ATTPIL4  DC    A(REALTIM4)
       0000E0   000002B0           135           DC    A(PSAVE4)
       0000E4   0004               136           DC    H'4'
       0000E6   FF00               137           DC    X'FF00'
       0000E8   00000000           138           DC    F'0'
                                   139  *
       0000EC   FF                 140  ENLIST14 DC    X'FF'                * ENABLE WITH RESET
       0000ED   02                 141           DC    HL1'2'               * LENGTH OF LIST
       0000EE   04                 142           DC    HL1'4'               * PIL 4
       0000EF   01                 143           DC    HL1'1'               * PIL 1
                                   144  *
       0000F0   00000154           145  RLIST    DC    A(RCBC)
       0000F4   00000101           146           DC    A(INCARD)
       0000F8   000000FC           147           DC    A(CON80)
       0000FC   20000050           148  CON80    DC    X'20000050'
       000100   40                 149           DC    C' '
       000101                      150  INCARD   DS    80C
       000154                      151  RCBC     DS    OF
       000154   05                 152           DC    AL1(SYSRDR)
       000155   0000000000000000   153           DC    39X'00'
                                   154  *
       00017C   0000018C           155  WLIST    DC    A(RCBW)
       000180   00000000           156  DATADR   DC    A(0)                 * SET BY PRINT 1 RTN
       000184   00000188           157           DC    A(CON81)
       000188   20000051           158  CON81    DC    X'20000051'
       00018C   07                 159  RCBW     DC    AL1(SYSLST)
       00018D   0000000000000000   160           DC    39X'00'
                                   161  *
                                   162  *                    CONSTANTS, ETC
                                   163  *
       0001B8                      164           DS    OD
       0001B8   0000000000000000   165  PSAVE1   DC    61F'0'               * SAVE AREA FOR PIL 1
       0002B0                      166           DS    OD
```

SAMPLE PROGRAM NO. 1

```
FLGS   L.CTR    OBJCODE    ADDR    STMT      SOURCE STATEMENT

       0002B0   0000000000000000    167  PSAVE4   DC    61F'0'              * SAVE AREA FOR PIL 4
       0003A4   00046500            168  CON15SEC DC    F'288000'           * 15 SECONDS IN 1/19200 SECONDS
       0003A8   00000000            169  LINECT   DC    F'0'                * LINE COUNTER
       0003AC   00000000            170  TIMESTOP DC    F'0'
       0003B0   F0E3C9D4C540D3C9    171  TIMEXP   DC    C'0TIME LIMIT OF 15 SECONDS EXPIRED, CARD LISTING TE'
       0003E2   D9D4C9D5C1E3C5C4    172           DC    C'RMINATED'
       0003EA   4040404040404040    173           DC    30C' '
       000408   F1                  174  EJECT    DC    C'1'
       000409   4040404040404040    175           DC    80C' '
                                    176  *
                                    177           LTORG
       000460   00000000        +   178   DC  F'0'
       000464   00000004        +   179   DC  F'4'
       000468   00000470        +   180   DC  A(TIMRTN)
       00046C   0001            +   181   DC  H'1'
       00046E   0032            +   182   DC  H'50'
                                    183           DROP  12


                                    185  *                        TIMER INTERRUPT ROUTINE
                                +   186   LTORG
       000470                       187  TIMRTN   CSECT
       000470   05C0                188           BALR  12,0                * SET UP BASE REGISTER
                        00472       189           USING *,12
       000472   5810 C00E   00480   190           L     1,=A(TIMESTOP)      * LOCATE TIMESTOP
       000476   4120 0001   00001   191           LA    2,1                 * SET TIMESTOP TO 1
       00047A   5021 0000   00000   192           ST    2,0(1)
       00047E   0A19                193           SVC   RTXITC
                                    194           LTORG
       000480   000003AC        +   195   DC  A(TIMESTOP)
                                    196           DROP  12


                                    198  *                        REAL TIME TASK 1 - PIL 1
                                +   199   LTORG
       000488                       200  REALTIM1 CSECT
       000488   05B0                201           BALR  11,0                * SET UP BASE REGISTER
                        0048A       202           USING *,11
       00048A   5810 B83E   00CC8   203           L     1,=A(INPUT)         * LOCATE INPUT WORD
       00048E   5821 0000   00000   204           L     2,0(1)              * MOVE INPUT WORD TO OUTPUT BUFFER
       000492   5830 B04A   004D4   205           L     3,ADRBUF
       000496   5023 0000   00000   206           ST    2,0(3)
       00049A   4133 0004   00004   207           LA    3,4(3)              R INCREMENT BUFFER ADDRESS
       00049E   5030 B04A   004D4   208           ST    3,ADRBUF
       0004A2   5810 B04E   004D8   209           L     1,COUNT             * TEST FOR 500 INTERRUPTS
       0004A6   5910 B842   00CCC   210           C     1,=F'500'
       0004AA   4780 B02E   004B8   211           BE    QUIT
       0004AE   5A10 B83A   00CC4   212           A     1,=F'1'
       0004B2   5010 B04E   004D8   213           ST    1,COUNT
       0004B6   0A23                214  ENDRTT1  SVC   RETURN              * END OF RTT1
       0004B8   4110 B052   004DC   215  QUIT     LA    1,QLIST             * QUEUE FGT2 AFTER 500 INTERRUPTS
       0004BC   0A32                216           SVC   MQUEUE
       0004BE   59F0 B836   00CC0   217           C     15,=F'0'            * TEST FOR SUCCESSFUL COMPLETION
```

SAMPLE PROGRAM NO. 1

```
L.CTR    OBJCODE   ADDR   STMT      SOURCE STATEMENT

0004C2  4780 B03E  004C8   218            BE    QOK                    IF NOT -
0004C6  0A2F               219            SVC   RTCANCEL                 CANCEL JOB
0004C8  4110 B82E  00CB8   220  QOK       LA    1,DLIST
0004CC  0A22               221            SVC   DISPILS            * DISABLE PIL 1 AND PIL 4
0004CE  47F0 B02C  004B6   222            B     ENDRTT1
                           223  *
                           224  *                        CONSTANTS
                           225  *
0004D2  0000
0004D4  000004E8           226  ADRBUF    DC    A(BUFFER)          * ADDRESS/INDEX OF BUFFER
0004D8  00000001           227  COUNT     DC    F'1'               * NUMBER OF TIMES EXECUTED
0004DC  01                 228  QLIST     DC    X'01'              * PRIORITY OF FGT
0004DD  000CF0             229            DC    AL3(TASK2)         * ENTRY POINT
0004E0  000004E8           230            DC    A(BUFFER)          * PARAMETER VALUE
0004E4  00000000           231            DC    F'0'               *QUEUE PRIORITY NUMBER
0004E8  0000000000000000   232  BUFFER    DC    500F'0'
000CB8  FF                 233  DLIST     DC    X'FF'              * CTL BYTE
000CB9  02                 234            DC    HL1'2'             LENGTH OF LIST
000CBA  01                 235            DC    HL1'1'             * PIL 1 AND
000CBB  04                 236            DC    HL1'4'                PIL 4 TO BE DISABLED
                           237            DROP  11
                           238            LTORG
000CC0  00000000       +   239   DC  F'0'
000CC4  00000001       +   240   DC  F'1'
000CC8  00000CE0       +   241   DC  A(INPUT)
000CCC  000001F4       +   242   DC  F'500'


                           244  *                        REAL TIME TASK 2 - PIL 4
                       +   245   LTORG
000CD0                     246  REALTIM4 CSECT
000CD0  05C0               247            BALR  12,0
                   00CD2   248            USING *,12
000CD2  5810 C00E  00CE0   249            L     1,INPUT            * INCREASE 'INPUT' BY 5 EACH TIME
000CD6  4A10 C016  00CE8   250            AH    1,=H'5'               PIL 4 INTERRUPTS
000CDA  5010 C00E  00CE0   251            ST    1,INPUT
000CDE  0A23               252            SVC   RETURN             * END OF RTT 4
000CE0  00000000           253  INPUT     DC    F'0'
                           254            LTORG
000CE8  0005           +   255   DC  H'5'
                           256            DROP  12


                           258  *                        FOREGROUND TASK 2
                       +   259   LTORG
000CF0                     260  TASK2     CSECT
000CF0  0590               261            BALR  9,0                * SET UP BASE REGISTER
                   00CF2   262            USING *,9
                           263  *                                 * REGISTER 1 CONTAINS THE ADDRESS
000CF2  1871               264            LR    7,1                   OF BUFFER
000CF4  5827 0000  00000   265  LINK01    L     2,0(7)             * LOAD FIRST/NEXT WORD FROM BUFFER
000CF8  4150 0000  00000   266            LA    5,0
000CFC  4130 90CE  00DC0   267            LA    3,OUTWORD
```

SAMPLE PROGRAM NO. 1

```
FLGS   L.CTR    OBJCODE    ADDR   STMT      SOURCE STATEMENT

       000D00   5840 9086   00D78   268            L      4,CONVERT
       000D04   1924                269  LINK02    CR     2,4
       000D06   4740 9022   00D14   270            BL     LINK03            * CONVERT TO DECIMAL
       000D0A   1B24                271            SR     2,4
       000D0C   4A50 9106   00DF8   272            AH     5,=H'1'
       000D10   47F0 9012   00D04   273            B      LINK02
       000D14   4253 0000   00000   274  LINK03    STC    5,0(3)
       000D18   4150 0000   00000   275            LA     5,0
       000D1C   96F0 3000   00000   276            OI     0(3),X'F0'
       000D20   4133 0001   00001   277            LA     3,1(3)
       000D24   1854                278            LR     5,4
       000D26   4140 0000   00000   279            LA     4,0
       000D2A   5D40 90FE   00DF0   280            D      4,=F'10'
       000D2E   1845                281            LR     4,5
       000D30   4150 0000   00000   282            LA     5,0
       000D34   5940 90FA   00DEC   283            C      4,=F'9'
       000D38   4720 9012   00D04   284            BH     LINK02
       000D3C   4223 0000   00000   285            STC    2,0(3)
       000D40   96F0 3000   00000   286            OI     0(3),X'F0'
       000D44   4110 908E   00D80   287            LA     1,LPARM
       000D48   0A05                288            SVC    WRITE
       000D4A   0A06                289            SVC    CHECK
       000D4C   59F0 90F2   00DE4   290            C      15,=F'0'
       000D50   4780 906C   00D5E   291            BE     ONCE
       000D54   59F0 90F6   00DE8   292            C      15,=F'4'
       000D58   4780 906C   00D5E   293            BE     ONCE
       000D5C   0A2F                294            SVC    RTCANCEL
                            00D5E   295  ONCE      EQU    *
       000D5E   4177 0004   00004   296            LA     7,4(7)            * INCRX TO NEXT WORD IN BUFFER
       000D62   5810 908A   00D7C   297            L      1,TOTAL
       000D66   5A10 90EE   00DE0   298            A      1,=F'1'
       000D6A   5010 908A   00D7C   299            ST     1,TOTAL
       000D6E   5910 9102   00DF4   300            C      1,=F'500'         * TEST FOR 500 WORDS PRINTED
       000D72   4740 9002   00CF4   301            BL     LINK01
                                    302  *
       000D76   0A2E                303            SVC    EORTJ             * END OF REAL TIME JOB
                                    304  *
                                    305  *
       000D78   3B9ACA00            306  CONVERT DC       F'1000000000'
       000D7C   00000000            307  TOTAL    DC      F'0'
       000D80   00000D90            308  LPARM    DC      A(RCBL)
       000D84   00000DB8            309           DC      A(AREA)
       000D88   00000D8C            310           DC      A(CON30)
       000D8C   2000001E            311  CON30    DC      X'2000001E'
       000D90   07                  312  RCBL     DC      AL1(SYSLST)
       000D91   0000000000000000    313           DC      39X'00'
       000DB8   40E5C1D3E4C57E40    314  AREA     DC      C' VALUE= '
       000DC0   4040404040404040    315  OUTWORD  DC      30C' '
                                    316           LTORG
       000DE0   00000001         +  317   DC  F'1'
       000DE4   00000000         +  318   DC  F'0'
       000DE8   00000004         +  319   DC  F'4'
       000DEC   00000009         +  320   DC  F'9'
       000DF0   0000000A         +  321   DC  F'10'
       000DF4   000001F4         +  322   DC  F'500'
       000DF8   0001             +  323   DC  H'1'
                            00000   324           END    TASK1
```

SAMPLE PROGRAM NO. 2

```
L.CTR    OBJCODE   ADDR    STMT      SOURCE STATEMENT

                            2  * SAMPLE PROGRAM 2 ILLUSTRATES THE USE OF THE DAMPS CHANNEL SCHEDULER.
                            3  * ALTHOUGH WRITTEN TO BE EXECUTED AT FOREGROUND LEVEL, ALL CODING
                            4  * WOULD BE VALID IN AN RTT EXCEPT THE OPEN AND CLOSE STATEMENTS AND THE
                            5  * I/O WAIT LOOPS.  THE PROBLEM IS TO TAKE 50 READINGS OF FOUR ANALOG
                            6  * INPUT POINTS (RANDOM MODE) AND RECORD ON MAGNETIC TAPE THE INPUT
                            7  * VALUES AND THE ENDING STATUS FROM EACH READ OPERATION.  PROGRAMMING
                            8  * REQUIRED FOR ANALYSIS AND PROCESSING OF I/O ERRORS IS OMITTED FROM
                            9  * THE EXAMPLE.
000000                     10          START 0
000000  0520               11  BEGIN   BALR  2,0
                   00002   12          USING *,2
000002  0A31               13          SVC   49                  IDENTIFY RTJ
000004  4110 216E  00170   14          LA    1,OPEN
000008  0A02               15          SVC   2                   OPEN TAPE
00000A  4150 0032  00032   16          LA    5,50                OUTPUT BLOCK COUNTER
00000E  4140 000A  0000A   17  INIT    LA    4,10                RECORDS/BLOCK
000012  9180 20C8  000CA   18  TESTINPT TM    ACOMM,X'80'         TEST ADC COMPLETE
000016  4780 2010  00012   19          BZ    *-4
00001A  9180 20E8  000EA   20          TM    MCOMM,X'80'         TEST MPX CTRL COMPLETE
00001E  4780 2018  0001A   21          BZ    *-4
000022  9120 20C8  000CA   22          TM    ACOMM,X'20'         TEST NON-OPERATIONAL ADC
000026  4710 20C4  000C6   23          BO    CANCEL
00002A  9120 20E8  000EA   24          TM    MCOMM,X'20'         TEST NON-OPERATIONAL MPX
00002E  4710 20C4  000C6   25          BO    CANCEL
000032  4700 203C  0003E   26          BC    0,MOVINPT           FIRST TIME NOP
000036  92F0 2031  00033   27          MVI   *-3,X'F0'
00003A  47F0 2072  00074   28          B     READ
00003E  5830 216A  0016C   29  MOVINPT L     3,BUFFNDX           MOVE INPUT DATA AND STATUS
                           30  *                                 TO BUFFER
000042  5800 2116  00118   31          L     0,INPUT             STORE INPUT DATA
000046  5003 0000  00000   32          ST    0,0(3)
00004A  5800 211A  0011C   33          L     0,INPUT+4
00004E  5003 0004  00004   34          ST    0,4(3)
000052  4800 20C8  000CA   35          LH    0,ACOMM
000056  4003 0008  00008   36          STH   0,8(3)              STORE ADC STATUS SUMMARY
00005A  4800 20D6  000D8   37          LH    0,ASTATUS
00005E  4003 000A  0000A   38          STH   0,10(3)             STORE ADC CSW STATUS
000062  5800 20DA  000DC   39          L     0,ASENSE
000066  5003 000C  0000C   40          ST    0,12(3)             STORE ACC SENSE DATA
00006A  4133 0010  00010   41          LA    3,16(3)             INCR BUFFER AREA POINTER
00006E  5030 216A  0016C   42          ST    3,BUFFNDX
000072  0640               43          BCTR  4,0                 DECR RCDS/BLOCK COUNTER
000074  4110 20C6  000C8   44  READ    LA    1,AIRCB             ADDRESS INPUT RCB
000078  0A2B               45          SVC   43                  INITIATE NEXT READ
00007A  1244               46          LTR   4,4                 TEST FULL BUFFER
00007C  4770 2010  00012   47          BNZ   TESTINPT            BR NOT FULL
000080  9180 2128  0012A   48  TESTOTPT TM    TCOMM,X'80'         TEST TAPE COMPLETE
000084  4780 207E  00080   49          BZ    *-4
000088  9120 2128  0012A   50          TM    TCOMM,X'20'         TEST NON-OPERATIONAL TAPE
00008C  4710 20C4  000C6   51          BO    CANCEL
000090  4700 20BC  000BE   52  ENDSW   BC    0,ENDJOB
000094  5810 212A  0012C   53          L     1,TCCWAD            ALTERNATE BUFFERS
000098  5800 2166  00168   54          L     0,TALTAD
00009C  5000 212A  0012C   55          ST    0,TCCWAD
0000A0  5010 2166  00168   56          ST    1,TALTAD
```

SAMPLE PROGRAM NO. 2

```
FLGS   L.CTR    OBJCODE    ADDR    STMT      SOURCE STATEMENT

       0000A4   5811 0008  00008     57            L     1,8(1)
       0000A8   5010 216A  0016C     58            ST    1,BUFFNDX
       0000AC   4110 2126  00128     59            LA    1,TAPRCB            ADDRESS OUTPUT RCB
       0000B0   0A2B               60            SVC   43                  WRITE TAPE
       0000B2   4650 200C  0000E     61            BCT   5,INIT              DECREMENT BLOCK COUNTER
       0000B6   92F0 208F  00091     62            MVI   ENDSW+1,X'F0'
       0000BA   47F0 207E  00080     63            B     TESTOTPT            BR TO TEST LAST WRITE
       0000BE   4110 2176  00178     64  ENDJOB    LA    1,CLOSE
       0000C2   0A03               65            SVC   3                   CLOSE OUTPUT
       0000C4   0A2E               66            SVC   46                  EOJS
       0000C6   0A2F               67  CANCEL    SVC   47                  CANCEL FOR NON-OPERATIONAL
                                   68  *
                                   69  *
       0000C8                      70            DS    0F
       0000C8   1C                 71  AIRCB     DC    X'1C'              *SYS012 - ADC
       0000C9   40                 72            DC    X'40'               OPTION - MULTIPLE OPERATION
       0000CA   8000               73  ACOMM     DC    X'8000'             ASSEMBLE COMPLETE ON
       0000CC   00000108           74            DC    A(ADCCCW)
       0000D0                      75            DS    2F
       0000D8                      76  ASTATUS   DS    H                   CSW STATUS BYTES
       0000DA                      77            DS    H
       0000DC                      78  ASENSE    DS    F
       0000E0                      79            DS    2F
       0000E8   1D00               80  MRCB      DC    X'1D00'            *SYS013 - MPX CTRL(NO OPTIONS)
       0000EA   8000               81  MCOMM     DC    X'8000'             ASSEMBLE COMPLETE ON
       0000EC   00000110           82            DC    A(MPXCCW)
       0000F0                      83            DS    6F
       000108   2200011820000008   84  ADCCCW    CCW   X'22',INPUT,X'20',8  READ ADC
       000110   0100012000000008   85  MPXCCW    CCW   1,PTTAB,0,8          WRITE MPX CTRL
       000118                      86  INPUT     DS    4H                  ADC INPUT AREA
       000120   13E8               87  PTTAB     DC    X'13E8'             POINT ADDRESSES
       000122   1007               88            DC    X'1007'
       000124   1004               89            DC    X'1004'
       000126   1006               90            DC    X'1006'
       000128   1E00               91  TAPRCB    DC    X'1E00'            *SYS014 - TAPE(NO OPTIONS)
       00012A   8000               92  TCOMM     DC    X'8000'             ASSEMBLE COMPLETE ON
       00012C   00000158           93  TCCWAD    DC    A(TAPCCW2)
       000130                      94            DS    6F
       000148                      95  TAPCCW1   DS    D
       000150   0100018020000A0    96            CCW   1,BUFFER1,X'20',160
       000158                      97  TAPCCW2   DS    D
       000160   0100022020000A0    98            CCW   1,BUFFER2,X'20',160
       000168   00000148           99  TALTAD    DC    A(TAPCCW1)
       00016C   00000180          100  BUFFNDX   DC    A(BUFFER1)
       000170   80                101  OPEN      DC    X'80'
       000171   000174            102            DC    AL3(OPENTAPE)
       000174   12010100          103  OPENTAPE  DC    X'12010100'         OPEN TAPE - PARAMETERS
       000178   80                104  CLOSE     DC    X'80'
       000179   00017C            105            DC    AL3(CLOSTAPE)
       00017C   12010100          106  CLOSTAPE  DC    X'12010100'         CLOSE TAPE - PARAMETERS
       000180                     107  BUFFER1   DS    40F
       000220                     108  BUFFER2   DS    40F
                              +   109   LTORG
                          00000  110            END   BEGIN
```

SAMPLE PROGRAM NO. 3

```
FLGS   L.CTR    OBJCODE    ADDR    STMT      SOURCE STATEMENT

                                     2  * SAMPLE PROGRAM 3 IS AN ILLUSTRATION OF THE USE OF THE DAMPS SUBROU-
                                     3  * TINES PROVIDED FOR CONSTRUCTION OF THE RCB AND CHANNEL PROGRAM AND
                                     4  * FOR EXECUTION OF INPUT/OUTPUT.  THE PROGRAM PERFORMS THE SAME
                                     5  * FUNCTIONS AS SAMPLE 1.
       000000                        6           START 0
                                     7           EXTRN FRCBSU
                                     8           EXTRN FAIRAN
                                     9           EXTRN FTWRIT
                                    10           EXTRN FRTIO
                                    11           EXTRN FCHECK
       000000  0520                 12  BEGIN    BALR  2,0
                          00002     13           USING *,2
       000002  0A31                 14           SVC   49                  IDENTIFY RTJ
       000004  41D0 20DE  000E0     15           LA    13,SAVE
       000008  58F0 23BA  003BC     16           L     15,=A(FRCBSU)
       00000C  4110 2126  00128     17           LA    1,INBUILD
       000010  05EF                 18           BALR  14,15               BUILD RCB - INPUT
       000012  4110 21E2  001E4     19           LA    1,OUTBUILD
       000016  05EF                 20           BALR  14,15               BUILD RCB - OUTPUT
       000018  58F0 23B6  003B8     21           L     15,=A(FAIRAN)
       00001C  4110 2196  00198     22           LA    1,AIBUILD
       000020  05EF                 23           BALR  14,15               BUILD CHANNEL PROGRAM - INPUT
       000022  58F0 23C6  003C8     24           L     15,=A(FTWRIT)
       000026  4110 2212  00214     25           LA    1,WRBLD1
       00002A  05EF                 26           BALR  14,15               BUILD CHANNEL PROGRAM - OUTPUT
       00002C  4110 221E  00220     27           LA    1,WRBLD2
       000030  05EF                 28           BALR  14,15
       000032  4110 2262  00264     29           LA    1,OPEN
       000036  0A02                 30           SVC   2                   OPEN TAPE
       000038  4150 0032  00032     31           LA    5,50                OUTPUT BLOCK COUNTER
       00003C  4140 000A  0000A     32  INIT     LA    4,10                RECORDS/BLOCK
       000040  4110 21CE  001D0     33  TESTINPT LA    1,CHKIN
       000044  58F0 23C2  003C4     34           L     15,=A(FCHECK)
       000048  05EF                 35           BALR  14,15               CHECK INPUT OPERATION
       00004A  4700 2054  00056     36           BC    0,MOVINPT           FIRST TIME NOP
       00004E  92F0 2049  0004B     37           MVI   *-3,X'F0'
       000052  47F0 208A  0008C     38           B     READ
       000056  5830 2272  00274     39  MOVINPT  L     3,BUFFNDX           MOVE PREVIOUS INPUT DATA AND STATUS
                                    40  *                                   TO BUFFER
       00005A  5800 218E  00190     41           L     0,INPUT             STORE INPUT DATA
       00005E  5003 0000  00000     42           ST    0,0(3)
       000062  5800 2192  00194     43           L     0,INPUT+4
       000066  5003 0004  00004     44           ST    0,4(3)
       00006A  4800 2140  00142     45           LH    0,ACOMM             STORE ADC STATUS SUMMARY
       00006E  4003 0008  00008     46           STH   0,8(3)
       000072  4800 214E  00150     47           LH    0,ASTATUS           STORE ADC STATUS DETAIL
       000076  4003 000A  0000A     48           STH   0,10(3)
       00007A  5800 2152  00154     49           L     0,ASENSE            STORE ADC SENSE DATA
       00007E  5003 000C  0000C     50           ST    0,12(3)
       000082  4133 0010  00010     51           LA    3,16(3)             INCR BUFFER AREA POINTER
       000086  5030 2272  00274     52           ST    3,BUFFNDX
       00008A  0640                 53           BCTR  4,0                 DECR RCDS/BLOCK COUNTER
       00008C  4110 21CA  001CC     54  READ     LA    1,RDPAR
       000090  58F0 23BE  003C0     55           L     15,=A(FRTIO)
       000094  05EF                 56           BALR  14,15               INITIATE NEXT READ
```

SAMPLE PROGRAM NO. 3

```
FLGS   L.CTR    OBJCODE   ADDR    STMT      SOURCE STATEMENT

       000096  1244                 57            LTR   4,4                 TEST FULL BUFFER
       000098  4770 203E  00040     58            BNZ   TESTINPT            BR NOT FULL
       00009C  4110 2256  00258     59  TESTOTPT  LA    1,CHKOUT
       0000A0  58F0 23C2  003C4     60            L     15,=A(FCHECK)
       0000A4  05EF                 61            BALR  14,15               CHECK OUTPUT OPERATION
       0000A6  4700 20D6  000D8     62  ENDSW     BC    0,ENDJOB            EOJ SWITCH
       0000AA  5810 21F2  001F4     63            L     1,TCCWAD            ALTERNATE BUFFERS
       0000AE  5800 224E  00250     64            L     0,TALTAD
       0000B2  5000 21F2  001F4     65            ST    0,TCCWAD
       0000B6  5010 224E  00250     66            ST    1,TALTAD
       0000BA  5811 0008  00008     67            L     1,8(1)
       0000BE  5010 2272  00274     68            ST    1,BUFFNDX           INIT BUFFER INDEX
       0000C2  4110 2252  00254     69            LA    1,WRPAR
       0000C6  58F0 23BE  003C0     70            L     15,=A(FRTIO)
       0000CA  05EF                 71            BALR  14,15               WRITE OUTPUT
       0000CC  4650 203A  0003C     72            BCT   5,INIT              BR NOT EOJ
       0000D0  92F0 20A5  000A7     73            MVI   ENDSW+1,X'F0'       SET EOJ SWITCH ON
       0000D4  47F0 209A  0009C     74            B     TESTOTPT            BR TO CHECK LAST OUTPUT
       0000D8  4110 226A  0026C     75  ENDJOB    LA    1,CLOSE
       0000DC  0A03                 76            SVC   3                   CLOSE OUTPUT
       0000DE  0A2E                 77            SVC   46                  EOJS
                                    78  *
                                    79  *
       0000E0                       80  SAVE      DS    18F
       000128  00000140             81  INBUILD   DC    A(AIRCB)            PARAMETER LIST - INPUT RCB
       00012C  00000180             82            DC    A(ADCSYS)
       000130  000001B0             83            DC    A(ADCCCW)
       000134  00000184             84            DC    A(AOPT)
       000138  0000018C             85            DC    A(MPXSYS)
       00013C  80                   86            DC    X'80'
       00013D  0001B8               87            DC    AL3(MPXCCW)
       000140                       88  AIRCB     DS    H                   INPUT RCB
       000142                       89  ACOMM     DS    H
       000144                       90            DS    3F
       000150                       91  ASTATUS   DS    H
       000152                       92            DS    H
       000154                       93  ASENSE    DS    F
       000158                       94            DS    10F
       000180  0000001C             95  ADCSYS    DC    F'28'              *ADC - SYS012
       000184  00000004             96  AOPT      DC    F'4'                RCB OPTION - MULTIPLE OPERATION
       000188  00000004             97  ADCCT     DC    F'4'                READ COUNT - 4 POINTS
       00018C  0000001D             98  MPXSYS    DC    F'29'               MPX SYS013
       000190                       99  INPUT     DS    4H                  INPUT AREA
       000198  000001B0            100  AIBUILD   DC    A(ADCCCW)           PARAMETER LIST - INPUT CHANNEL PROG
       00019C  00000190            101            DC    A(INPUT)
       0001A0  00000188            102            DC    A(ADCCT)
       0001A4  000001C0            103            DC    A(AIOPT)
       0001A8  000001B8            104            DC    A(MPXCCW)
       0001AC  80                  105            DC    X'80'
       0001AD  0001C4              106            DC    AL3(PTTAB)
       0001B0                      107  ADCCCW    DS    D                   ADC CHANNEL PROGRAM
       0001B8                      108  MPXCCW    DS    D                   MPX CHANNEL PROGRAM
       0001C0  00000000            109  AIOPT     DC    F'0'                RANDOM ANALOG INPUT OPTION-STANDARD
       0001C4  13E8                110  PTTAB     DC    X'13E8'             INPUT POINT ADDRESSES
       0001C6  1007                111            DC    X'1007'
```

SAMPLE PROGRAM NO. 3

```
FLGS   L.CTR    OBJCODE   ADDR   STMT      SOURCE STATEMENT

       0001C8   1004               112            DC    X'1004'
       0001CA   1006               113            DC    X'1006'
       0001CC   00000140           114   RDPAR    DC    A(AIRCB)          READ - PARAMETER LIST
       0001D0   00000140           115   CHKIN    DC    A(AIRCB)          CHECK INPUT - PARAM LIST
       0001D4   000001E0           116            DC    A(RETURN)
       0001D8   80                 117            DC    X'80'
       0001D9   0001DC             118            DC    AL3(CHKOPT)
       0001DC   00000001           119   CHKOPT   DC    F'1'              CHECK OPTION - WAIT
       0001E0                      120   RETURN   DS    F                 CHECK RETURN CODE
       0001E4   000001F0           121   OUTBUILD DC    A(TAPRCB)         PARAMETER LIST - OUTPUT RCB
       0001E8   00000210           122            DC    A(TAPSYS)
       0001EC   80                 123            DC    X'80'
       0001ED   000240             124            DC    AL3(TAPCCW2)
       0001F0                      125   TAPRCB   DS    F                 OUTPUT RCB
       0001F4                      126   TCCWAD   DS    F
       0001F8                      127            DS    6F
       000210   0000001E           128   TAPSYS   DC    F'30'            *TAPE - SYS014
       000214   00000230           129   WRBLD1   DC    A(TAPCCW1)        PARAMETER LIST - OUTPUT CHANNEL PROG
       000218   00000278           130            DC    A(BUFFER1)
       00021C   80                 131            DC    X'80'
       00021D   00022C             132            DC    AL3(TAPCT)
       000220   00000240           133   WRBLD2   DC    A(TAPCCW2)        PARAMETER LIST - OUTPUT CHANNEL PROG
       000224   00000318           134            DC    A(BUFFER2)
       000228   80                 135            DC    X'80'
       000229   00022C             136            DC    AL3(TAPCT)
       00022C   000000A0           137   TAPCT    DC    F'160'            WRITE COUNT - 160 BYTES
       000230                      138   TAPCCW1  DS    2D                TAPE CHANNEL PROGRAM
       000240                      139   TAPCCW2  DS    2D                TAPE CHANNEL PROGRAM
       000250   00000230           140   TALTAD   DC    A(TAPCCW1)
       000254   000001F0           141   WRPAR    DC    A(TAPRCB)
       000258   000001F0           142   CHKOUT   DC    A(TAPRCB)         CHECK OUTPUT - PARAM LIST
       00025C   000001E0           143            DC    A(RETURN)
       000260   80                 144            DC    X'80'
       000261   0001DC             145            DC    AL3(CHKOPT)
       000264   80                 146   OPEN     DC    X'80'
       000265   000268             147            DC    AL3(OPENTAPE)
       000268   12010100           148   OPENTAPE DC    X'12010100'       OPEN TAPE - PARAMETERS
       00026C   80                 149   CLOSE    DC    X'80'
       00026D   000270             150            DC    AL3(CLOSTAPE)
       000270   12010100           151   CLOSTAPE DC    X'12010100'       CLOSE TAPE - PARAMETERS
       000274   00000278           152   BUFFNDX  DC    A(BUFFER1)
       000278                      153   BUFFER1  DS    40F               OUTPUT BUFFER 1
       000318                      154   BUFFER2  DS    40F               OUTPUT BUFFER 2
                              +    155    LTORG
       0003B8   00000000      +    156    DC   A(FAIRAN)
       0003BC   00000000      +    157    DC   A(FRCBSU)
       0003C0   00000000      +    158    DC   A(FRTIO)
       0003C4   00000000      +    159    DC   A(FCHECK)
       0003C8   00000000      +    160    DC   A(FTWRIT)
                         00000     161            END   BEGIN
```

```
C
C
C
C
C         SAMPLE PROGRAM NO. 4
C
C
C
C
C         RE-ENTRANCY TEST PROBLEM.
C
C         THIS EXAMPLE OF REENTRANCY TEST IS CONSISTED OF TWO REAL
C         TIME TASKS(RTT'S) AND TWO FOREGROUND TASKS(FGT'S) ONE IS THE
C         MAIN PROGRAM AND THE OTHER THE TERMINATING PROGRAM.
C         BOTH REAL TIME TASKS COMPUTE THE SAME MATHEMATICAL
C         FUNCTIONS BUT THE PIL'S ARE SET UP IN DIFFERENT
C         TIME INTERVALS SO THAT AN INTERRUPT WILL OCCUR
C         WHILE COMPUTING A MATHEMATICAL FUNCTION AND THIS
C         FUNCTION WILL BE IMMEDIATELY CALLED BY THE OTHER ROUTINE.
C         IN ORDER TO ASCERTAIN THAT AN INTERRUPT
C         REALLY OCCURED WE COMPUTE ONE OF THE ROUTINES
C         THE INITIAL ONE   TO TEST FOR DISCREPANCY. IF AN
C         6000 TIMES. IN THE BEGINNING THE FUNCTIONS ARE
C         COMPUTED ONCE. THEN EACH TIME THE FUNCTIONS
C         ARE RECOMPUTED THE RESULTS ARE COMPARED AGAINST
C         ERROR OCCURED THE PROGRAM BRANCHES OUT TO
C         TERMINATING ROUTINE.
C
C
          DIMENSION T(6), X(6), Y(6)
          DIMENSION A(6), B(6), C(6)
          DIMENSION F(100), G(100)
          DIMENSION E(100), H(100)
          DIMENSION  ITRCB(8)
          DIMENSION  TPCCW(2)
          DOUBLE PRECISION  TPCCW
          COMPLEX TC1, TC2, CL1, CL2, C1, C2
          COMMON T, X, Y
          COMMON A, B, C
          COMMON F, G
          COMMON E, H
          COMMON  ITRCB, TPCCW
          COMMON  TC1, TC2, CL1, CL2, C1, C2
          COMMON IEOJ
          EXTERNAL PRG1, PRG2
          DOUBLE PRECISION SAVE1D
          DIMENSION  SAVE1D(4)
          DOUBLE PRECISION SAVE1(31)
          DOUBLE PRECISION SAVE2D
          DIMENSION SAVE2D(4)
          DOUBLE PRECISION SAVE2(31)
C
C         TELLS THE SUPERVISOR THAT THIS IS A REAL TIME JOB
C
          CALL FIDRTJ
C
C
C         BUILD RCB AND CHANNEL PROGRAM FOR TAPE WRITING
C
          CALL FRCBSU(ITRCB, 28, TPCCW)
          CALL FTWRIT(TPCCW, X, 24)
C
C         ATTACH PRIORITY INTERRUPTS TO THE RTT
C
          CALL FATACH (PRG1, SAVE1, 1,  SAVE1D)
    110   CALL FATACH (PRG2, SAVE2, 16, SAVE2D)
          CALL FENPIL(255, 1, 16)
          CALL FDONE
          END
```

```
C
C         PRIMARY PROGRAM PRG1
C
          SUBROUTINE PRG1
          DIMENSION  T(6),X(6), Y(6)
          DIMENSION A(6), B(6), C(6)
          DIMENSION F(100), G(100)
          DIMENSION E(100), H(100)
          DIMENSION  ITRCB(8)
          DIMENSION  TPCCW(2)
          DOUBLE PRECISION  TPCCW
          COMPLEX TC1, TC2, CL1, CL2, C1, C2
          COMMON T, X, Y
          COMMON A, B, C
          COMMON F, G
          COMMON E, H
          COMMON  ITRCB, TPCCW
          COMMON  TC1, TC2, CL1, CL2, C1, C2
          COMMON IEOJ
          EXTERNAL TERM
    390   C1 = (2.0,3.0)
          Y(1) = 26.0
          Y(2) = .020
          Y(3) = .025
          Y(4) = 100.0
          Y(5) = .010
          CALL FUNMSK(ITRCB)
C
C         COMPUTE ROUTINES ONCE AND SET SWITCH TO PREVENT FROM BEING
C         RECOMPUTED AGAIN
C
          IF(M.EQ.13579) GO TO 399
          M = 13579
          IEOJ = 0
          TC1 = CLOG(C1)
          T(1) = SQRT(Y(1))
          T(2) = EXP(Y(2))
          T(3) = ALOG(Y(3))
          T(4) = ALOG10(Y(4))
          T(5) = SIN(Y(5))
          T(6) = COS(Y(5))
    399   IF(K.EQ.98765) GO TO 400
          K = 98765
          N = 0
C
C         THIS ROUTINE MUST LOOP AND COMPUTE THE ROUTINES 2000 TIMES
C         EACH TIME THE RESULTS ARE COMPARED AGAINST THE ORIGINAL
C         COMPUTATIONS AND IF A DISCREPANCY IS DETECTED THE RUN
C         IS IMMEDIATELY TERMINATED.
C
    400   X(1) = SQRT(Y(1))
          IF(X(1) - T(1)) 405, 401, 405
    401   N = N + 1
          CALL FCHECK(ITRCB, IRET)
          IF(IRET.EQ.4)  GO TO 100
          IF(IRET.NE.0) GO TO 404
          CALL FRTIO(ITRCB)
    100   IF(N.EQ.6000) GO TO 404
          CALL FRETRN
    404   CALL FDSPIL(0, 1, 16)
          IEOJ = 1
          CALL MQUEUE(0, TERM, IR1, 0, 0)
          CALL FRETRN
    405   CALL FCANCL
          END
```

```
          C
          C         THIS IS THE SECOND RTT PROGRAM WHEN THE FIRST GETS
          C         INTERRUPTED THIS TAKES OVER BOTH USE THE SAME
          C         ROUTINES TO TEST THEM FOR REENTRANCY
          C
          C
          C         SECONDARY PROGRAM PRG2
          C
0001                SUBROUTINE PRG2
0002                DIMENSION T(6), X(6), Y(6)
0003                DIMENSION A(6), B(6), C(6)
0004                DIMENSION F(100), G(100)
0005                DIMENSION E(100), H(100)
0006                DIMENSION  ITRCB(8)
0007                DIMENSION  TPCCW(2)
0008                DOUBLE PRECISION  TPCCW
0009                COMPLEX TC1, TC2, CL1, CL2, C1, C2
0010                COMMON T, X, Y
0011                COMMON A, B, C
0012                COMMON F, G
0013                COMMON E, H
0014                COMMON  ITRCB, TPCCW
0015                COMMON  TC1, TC2, CL1, CL2, C1, C2
0016                COMMON IEOJ
0017                C2 = (.60,.80)
0018                A(1) = 30.0
0019                A(2) = .030
0020                A(3) = .027
0021                A(4) = 115.0
0022                A(5) = .015
0023                IF(L.EQ.02468) GO TO 409
0024                L = 02468
0025                TC2 = CLOG(C2)
0026                B(1) = SQRT(A(1))
0027                B(2) = EXP(A(2))
0028                B(3) = ALOG(A(3))
0029                B(4) = ALOG10(A(4))
0030                B(5) = SIN(A(5))
0031                B(6) = COS(A(5))
0032          409   C(1) = SQRT(A(1))
0033                CALL FRETRN
0034                END
```

```
          C
0001                SUBROUTINE TERM
          C
          C         TERMINATE REAL TIME JOB
          C
0002                CALL FEORTJ
0003                CALL FDONE
0004                END
```

```
C
C        SAMPLE 5
C
C  THIS PROGRAM WILL READ A TAPE AND WRITE IT TO THE 1827.  THIS
C  TAPE WAS PREPARED BY A PREVIOUS PROGRAM (LOGIN) AND IS A DIGITAL
C  ENCODING OF ANALOG SIGNALS FED TO THE 1827.
C
      REAL *8 SAVE1D(4),SAVE1(31)
      INTEGER *4 RCBA1(8),RCBA2(8),RCBT1(8),RCBT2(8),RCBT3(8)
      INTEGER *4 CCWA1(2),CCWA2(2),CCWT1(4),CCWT2(4),CCWT3(4)
      INTEGER *2 ARRAY1(2001),ARRAY2(2001)
      INTEGER *2 SWITCH,ERRCNT,TAPE
      COMMON/COMM/ RCBA1,RCBA2,RCBT1,RCBT2,RCBT3
      COMMON /COMM/CCWA1,CCWA2,CCWT1,CCWT2,CCWT3
      COMMON /COMM/ARRAY1,ARRAY2,SWITCH,ERRCNT,TAPE
      EXTERNAL  PRG1
C
      ERRCNT=0
      TAPE=1
      SWITCH = 1
      ARRAY1(1)=6
      ARRAY2(1)=6
      CALL  FIDRTJ
C
C  BUILD THE RCBS
C
      CALL  FRCBSU(RCBA1,30,CCWA1)
      CALL  FRCBSU(RCBA2,30,CCWA2)
      CALL  FRCBSU(RCBT1,28,CCWT1)
      CALL  FRCBSU(RCBT2,28,CCWT2)
      CALL  FRCBSU(RCBT3,28,CCWT3)
C
C  BUILD THE CCWS
C
      CALL  FDAO(CCWA1,ARRAY1,2000,12)
      CALL  FDAO(CCWA2,ARRAY2,2000,12)
      CALL  FTREAD(CCWT1,ARRAY1(2),4000)
      CALL  FTREAD(CCWT2,ARRAY2(2),4000)
      CALL  FTREW(CCWT3)
C
C  REWIND THE TAPE
C
  100 CALL  FRTIO(RCBT3)
      CALL  FCHECK(RCBT3,IRETT3,1)
      IF ( IRETT3.NE.0 ) CALL  FCANCL
C
C
C  ISSUE A READ TO READ THE DUMMY RECORD PLACED ON THE TAPE BY
C  THE LOGIN PROGRAM.  THIS DUMMY RECORD WAS PLACED THERE TO MOVE
C  THE TAPE FROM THE LOAD POINT.
C
      CALL  FRTIO(RCBT1)
      CALL  FCHECK(RCBT1,IRETT1,1)
      IF ( IRETT1.EQ.0 ) GO TO 1000
C
      ERRCNT=ERRCNT+1
      GO TO 100
C
C  READ THE FIRST TWO TAPE RECORDS
C
 1000 CALL  FRTIO(RCBT1)
      CALL  FRTIO(RCBT2)
C
      CALL  FCHECK(RCBT1,IRETT1,1)
      IF ( IRETT1.EQ.0 ) GO TO 2000
C
C  INCREMENT THE TAPE ERROR COUNTER AND CONTINUE
C
      ERRCNT=ERRCNT+1
C
C  ATTACH THE RTT
C
 2000 CALL  FATACH(PRG1,SAVE1,7,SAVE1D)
      CALL  FENPIL(255,7)
      CALL  FDONE
      END
```

```
0001              SUBROUTINE  PRG1
0002              INTEGER *4 RCBA1(8),RCBA2(8),RCBT1(8),RCBT2(8),RCBT3(8)
0003              INTEGER *4 CCWA1(2),CCWA2(2),CCWT1(4),CCWT2(4),CCWT3(4)
0004              INTEGER *2 ARRAY1(2001),ARRAY2(2001)
0005              INTEGER *2 SWITCH,ERRCNT,TAPE
0006              INTEGER *2 SYSUN1,SYSUN2
0007              COMMON/COMM/ RCBA1,RCBA2,RCBT1,RCBT2,RCBT3
0008              COMMON /COMM/CCWA1,CCWA2,CCWT1,CCWT2,CCWT3
0009              COMMON /COMM/ARRAY1,ARRAY2,SWITCH,ERRCNT,TAPE
            C
0010              GO TO (500,1000,2000,3000,4000),SWITCH
            C
0011           500  CALL  FRTIO(RCBA1)
            C
0012              SWITCH = 2
0013              SYSUN1=8704
0014              SYSUN2=8960
            C
            C  CHECK FOR THE COMPLETION OF THE TAPE READ INTO SECOND BUFFER
            C
0015           1000 CALL  FCHECK(RCBT2,IRETT2)
0016              IF (IRETT2.EQ.0) GO TO 1100
0017              IF (IRETT2.EQ.1) GO TO 6000
0018              IF (IRETT2.EQ.4) CALL FRETRN
            C
            C  INCREMENT THE TAPE ERROR COUNTER AND CONTINUE
            C
0019              ERRCNT=ERRCNT+1
            C
            C  WHEN BUFFER 2 IS FULL WRITE IT TO THE 1827
            C
0020           1100 CALL  FRTIO(RCBA2)
0021              SWITCH = 3
            C
            C  WAIT FOR THE FIRST BUFFER TO BE WRITTEN TO THE 1827
            C
0022           2000 CALL  FCHECK(RCBA1,IRETA1)
0023              IF (IRETA1.EQ.0) GO TO 2100
0024              IF (IRETA1.EQ.4 ) CALL FRETRN
0025              CALL  FCANCL
            C
            C  REFILL BUFFER ONE FROM TAPE
            C
0026           2100 CALL  FRTIO(RCBT1)
0027              SWITCH = 4
0028              CALL  FRETRN
            C
            C  WAIT FOR THE BUFFER TO BE FILLED
            C
```

```
0029          3000 CALL  FCHECK(RCBT1,IRETT1)
0030               IF (IRETT1.EQ.0) GO TO 3100
0031               IF (IRETT1.EQ.1) GO TO 6000
0032               IF (IRETT1.EQ.4) CALL FRETRN
            C
            C  INCREMENT THE TAPE ERROR COUNTER AND CONTINUE
            C
0033               ERRCNT=ERRCNT+1
            C
            C  WHEN THE BUFFER IS FILLED WRITE IT TO THE 1827
            C
0034          3100 CALL  FRTIO(RCBA1)
0035               SWITCH = 5
            C
            C  WAIT FOR THE SECOND BUFFER TO BE EMPTIED BY THE 1827
            C
0036          4000 CALL  FCHECK(RCBA2,IRETA2)
0037               IF (IRETA2.EQ.0) GO TO 4100
0038               IF (IRETA2.EQ.4) CALL FRETRN
0039               CALL  FCANCL
            C
            C  FILL THE BUFFER FROM THE TAPE
            C
0040          4100 CALL  FRTIO(RCBT2)
0041               SWITCH = 2
0042               CALL  FRETRN
            C
            C
            C  AN END OF TAPE HAS BEEN DETECTED.  REWIND THE TAPE AND SWITCH TAPE
            C  DRIVES.
            C
0043          6000 GO TO (6010,6020),TAPE
            C
0044          6010 RCBT3(1)=SYSUN1
0045               GO TO 6050
            C
0046          6020 RCBT3(1)=SYSUN2
            C
0047          6050 CALL  FRTIO(RCBT3)
            C
            C  CHANGE THE SYSTEM UNIT INDEX IN THE RCB
            C
0048               GO TO (6200,6300),TAPE
            C
0049          6200 RCBT1(1)=SYSUN2
0050               RCBT2(1)=SYSUN2
0051               TAPE=2
0052               GO TO 6400
            C
0053          6300 RCBT1(1)=SYSUN1
0054               RCBT2(1)=SYSUN1
0055               TAPE=1
            C
            C  RETURN TO WRITING
            C
0056          6400 GO TO (6400,1100,6400,3100,6400),SWITCH
            C
0057               END
```

```
C     SAMPLE PROGRAM NO. 6
C
C     THIS PROGRAM WILL TEST THE FIRST FIVE LEVELS OF THE PRIORITY
C     INTERRUPT SPECIAL FEATURE .
C     EACH TIME A PRIORITY INTERRUPT OCCURS , THE ASSOCIATED LEVEL
C     NUMBER WILL BE PRINTED.
C
C     SET ASIDE FLOATING POINT REGISTER SAVE AREAS
      DOUBLE PRECISION FPSV0(4)
      DOUBLE PRECISION FPSV1(4)
      DOUBLE PRECISION FPSV2(4)
      DOUBLE PRECISION FPSV3(4)
      DOUBLE PRECISION FPSV4(4)
C     SET ASIDE SAVE AREA OF 61 WORDS  FOR EACH  RTT
         DOUBLE PRECISION SAVE0(31)
         DOUBLE PRECISION SAVE1(31)
         DOUBLE PRECISION SAVE2(31)
         DOUBLE PRECISION SAVE3(31)
         DOUBLE PRECISION SAVE4(31)
      COMMON I
      EXTERNAL RTT0,RTT1,RTT2,RTT3,RTT4
      I=-1
      CALL FIDRTJ
C     ATTACH  RTT'S  TO APPROPRIATE  LEVEL
      CALL FATACH(RTT0,SAVE0,0,FPSV0)
      CALL FATACH(RTT1,SAVE1,1,FPSV1)
      CALL FATACH(RTT2,SAVE2,2,FPSV2)
      CALL FATACH(RTT3,SAVE3,3,FPSV3)
      CALL FATACH(RTT4,SAVE4,4,FPSV4)
      CALL FENPIL(0,0,1,2,3,4)
C     BEGIN  A  ONE HOUR LOOP
      DO1K=1,7000
      DO1J=1,3000
C     IF A PI(PRIORITY INTERUPT)OCCURS AT LEVEL N  I IS SET EQUAL TO N
C     AND THEN PRINTED  AND THEN SET  BACK TO -1
      IF(I.EQ.-1)GO TO 1
      WRITE(6,2)I
2     FORMAT('OPRIORITY INTERRUPT DETECTED AT LEVEL'I3)
      I=-1
1     A=SQRT(FLOAT(K+J))
C
C     VARIABLE   'I' CHANGED FROM -1 ONLY IF A PRIORITY INTERUPT HAS
C     BEEN  ACCEPTED.  PRIORITY  INTERRUPTS ARE CAUSED,MOST SIMPLY  BY
C     CLOSING  THE CONTACT  BETWEEN  TWO  CORRESPONDING  SERPENT
C     CONNECTORS  ON  CPU  TAILGATE. 'TIMING OUT' CONTACT MUST BE
C     CONNECTED INTO 'TIMING IN'. SEE  FORM  A22-6900.
      CALL FEORTJ
      END
```

```
      SUBROUTINE RTT0
      COMMON I
      I=0
      CALL FRETRN
      END
```

```
      SUBROUTINE RTT1
      COMMON I
      I=1
      CALL FRETRN
      END
```

```
0001          SUBROUTINE RTT2
0002          COMMON I
0003          I=2
0004          CALL FRETRN
0005          END
```

```
0001          SUBROUTINE RTT3
0002          COMMON I
0003          I=3
0004          CALL FRETRN
0005          END
```

```
0001          SUBROUTINE RTT4
0002          COMMON I
0003          I=4
0004          CALL FRETRN
0005          END
```

```
        C     SAMPLE PROGRAM NO. 7
        C
        C     THIS PROGRAM READS 1000 TIMES AT 1851 RELAY MULTIPLEXOR
        C     ADDRESS POINT 6 AND AT SOLID STATE MULTIPLEXOR ADDRESS
        C     POINT 9  (4105-4096)
0001          DOUBLE PRECISION CCW1,CCW2,CCW3,CCW4
0002          DIMENSION AIRCB(16),INA1(1000),INPT(1000)
0003          INTEGER*2 INA1,INPT
0004          DIMENSION HSINPT(1000),HSRCB(16)
0005          INTEGER*2 HSINPT
        C     DEFINE INPUT ADDRESSES
0006          DO 1 I=1,1000
0007          HSINPT(I)=4105
0008    1     INPT(I)=6
0009          CALL FIDRTJ
        C     SET UP RCB FOR RANDOM ANALOG INPUT
0010          CALL FRCBSU(AIRCB,28,CCW1,4,29,CCW2)
0011          CALL FRCBSU(HSRCB,28,CCW3,4,29,CCW4)
        C     SIGNIFY 14 BIT PRECISION
0012          IOPT=01
        C     CONSTRUCT ACTUAL CHANNEL PROGRAMS
0013          CALL FAIRAN(CCW3,INA1,1000,IOPT,CCW4,HSINPT)
0014          CALL FAIRAN(CCW1,INA1,1000,IOPT,CCW2,INPT)
0015    4      CONTINUE
        C     EXEC REAL TIME I/O ,PRINT RESULTS, PAUSE
0016          CALL FRTIO(AIRCB)
0017          WRITE(6,3)INA1
0018    3     FORMAT('0'15I7)
0019          CALL FRTIO(HSRCB)
0020          WRITE(6,3)INA1
0021          PAUSE  10
        C     PAUSE PERMITS ADJUSTMENT OF THE INPUTS TO THE MULTIPLEXORS
        C     SUITABLE INPUT MIGHT BE 10,000 HZ SQUARE WAVE ON MPX/S OR10 HZ ON MPX/R
0022          GO TO 4
        C     HIT EOB ON CONSOLE TO TERMINATE PAUSE
0023          END
```

```
      C     SAMPLE PROGRAM NO. 8
      C
      C     THIS PROGRAM DEMONSTRATES USE OF DOUBLE BUFFERS FOR SIMULTANEOUS
      C     OUTPUT ON 2 POINTS OF THE 1856
      C
0001        DOUBLE PRECISION AOCCW1(2),AOCCW2(2)
0002        DIMENSION AORCB1(8),AORCB2(8)
0003        INTEGER*2 IOA1(3),IOA2(3),CTLA
      C
0004        CTLA=64
      C
      C     CLTA=64 INDICATES WE WANT TO USE DOUBLE BUFFER FACILITY
      C
0005        IOA1(1)=0
0006        IOA2(1)=0
      C
0007        IOA1(2)=670
0008        IOA1(3)=1340
      C
      C     670 IS ABOUT .1 VOLTS  ON OUTPUT TERMINAL
      C
0009        IOA2(2)=0
0010        IOA2(3)=0
      C
0011        CALL FIDRTJ
0012        CALL FRCBSU(AORCB1,28,AOCCW1)
0013        CALL FRCBSU(AORCB2,28,AOCCW2)
0014        CALL FDAO(AOCCW1,IOA1,2,16,CTLA)
0015        CALL FDAO(AOCCW2,IOA2,2,0)
      C
      C     BUFFERS ASSUMED PRESENT ON POINTS 0,1  ,ABSENT ON POINTS  2,3
      C
0016  9     CALL FRTIO(AORCB1)
0017        CALL FRTIO(AORCB2)
0018        GO TO 9
      C
      C     TO SEE  THE EFFECT OF THE BUFFERS, SUPERIMPOSE POINTS 0,1  ON SWEEP
      C     OSCILLOSCOPE. NOTE RISE CURVE SHAPE . REPEAT  PROGRAM  WITH CTLA=0
      C
0019        END
```

```
        C     SAMPLE PROGRAM NO. 9
        C
        C        THIS PROGRAM TESTS DIGITAL INPUT.
        C
        C     VARIABLES  BEGINNING  IN  DI,DO,AI,AO  RESPECTIVELY SIGNIFY
        C     DIGITAL INPUT, DIGITAL OUTPUT,ANALOG INPUT,ANALOG OUTPUT
        C
        C     VARIABLES ENDING IN  RCB,CCW,CNT,IOA  RESPECTIVELY  SIGNIFY
        C     AREAS SET ASIDE FOR REQUEST CONTROL BLOCKS, CHANNEL PROGRAMS,
        C     COUNT VARIABLES, AND AREAS WHICH CONTAIN THE ACTUAL  DATA
        C     FOR THE I/O  OPERATION
        C
        C
        C     RESERVE THE NECESSARY DOUBLE WORD,FULL WORD,AND HALF WORD AREAS
        C     AS REQUIRED FOR CHANNEL PROGRAMS, RCBS,AND I/O AREAS
        C
        C     THIS , LIKE ALL THESE PROGRAMS IS SOMEWHAT INSTALLATION DEPENDANT
        C     FOR EXAMPLE DEVICE ADDRESSES,AND THE PHYSICAL  CHARACTERISTICS
        C     OF THE FRONT END REAL TIME HARDWARE
        C
        C     SET UP  AREA FOR CHANNEL PROGRAM
        C     DOUBLE PRECISION MAKES SURE CHANNEL PGM IS ALLIGNED ON DOUBLE WORD BNDRY
0001          DOUBLE PRECISION DICCW(16)
        C     CREATE AREA FOR RCB
0002          DIMENSION DIRCB(8)
        C     SPECIFY HALF WORD AREA FOR  INPUT AREAS, POINT ADDRESSES
0003          INTEGER*2INA(8),IPTADD(8)
0004          INTEGER*2M1
0005          DATA M1/-1/
        C     THIS PROGRAM WILL READ UP TO 8 DIG INPUT GROUPS OF 16 POINTS
0006          DATA IPTADD/64,65,66,67,68,69,70,71/
0007          IOPT=0
0008          ICNT=8
        C     IDENTIFY REAL TIME JOB
0009          CALL FIDRTJ
        C     CREATE RCB
0010          CALL FRCBSU(DIRCB,28,DICCW)
        C     CREATE CHANNEL PROGRAM
0011          CALL FDIG(DICCW,INA,ICNT,IPTADD,IOPT)
        C     UNMASK CHANNEL(IF NECSSARY)
0012          CALL FUNMSK(DIRCB)
0013    69    DO 3 IJ=1,522
        C     EXEC REAL TIME I/O  (IE READ DIG INPUT PTS64-71)
0014          CALL FRTIO(DIRCB)
        C     PRINT RESULTS OF A DIGITAL INPUT DETECT OR PRINT  'HERE'
0015          DO 50 I=1,8
0016    50    IF(INA(I).NE.M1)WRITE(6,11)INA
0017    11    FORMAT(1H 8I12)
0018    3     CONTINUE
0019          WRITE(6,65)
0020    65    FORMAT(' HERE')
0021          GO TO 69
        C     CANCEL BY TYPING  'CANCEL' OR PUSHING PSW RESTART
0022          END
```

```
C     SAMPLE PROGRAM NO. 10
C
C     THIS PROGRAM ILLUSTRATES THE USE OF ANALOG OUTPUT.
C
      DOUBLE PRECISION AOCCW
      DIMENSION AORCB(8),AOIOA(17)
      INTEGER*2 AOIOA
C
      AOIOA(1)=0
C
C     WE ARE WRITING ON 16 CONSECUTIVE ANALOG POINTS STARTING WITH ZERO
C
      CALL FIDRTJ
      CALL FRCBSU(AORCB,28,AOCCW)
      CALL FDAO(AOCCW,AOIOA,16,0)
C
C     THIS FDAO CALL SETS UP FOR 16  POINTS TO BE WRITTEN,STANDARD MODE
C
C     THIS NEXT LOOP SETS UP THE VALUES TO BE WRITTEN (EACH OF 16 POINTS
C     GETS SAME VALUE)  6600 = 1 VOLT  ON TERMINAL( 2 VOLTS IF YOU HAVE
C     DRIVER AMPLIFIERS)
C     USE VOLTMETER TO SEE EFFECT
C
6     DO 69 K=1,4
      DO 5 L=2,17
C
5     AOIOA(L)=6600*(K-1)
C
C     LOOP FOR ABOUT 1 SECOND
C
      DO 2 I=1,7000
      A=I
2     A=SQRT(A)
C
C     PERFORM I/O OPERATION - FOUR  STEPS OF ONE VOLT AT ONE SEC. INTRVLS
C
      CALL FRTIO(AORCB)
69    CONTINUE
      GO TO 6
C
      END
```

```
C     SAMPLE PROGRAM NO. 11
C
C     THIS PROGRAM ILLUSTRATES THE USE OF ANALOG OUTPUT BY
C     PLOTTING SEVERAL MATHEMATICAL CURVES ON A MEMORY-OSCILLOSCOPE.
C     MAIN ROUTINE
      INTEGER*2 OUTPUT(3),FINISH,INPT
C
      COMMON DELTA,JUNIT,THETA,OUTPUT,FINISH,INPT
      EXTERNAL  CIRCLE, CARD, HYPCYC
C
      CALL FIDRTJ
C
C     UNIT 29 IS SYS013
      JUNIT = 29
      OUTPUT(1)=4
      THETA = 0.0
      DELTA = .05
      INPT = 4105
C
         CALL MQUEUE(3,CIRCLE,IRTRN,INT,0)
      IF (IRTRN .NE. 0) CALL FCANCL
         CALL MQUEUE(5,CARD,IRTRN,INT,2)
      IF (IRTRN .NE. 0) CALL FCANCL
         CALL MQUEUE(7,HYPCYC,IRTRN,INT,3)
      IF (IRTRN .NE. 0) CALL FCANCL
C
      CALL FDONE
      END
```

```
0001          SUBROUTINE CIRCLE
0002          INTEGER*2 OUTPUT(3),FINISH,INPT
0003          DOUBLE PRECISION CHNPRG,AICC1,MPXCC
0004          COMMON DELTA,JUNIT,THETA,OUTPUT,FINISH,INPT
0005          DIMENSION RCB1(8),RCB2(16)
        C
        C     SET UP RCB'S AND CHANNEL PROGRAMS
0006          CALL FRCBSU(RCB1,JUNIT,CHNPRG)
0007          CALL FRCBSU(RCB2,30,AICC1,4,17,MPXCC)
0008          CALL FDAO (CHNPRG, OUTPUT, 2)
0009          CALL FAIRAN(AICC1,FINISH,1,1,MPXCC,INPT)
        C
0010       10 DO 25 L = 1,2000
0011       20 OUTPUT(2) = 2047*SIN(THETA)+ 2047
0012          OUTPUT(3) =2047 *COS(THETA)+2047
0013          CALL FCHECK(RCB1,IRTRN,1)
0014          IF (IRTRN.NE.0) CALL FCANCL
0015          CALL FRTIO(RCB1)
0016          THETA = THETA + DELTA
0017          THETA = AMOD(THETA,6.2832)
0018       25 CONTINUE
        C
        C        TEST FOR END OF ROUTINE
0019          CALL FRTIO(RCB2)
0020          CALL FCHECK(RCB2,I,1)
0021          IF (I .NE. 0) CALL FCANCL
0022          IF (FINISH .GT. 0) GO TO 10
0023          WRITE(6,99)FINISH
0024    99    FORMAT('0FINISH='I9)
        C
0025          CALL FDONE
0026          END
```

```
0001          SUBROUTINE CARD
0002          INTEGER*2 OUTPUT(3),FINISH,INPT
0003          DOUBLE PRECISION CHNPRG,AICC1,MPXCC
0004          COMMON DELTA,JUNIT,THETA,OUTPUT,FINISH,INPT
0005          DIMENSION RCB1(8),RCB2(16)
        C
        C     SET UP RCB'S AND CHANNEL PROGRAMS
0006          CALL FRCBSU(RCB1,JUNIT,CHNPRG)
0007          CALL FRCBSU(RCB2,30,AICC1,4,17,MPXCC)
0008          CALL FDAO (CHNPRG, OUTPUT, 2)
0009          CALL FAIRAN(AICC1,FINISH,1,1,MPXCC,INPT)
        C
0010       10 DO 25 L = 1,2000
0011       20 OUTPUT(2) =  682 *(2*COS(THETA) - COS(2*THETA)) + 2046
0012          OUTPUT(3) =  682 *(2*SIN(THETA) - SIN(2*THETA)) + 2046
0013          CALL FCHECK(RCB1,IRTRN,1)
0014          IF (IRTRN.NE.0) CALL FCANCL
0015          CALL FRTIO(RCB1)
0016          THETA = THETA + DELTA
0017          THETA = AMOD(THETA,6.2832)
0018       25 CONTINUE
        C
        C        TEST FOR END OF ROUTINE
0019          CALL FRTIO(RCB2)
0020          CALL FCHECK(RCB2,I,1)
0021          IF (I .NE. 0) CALL FCANCL
0022          IF (FINISH .GT. 0) GO TO 10
0023          WRITE(6,99)FINISH
0024    99    FORMAT('0FINISH='I9)
        C
0025          CALL FDONE
0026          END
```

```
0001              SUBROUTINE HYPCYC
0002              INTEGER*2 OUTPUT(3),FINISH,INPT
0003              DOUBLE PRECISION CHNPRG,AICC1,MPXCC
0004              COMMON DELTA,JUNIT,THETA,OUTPUT,FINISH,INPT
0005              DIMENSION RCB1(8),RCB2(16)
            C
            C     SET UP RCB'S AND CHANNEL PROGRAMS
0006              CALL FRCBSU(RCB1,JUNIT,CHNPRG)
0007              CALL FRCBSU(RCB2,30,AICC1,4,17,MPXCC)
0008              CALL FDAO (CHNPRG, OUTPUT, 2)
0009              CALL FAIRAN(AICC1,FINISH,1,1,MPXCC,INPT)
            C
0010           10 DO 25 L = 1,2000
0011           20 OUTPUT(2) =  2047 * SIN(THETA)**3 + 2047
0012              OUTPUT(3) =  2047 * COS(THETA)**3 + 2047
0013              CALL FCHECK(RCB1,IRTRN,1)
0014              IF (IRTRN.NE.0) CALL FCANCL
0015              CALL FRTIO(RCB1)
0016              THETA = THETA + DELTA
0017              THETA = AMOD(THETA,6.2832)
0018           25 CONTINUE
            C
            C        TEST FOR END OF ROUTINE
0019              CALL FRTIO(RCB2)
0020              CALL FCHECK(RCB2,I,1)
0021              IF (I .NE. 0) CALL FCANCL
0022              IF (FINISH .GT. 0) GO TO 10
0023              WRITE(6,99)FINISH
0024        99    FORMAT('0FINISH='I9)
            C
0025               CALL FEORTJ
0026              END
```

```
            C     SAMPLE PROGRAM NO. 12
            C
            C     THIS PROGRAM GENERATES A FOUR-LEAF ROSE ON MEMORY-OSCILLOSCOPE
            C     ATTACHED TO POINT 4,5 OF  1856 ANALOG OUTPUT
0001              DOUBLE PRECISION CCW
0002              INTEGER*2 XN(2000),YN(2000)
0003              INTEGER*2 ARRAY(8000)
0004              DATA ARRAY(1)/4/
0005              DIMENSION NRCB(8)
0006              CALL FIDRTJ
0007              CALL FRCBSU(NRCB,28,CCW)
0008              CALL FDAO(CCW,ARRAY,4000,3)
0009              CALL FUNMSK(NRCB)
0010              AAK=6.283185/1999.
0011              DO 20 I=1,2000
0012              R=I-1
0013              TH=R*AAK
0014              XN(I)=32767.*COS(TH)*SIN(2.*TH)
0015              YN(I)=32767.*SIN(TH)*SIN(2.*TH)
0016        20    CONTINUE
0017              DO 30 I=1,2000
0018              K=4*I
0019              ARRAY(K-3)=4
0020              ARRAY(K-2)=XN(I)
0021              ARRAY(K-1)=5
0022              ARRAY(K)=YN(I)
0023          30  CONTINUE
0024        10    CONTINUE
0025              CALL FRTIO(NRCB,IRET)
0026              CALL          FCHECK(NRCB,IJRET,1)
0027              GO TO 10
0028        21    CALL EXIT
0029              END
```

BIBLIOGRAPHY

The required material has been divided into five categories, according to the needs of various users.

Category 1. All readers must have a thorough understanding of the material covered in the following documents:

IBM System/360 Principles of Operation (A22-6821)

IBM System/360 Model 44 Functional Characteristics (A22-6875)

Data Acquisition Special Features for the IBM System/360 Model 44 (A22-6900)

IBM System/360 Model 44 Programming System: Concepts and Facilities (C28-6810)

Category 2. DAMPS Assembler Language programmers must have a thorough understanding of the following documents, in addition to those in category 1.

IBM System/360 Model 44 Programming System: Assembler Language (C28-6811)

IBM System/360 Model 44 Programming System: Guide to System Use (C28-6812)

Category 3. DAMPS FORTRAN programmers must have a thorough understanding of the following documents, in addition to those in category 1.

IBM System/360 FORTRAN IV Language (C28-6515)

IBM System/360 Model 44 Programming System: Guide to System Use for FORTRAN Programmers (C28-6813)

Category 4. All DAMPS programmers who wish to make use of the 1827 DCU must be familiar with the following documents, in addition to thoroughly understanding those in category 2 or category 3.

IBM System/360 Component Description, IBM 1827 Data Control Unit (A22-6868)

IBM 1800 - System/360 Adapter and 1827 Data Control Unit: Programmer's Guide (C26-3707)

Category 5. Any system construction or modification requires a thorough understanding of the following document (see the DAMPS Application Directory for additional requirements in this area).

IBM System/360 Model 44 Programming System: System Programmer's Guide (C28-6814)

Printed in U.S.A. H20-0537-1

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
(USA Only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

scientific resources corporation

hybrid systems, inc.

SS-100 analog/hybrid computer patching reference manual

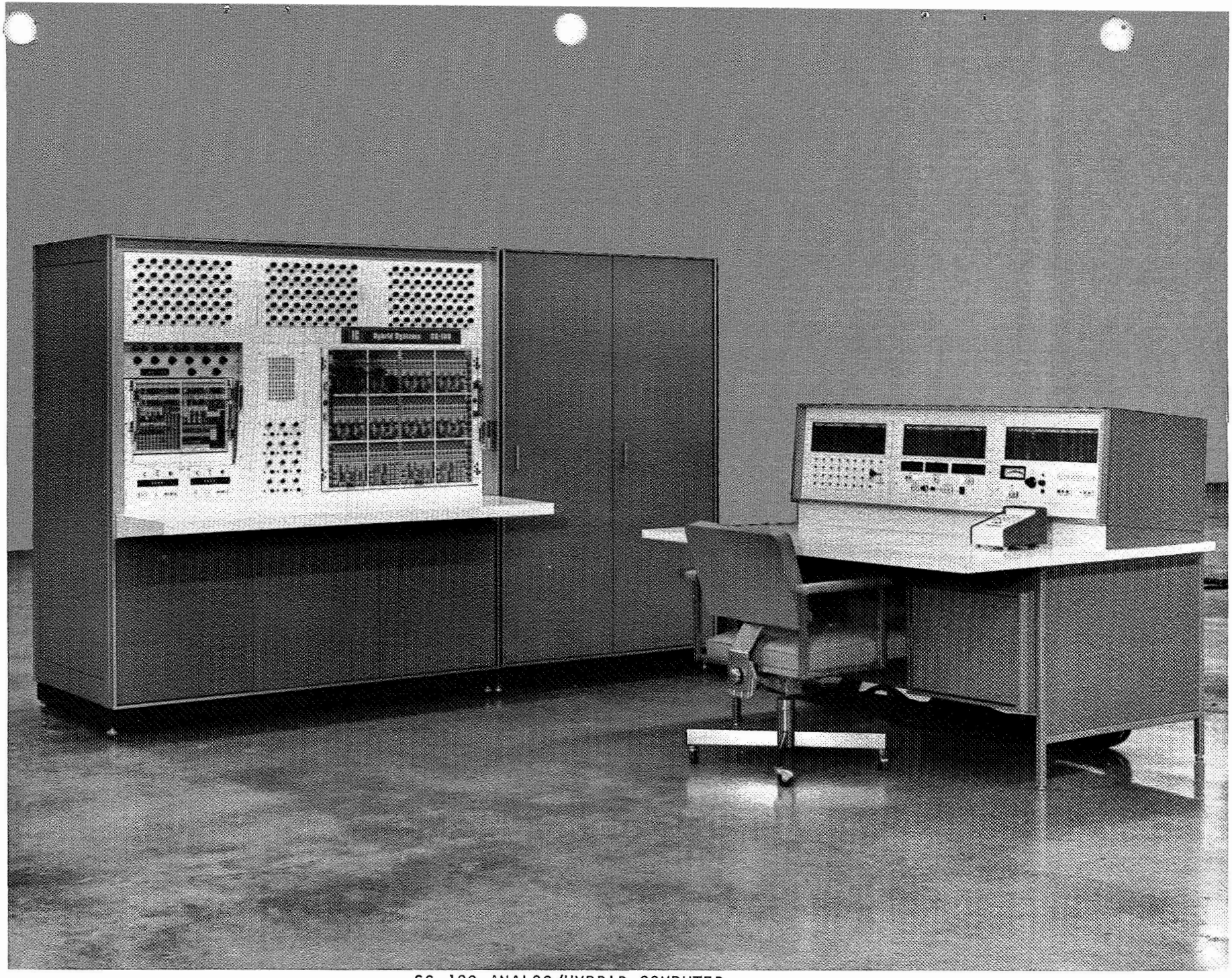
SS-100 ANALOG/HYBRID COMPUTER

PATCHING REFERENCE MANUAL

HYBRID SYSTEMS, INC.
A SUBSIDIARY OF
SCIENTIFIC RESOURCES CORPORATION

Copyright, 1969, by Hybrid Systems, Inc., Houston, Texas.
Printed in the United States of America. This book, or parts thereof, may not be reproduced in any form without permission of Hybrid Systems, Inc.

Publication No. HS/40001.0
1/69



SS-100 ANALOG/HYBRID COMPUTER

page iv blank

PREFACE

This manual is written specifically for the SS-100 Analog/Hybrid Computer operator. It is single, complete reference source for patching all components on the computer. For detailed information on the theory of operation, maintenance or adjustment of the various components, and other related information, refer to the following publications:

SS-100 Analog/Hybrid Computer,
System Reference Manual,
Publication No. HS/30001.0

SS-100 Analog/Hybrid Computer,
Maintenance Reference Manual,
Publication No. HS/30002.0

SS-100 Analog/Hybrid Computer,
Computer Programming Techniques,
Publication No. HS/40002.0

SS-100 Analog/Hybrid Computer Specifications,
Publication No. 6016.0

page vi blank

TABLE OF CONTENTS

Section/Paragraph	Page
1. GENERAL DESCRIPTION	1-1
1.1 Analog Patch Panel	1-1
1.2 Logic Patch Panel	1-2
1.3 Patch Cords	1-2
1.4 Helpful Hints During Patching	1-12
2. COMPONENT PATCHING	2-1
2.1 Analog Elements	2-1
2.1.1 Potentiometers	2-2
2.1.2 Summer/Inverter	2-4
2.1.3 Summer/Integrator	2-6
2.1.4 Multiplier	2-8
2.1.5 Multiplier Output Amplifier Gain	2-10
2.1.6 Divider	2-12
2.1.7 Square Root	2-14
2.1.8 Fixed Diode Function Generator	2-16
2.1.9 Log X Diode Function Generator	2-18
2.1.10 Manual Diode Function Generator	2-20
2.1.11 Resolver	2-22
2.1.11.1 Resolver Normal Mode	2-24
2.1.11.2 Forward Resolution	2-26
2.1.11.3 Double Forward Resolution	2-28
2.1.11.4 Inverse Resolution	2-30
2.1.11.5 Rotation of Axis	2-32
2.1.11.6 Resolver Rate Generator	2-34
2.1.12 Free Components/Bridge Limiter	2-36

TABLE OF CONTENTS (CONT'D)

		Page
2.1.13	D-R Converter/Sample-Hold Amplifier	2-38
2.1.14	Sample-Hold Amplifier/High Speed Integrator . . .	2-40
2.1.15	Function Switch/Function Relay	2-42
2.1.16	Comparator	2-46
2.1.17	Digital-to-Analog Switch	2-48
2.2	Logic Elements	2-51
2.2.1	Integrator Mode and Time Scale Patching	2-52
2.2.2	Gates	2-56
2.2.3	Flip-Flops	2-58
2.2.4	Counter/Register	2-60
2.2.5	One Shot	2-62
2.2.6	Differentiator	2-64
2.2.7	Logic Pushbutton and General Purpose Indicator . .	2-66
2.2.8	Preset Thumbwheel Switches	2-68

Appendix

A Patch Panels Termination Coding and Functions A-1

LIST OF ILLUSTRATIONS

Figure		Page
Frontispiece	SS-100 Analog/Hybrid Computer	iii
1-1	Photograph of the Analog Patch Panel	1-3
1-2	Analog Patch Panel Layout	1-5
1-3	Photograph of the Logic Patch Panel	1-7
1-4	Logic Panel Mode Control and Time Scale Selection Sector Arrangments	1-9
1-5	Photograph of Analog Patching Accessories	1-10
1-6	Photograph of Logic Patching Accessories	1-11
2-1	Potentiometer Patching	2-3
2-2	Summer/Inverter Patching	2-5
2-3	Summer/Integrator Patching	2-7
2-4	Multiplier Patching	2-9
2-5	Multiplier Output Amplifier (Gain Control Option) Patching	2-11
2-6	Divider Patching	2-13
2-7	Square Root Patching	2-15
2-8	Fixed Diode Function Generator Patching	2-17
2-9	Log X Diode Function Generator Patching	2-19
2-10	Manual Diode Function Generator Patching	2-21
2-11	Resolver Patching	2-23
2-12	Resolver - Normal Mode Patching	2-25
2-13	Forward Resolution (1R) Patching	2-27
2-14	Double Forward Resolution (2R) Patching	2-29
2-15	Inverse Resolution (INV) Patching	2-31
2-16	Rotation of Axis Patching	2-33
2-17	Resolver Rate Generator Patching	2-35
2-18	Free Components and Bridge Limiter Patching	2-37
2-19	D-R Converter and Sample - Hold Amplifier Patching	2-39

LIST OF ILLUSTRATIONS (CONT'D)

Figure		Page
2-20	Sample-Hold Amplifier/High Speed Integrator Patching . . .	2-41
2-21	Function Switch and Function Relay Patching	2-43/45
2-22	Comparator Patching	2-47
2-23	Digital-to-Analog Switch Patching	2-49
2-24	Integrator Mode and Time Scale Patching	2-53
2-25	Gates Patching	2-57
2-26	Flip-Flops Patching	2-59
2-27	Counter/Register Patching	2-61
2-28	One Shot Patching	2-63
2-29	Differentiator Patching	2-65
2-30	Logic Pushbutton and General Purpose Indicator Patching .	2-67
2-31	Preset Thumbwheel Switches Patching	2-69

LIST OF TABLES

Table		Page
A-1	Analog Patch Panel Color Scheme	A-2
A-2	Analog Patch Panel Termination Coding and Functions	A-3
A-3	Logic Patch Panel Termination Coding and Functions	A-7

1. GENERAL DESCRIPTION

The analog patch panel and the logic patch panel are component units of the SS-100 Analog/Hybrid Computer mainframe, which contains all computing components and patch panels. The patch panels are removable from the mainframe for patching and/or storage for later use.

Each hole in the patch panels is color coded and designated according to its function. Patch cords are typically inserted in the patch panels while panels are removed from the patchbay; however, cords also can be inserted while the panels are engaged. Patch cords are available in assorted lengths; the shortest possible connection between holes enhances neatness and bandwidth.

When panels have been patched remotely from the mainframe, the panels are simply slid into the patchbay and locked in place. The action of installing the patch panel in the patchbay causes the patch cord plug tips to wipe the receptacle contact springs to assure positive connection.

1.1 ANALOG PATCH PANEL

The analog patch panel contains a total of 3672 holes, each of which is color coded and designated in accordance with their individual functions. Figure 1-1 is a pictorial view in color of the unpatched analog patch panel. Table A-1 in Appendix A delineates the color scheme; Table A-2 gives a complete listing of coding (designation and color) and function of each hole in the analog patch panel.

To facilitate in identifying and locating the various components, the panel is divided into groups of components known as sectors. There are six sectors as shown in Figure 1-2. The components is a sector are numbered from 1 through some upper limit usually 30. This component number is added to the three digit sector number to obtain the component address. For example, the address of amplifier 7 in sector 100 is A107. Potentiometer 11 in the 500 sector has an address of P511. Any component can be quickly located on the panel once the numbering system and general panel layout is understood.

1.2 LOGIC PATCH PANEL

The logic patch panel as pictured in Figure 1-3 has 1632 holes. The coding and functions of each hole in the logic patch panel is given in Table A-3 in the Appendix A. This panel uses a different sector arrangement and numbering system than the analog panel. Because of the smaller complement of equipment, each different type of component is numbered sequentially from 1 to its upper limit. However, the exception to this scheme is the numbering and arrangement of the integrator mode control and time scale patching holes. They are numbered to correspond to the related integrator amplifier addresses; the physical arrangement closely duplicates the sector arrangement for these functions only as illustrated in Figure 1-4.

1.3 PATCH CORDS

There are two types of patch cords provided to wire the SS-100 patch panels. To provide full shielding, the cords for the analog patch panel are made of coaxial cable. The center conductor of the coax carries the signal while the outer braid completes the shielding through the ground at the aluminum patch panel. Patch cord sizes of 5, 7, 9, 11, 15, 19, 27 and 36 inches are available. These cords are color coded for easy identification. Also available are shunt or bottle plugs for patching adjacent holes on the panel and an eight-input commoning block for off-the-panel multiple connections. See Figure 1-5.

A second type of cord is used for patching the logic patch panel. These cords are a single unshielded wire type and are available in sizes of 5, 7, 9, 11, and 15 inches. Shunt plugs and six-hole commoning blocks for logic panel patching are also available. See Figure 1-6.

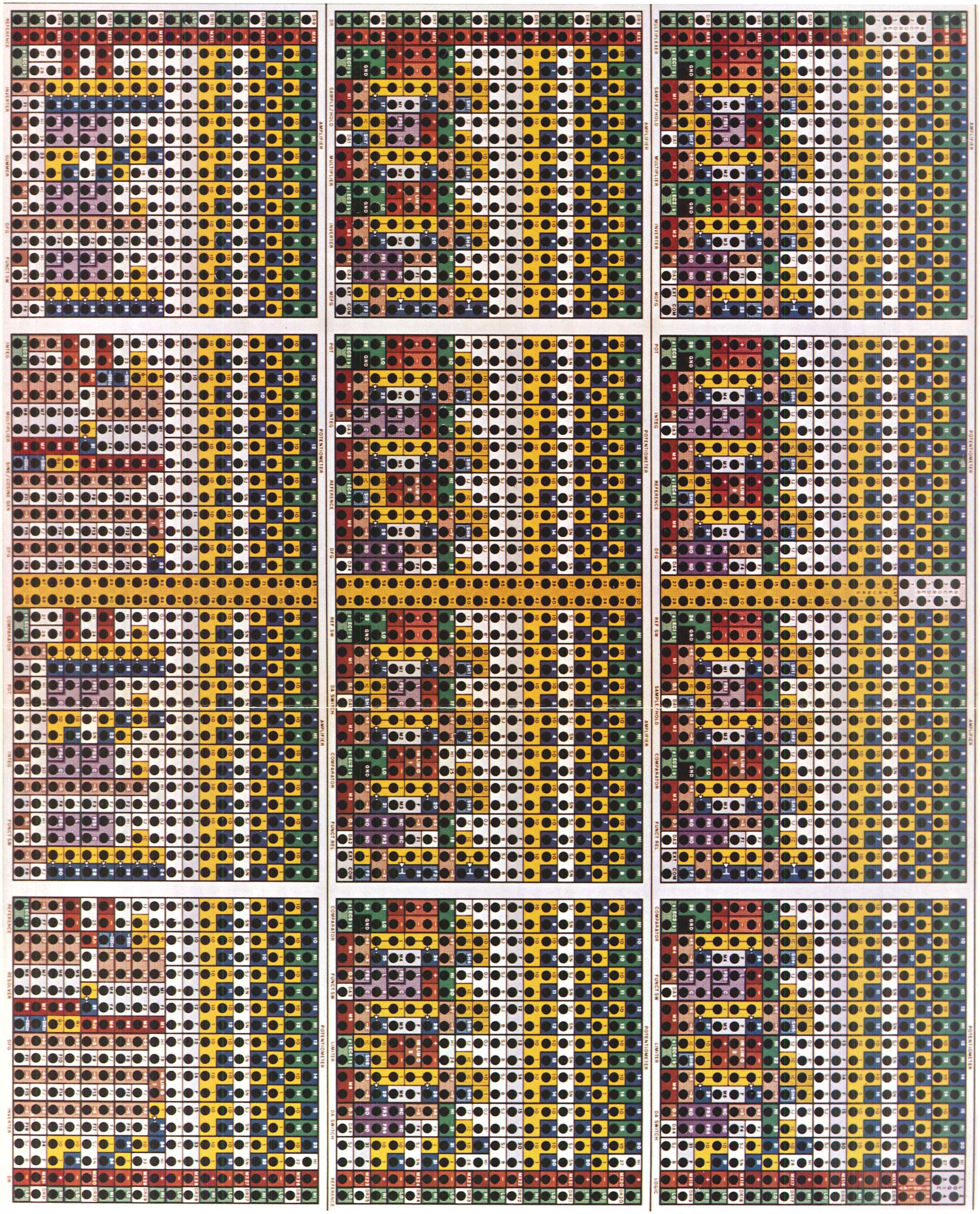


FIGURE 1-1 PHOTOGRAPH OF ANALOG PATCH PANEL

Page 1-4 blank

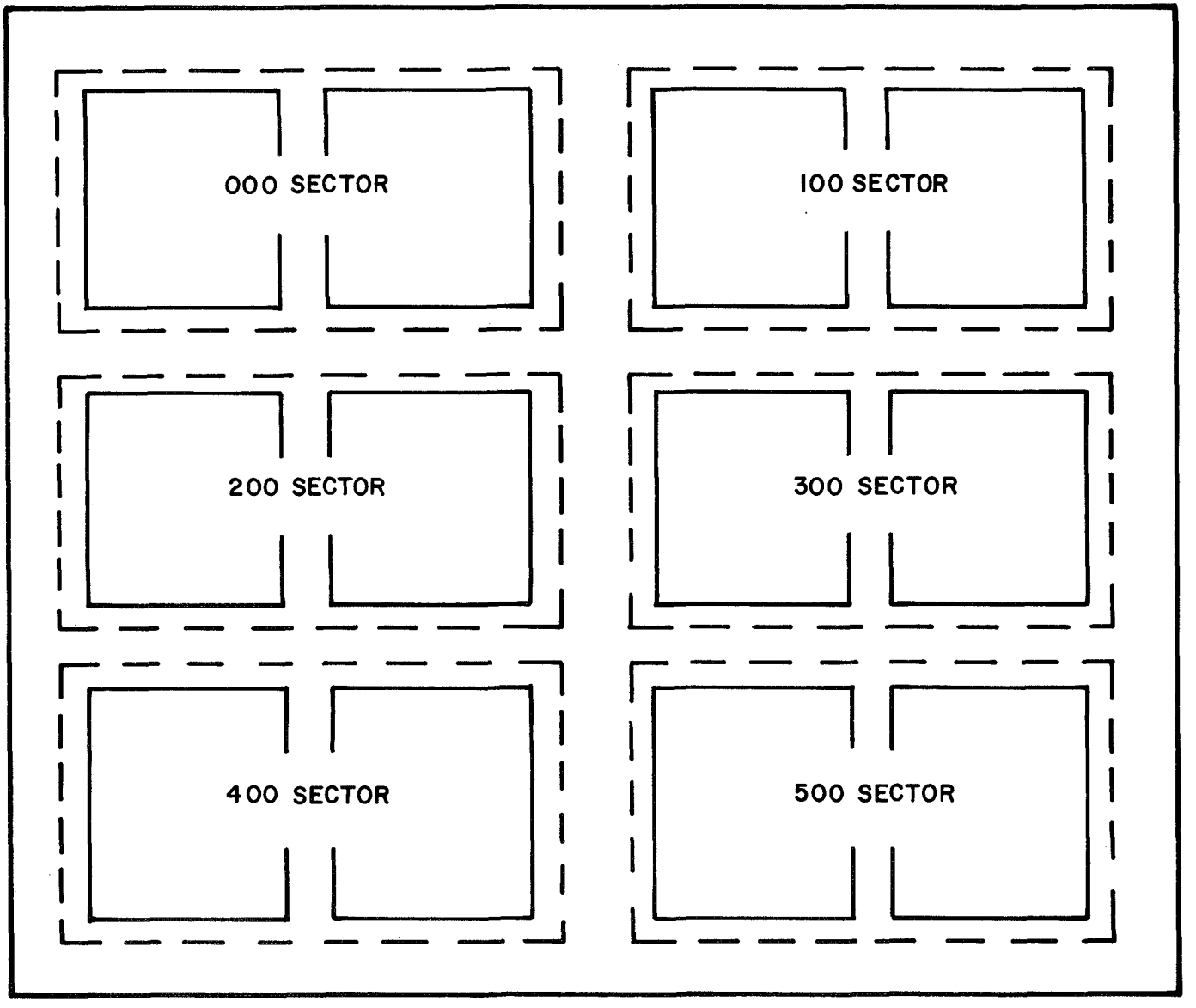


FIGURE 1-2 ANALOG PATCH PANEL LAYOUT

page 1-6 blank

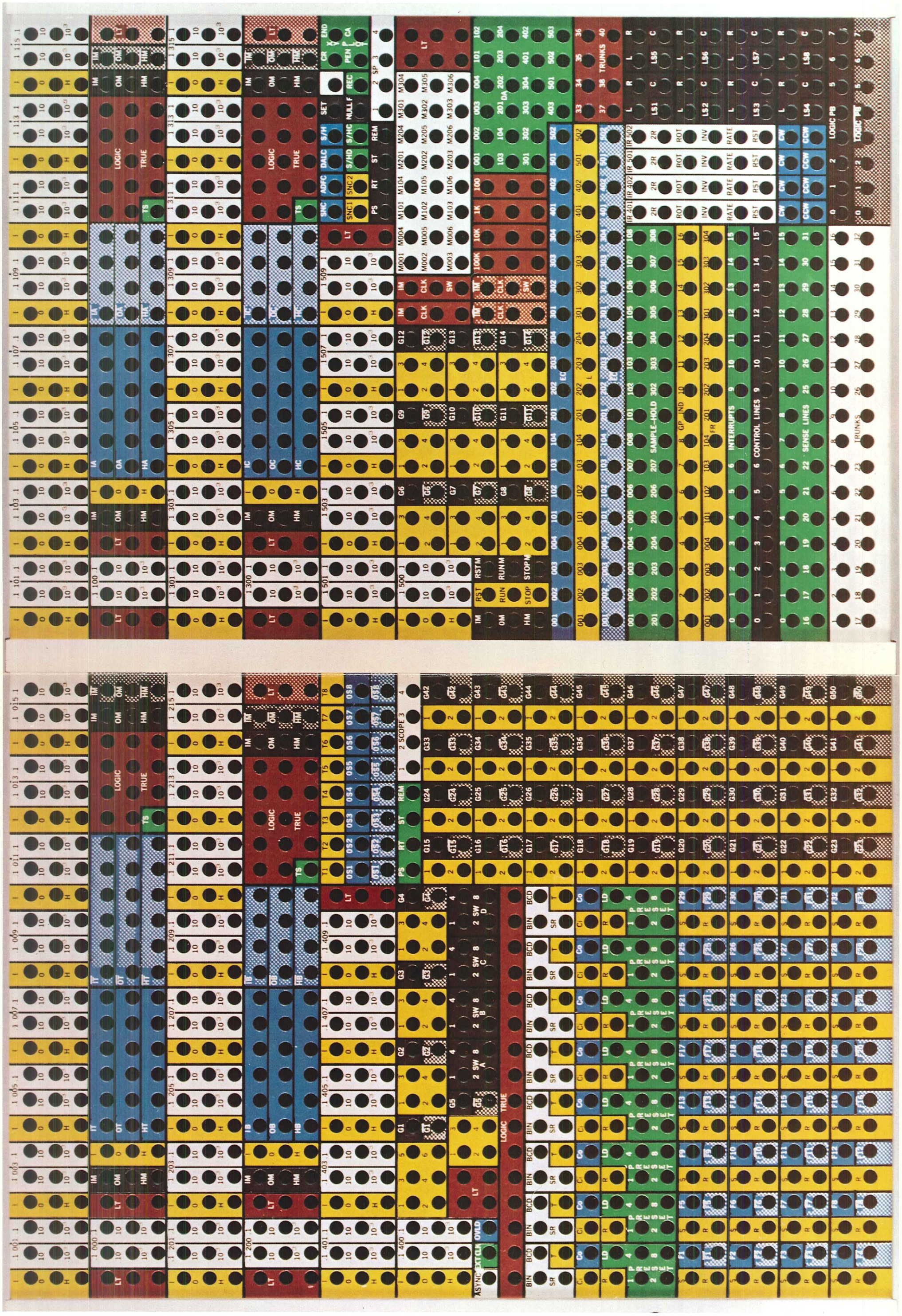

LOGIC
TRUE
SAMPLE-HOLD
INTERRUPTS
CONTROL LINES
SENSE LINES
TRUNKS
LOGIC PB
LOGIC PB

FIGURE 1-3 PHOTOGRAPH OF LOGIC PATCH PANEL

page 1-8 blank

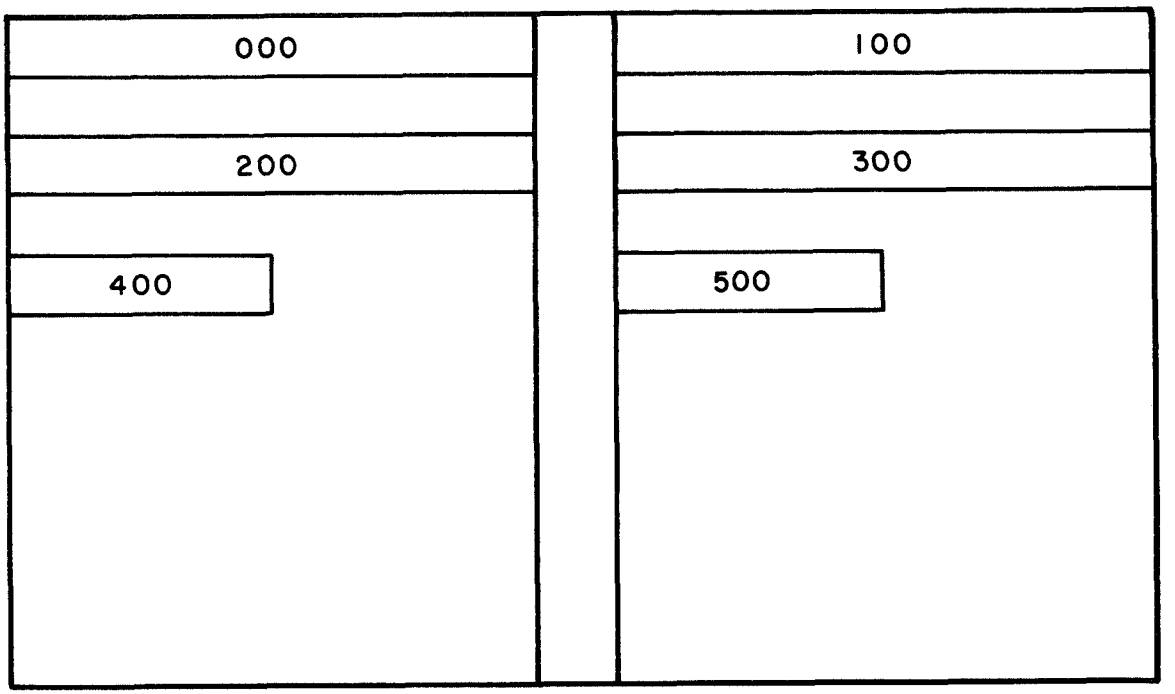


FIGURE 1-4 LOGIC PANEL
MODE CONTROL AND TIME SCALE
SELECTION SECTOR ARRANGEMENTS

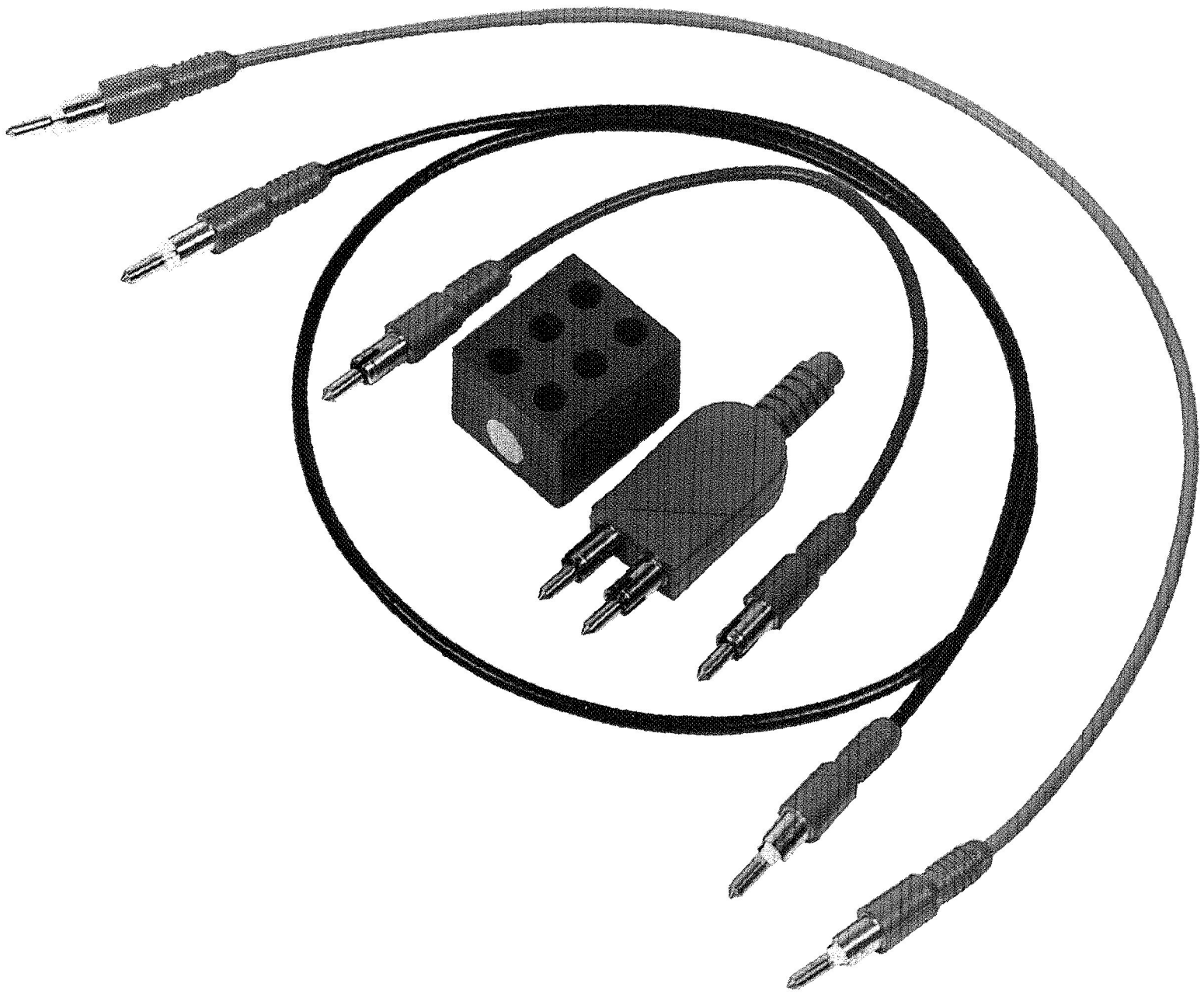


FIGURE 1-5 PHOTOGRAPH OF ANALOG PTACHING ACCESSORIES

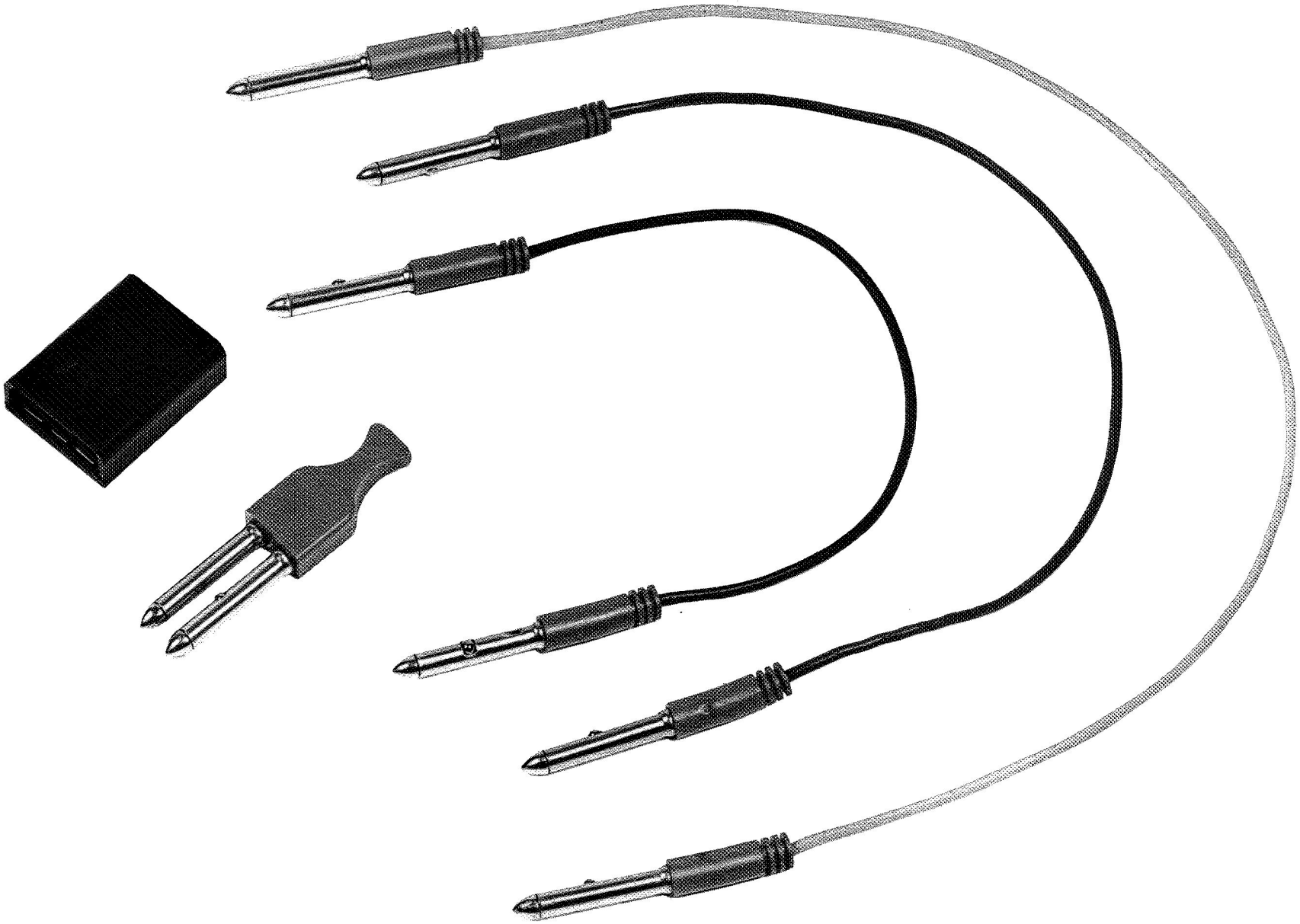


FIGURE 1-6 PHOTOGRAPH OF LOGIC PATCHING ACCESSORIES

1.4 HELPFUL HINTS DURING PATCHING

The following helpful hints are recommended to facilitate easy and efficient patching.

Patch panels remotely from the mainframe. The patchboard includes protective rails to permit patching while on a flat surface. The programmed patchboard may be stored for later use or reuse.

Use an analog diagram representing the problem to be solved during patching. Component numbers should be assigned to the various computing elements prior to patching rather than making assignments during patching; this technique speeds up the patching process and minimizes patching errors. In addition, after a connection is made, "red line" the analog diagram showing that the connection was made. This also minimizes errors and aids in maintaining your place during interruptions.

While patching, keep all connections as short and direct as possible. Panel wiring will be neater and optimum bandwidth will be achieved. Use bottle plugs for patching adjacent holes. Arrange the components in your problem so that lead lengths are kept short.

2. COMPONENT PATCHING

This section contains complete details on patching each of the computing elements in the mainframe. The analog patch panel is covered in Subsection 2.1, while the logic patch panel is covered in Subsection 2.2. Complete instructions for patching each element are given; but for detailed information on holes not identified in the patching descriptions, refer to Tables A-1 thru A-3 in Appendix A.

2.1 ANALOG ELEMENTS

While each analog computing component was designed for a specific purpose, usually it can be used for a variety of other operations if patched properly.

In the following paragraphs, the patching of each element in its designated mode is shown; but where other useful and obvious arrangements are possible, they are described. All possible applications are not covered, however, and reference to an analog computer programming and applications text is recommended.

2.1.1 Potentiometers

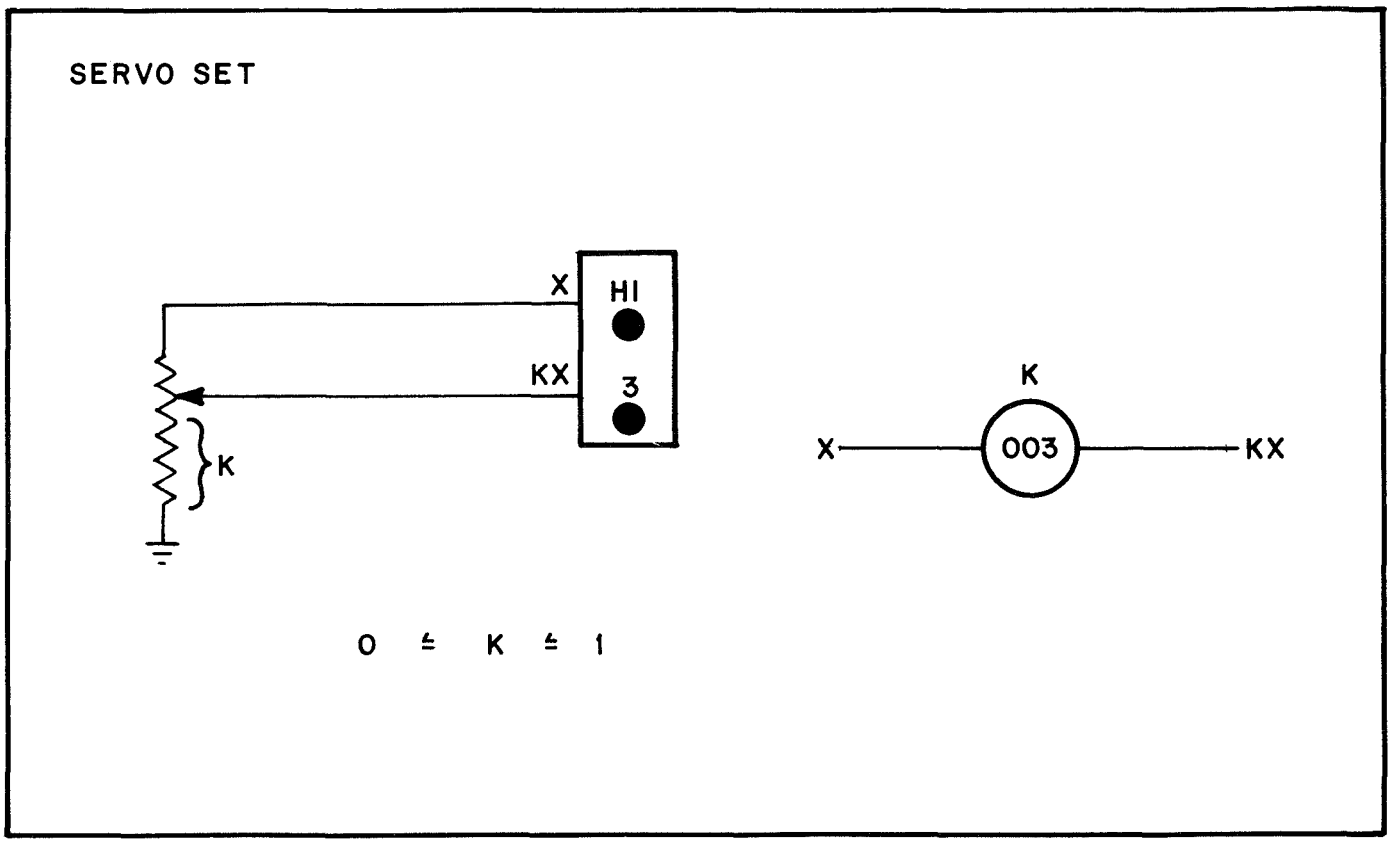
The SS-100 contains two different types of potentiometers: servo set and hand set (see Figure 2-1). Both are 10-turn units with a setting accuracy of 0.015%. The servo pots can be set from the digital computer, the keyboard, or manually by the operator with the front panel knob. One end of the resistance element of the servo set pots is permanently connected to ground. The high side and the pot arm terminate at the analog panel. A total of 120 servo pots are in a fully expanded system.

There are 12 hand set pots in the SS-100. The low side of these pots is not permanently wired to ground. Instead, all three pot connections terminate at the analog panel. This permits a variety of useful operations to be performed. If desired, the low side of the pot may be patched to ground for conventional operation.

Each hand set pot is also provided with a three-position switch that selects the input to the high side of the pot. In the center position, the patch panel input is applied to the pot. In either the left or right positions, +100 or -100 volts reference will be applied. When reference voltage is applied through the switch, the low side of the pot is automatically grounded. The hand set pot is also provided with a calibrated, locking dial.

Each pot in the system is supplied with a 1/32 ampere fuse in series with the arm to protect against shorts to ground and incorrect patching.

POTENTIOMETER PATCHING



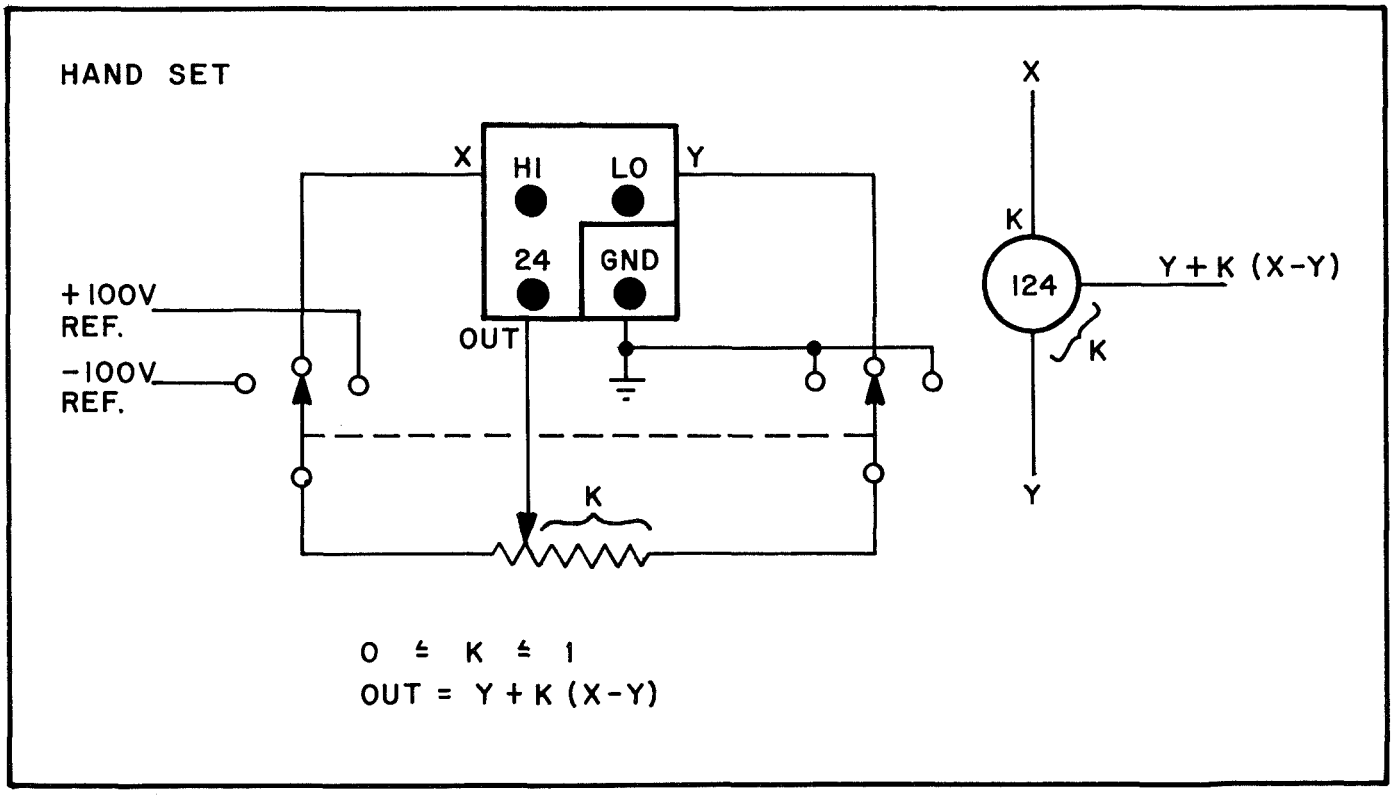


FIGURE 2-1

2.1.2 Summer/Inverter

The summer/inverter amplifier can be used in a variety of modes by patching the various inputs and outputs as desired. Figure 2-2 shows the standard configuration.

Patching SJ to B connects the input summing network to the input or base of the amplifier. This provides two gain 1 inputs and a gain 10 input. Patching the amplifier output 12 of F connects the standard 1 megohm feedback resistor.

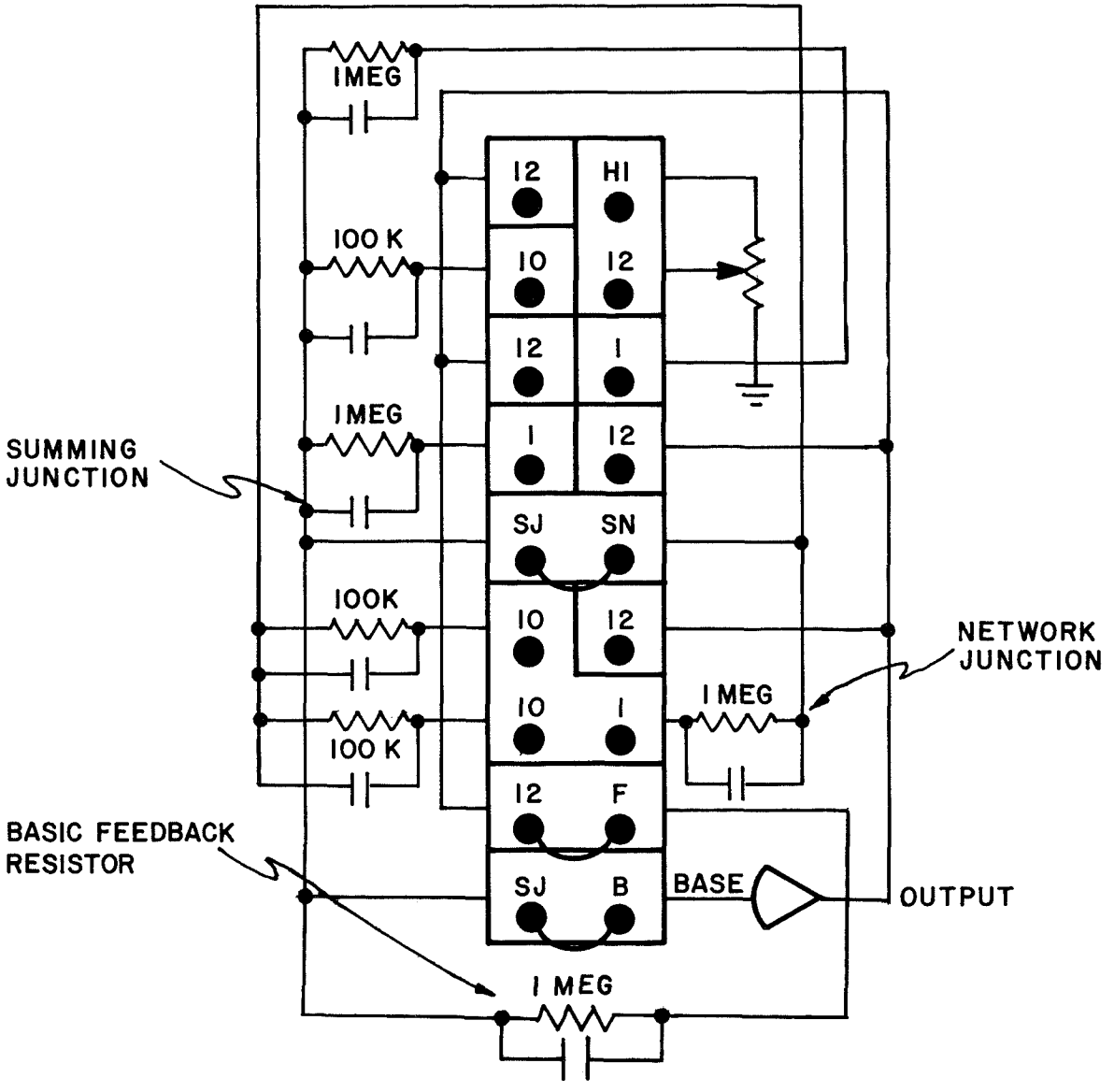
Additional input resistors are obtained by patching SJ to SN. This adds two gain 10 inputs and a gain 1 input.

A 100K feedback resistor is obtained by removing the output 12 to F patch and patching the output 12 to a gain 10 input. The SJ to B patch remains as before. In this configuration, there are two unity gain inputs (gain 10 inputs) and four 10th gain input (gain 1 inputs). The F hole is one of the 10th gain inputs.

With the 100K feedback resistor, the summer/inverter may be used as an output amplifier for function generators. To accomplish this, remove the SJ to SN patch and connect the function generator output to SJ.

All resistors can be borrowed to be used with another amplifier, if they are not used with the amplifier they are associated with. As in the case of a log/antilog configuration, the feedback resistor can also be borrowed since it is not needed in this configuration.

SUMMER/INVERTER PATCHING



PATCHING SHOWN IS FOR BASIC SUMMER INVERTER USE. 1 MEG OHM FEEDBACK RESISTOR AND BORROWABLE NETWORK ARE USED.

FIGURE 2-2

2.1.3 Summer/Integrator

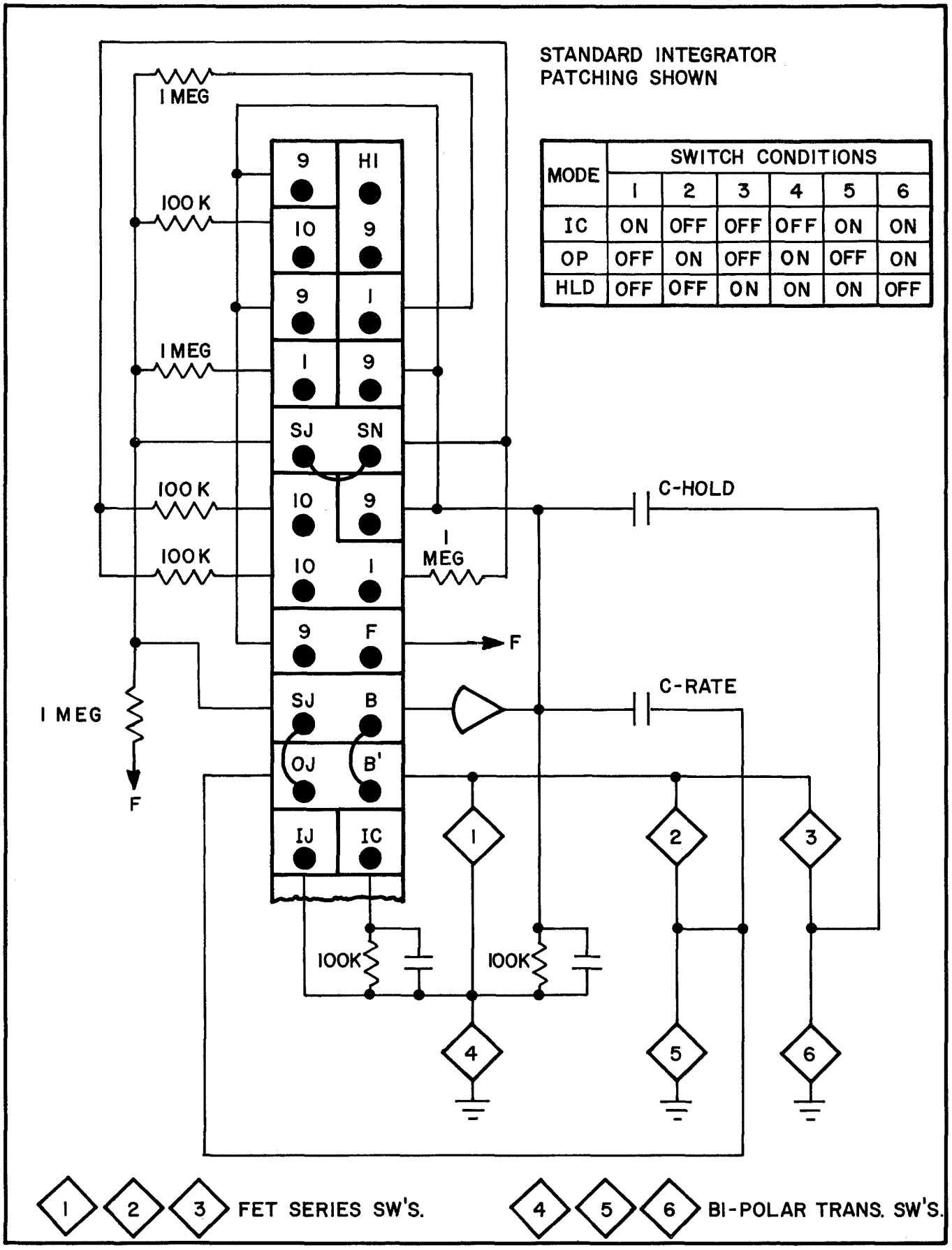
Figure 2-3 shows the standard configuration for patching an integrator. Patching the SJ to OJ and B to B' connects the summer/inverter amplifier with its input resistor network to the rate and hold capacitors, and the electronic mode control switches for operation as an integrator. As in the summer/inverter, the SJ to SN connection adds two 100K and one 1 megohm input resistors.

The diamond shaped units represent the electronic mode control switches. Units 1, 2 and 3 are series FET switches, while units 4, 5 and 6 are bipolar transistor switches. The table shows the condition of each switch in each mode. The terms ON and OFF, in reference to the mode control switches, mean closed and open.

The mode control switches are operated by logic levels applied to them at the logic patch panel. The rate and hold capacitors are selected by patching at the logic panel also. Refer to the logic patching section (Figure 2-24) of this manual for details.

The summer/integrator may also be used as a conventional summer/inverter by removing the SJ to OJ and B to B' connection and following the patching instructions given earlier on the summer/inverter.

SUMMER/INTEGRATOR PATCHING



MODE	SWITCH CONDITIONS					
	1	2	3	4	5	6
IC	ON	OFF	OFF	OFF	ON	ON
OP	OFF	ON	OFF	ON	OFF	ON
HLD	OFF	OFF	ON	ON	ON	OFF

FIGURE 2-3

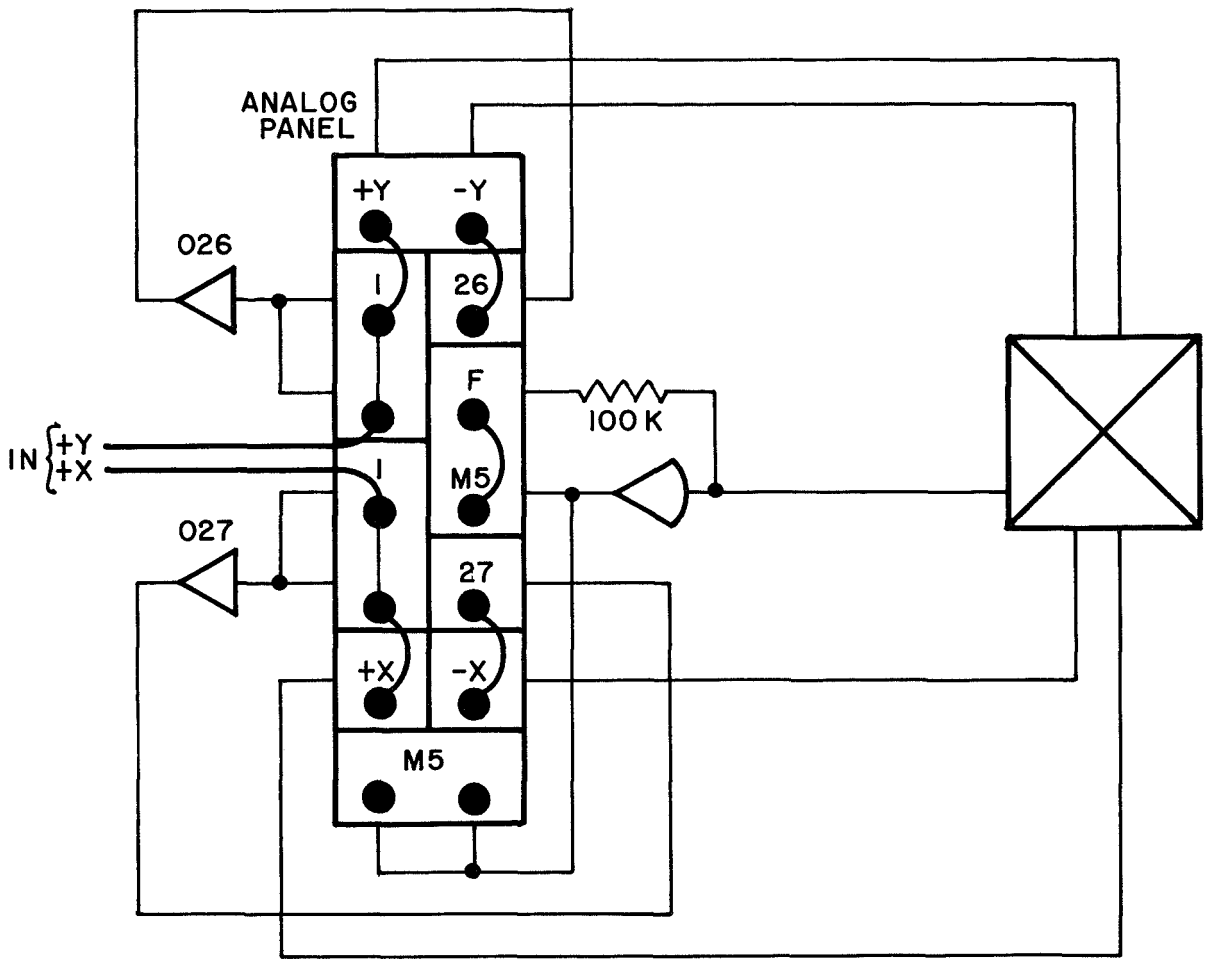
2.1.4 Multiplier

The quarter square multiplier is patched for four quandrant multiplication as shown in Figure 2-4. The inputs +X and +Y are supplied to the squaring circuitry and to inverters 26 and 27 to produce -X and -Y. The output amplifier is furnished with a 100K feedback resistor by patching F to M5. The input voltage range is ± 100 volts for either X or Y, and the output is -XY/100.

For squaring operations, the multiplier patching arrangement as shown is used, but the +X and +Y input lines are tied together. The output is $-X^2/100$.

When the multiplier is not being used, the F to M5 patch should still be used to provide a feedback resistor for the output amplifier. All other patching may be removed. In this configuration, amplifiers 026 and 027 may be used as free unity gain inverters (100K/100K).

MULTIPLIER PATCHING



OUTPUT = $\frac{-XY}{100}$

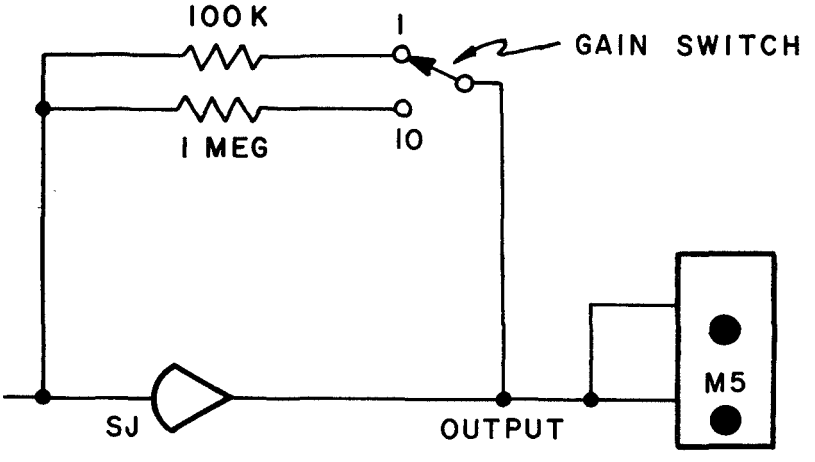
NOTE: FOR SQUARING OPERATIONS (X^2), SIMPLY CONNECT X AND Y INPUTS TOGETHER.

FIGURE 2-4

2.1.5 Multiplier Output Amplifier Gain

In some SS-10 configurations as in Figure 2-5, the multiplier output amplifiers are provided with a manually selectable gain. Feedback resistors of 100K (standard) and 1 megohm are supplied to produce gains of 1 and 10. Switches in the multiplier module are used to select the desired value. The multiplier output expressions are -XY/100 with a gain of 1 (100K) and -XY/10 for a gain of 10 (1 megohm).

MULTIPLIER OUTPUT AMPLIFIER (GAIN CONTROL OPTION) PATCHING



$$\text{OUTPUT} = \frac{-XY}{100} \qquad (\text{GAIN} = 1)$$

$$\text{OUTPUT} = \frac{-XY}{10} \qquad (\text{GAIN} = 10)$$

FIGURE 2-5

2.1.6 Divider

The quarter square multiplier may be patched to produce division as shown in Figure 2-6. In this configuration, the squaring circuitry is connected in the feedback patch of the operational amplifier. To use the divider, the following rules apply:

a. The denominator Y must be positive.

b. The denominator Y must remain greater than the numerator X at all times.

c. The denominator must not go to zero.

d. The output is equal to -100 X/Y when patched as shown.

To provide stability in this mode, it is usually necessary to provide a small amount of capacitive feedback for the output amplifier. This is done by patching the appropriate hole on the logic panel to logic true (LT). This pulls in a relay that connects the capacitor to the amplifier.

DIVIDER PATCHING

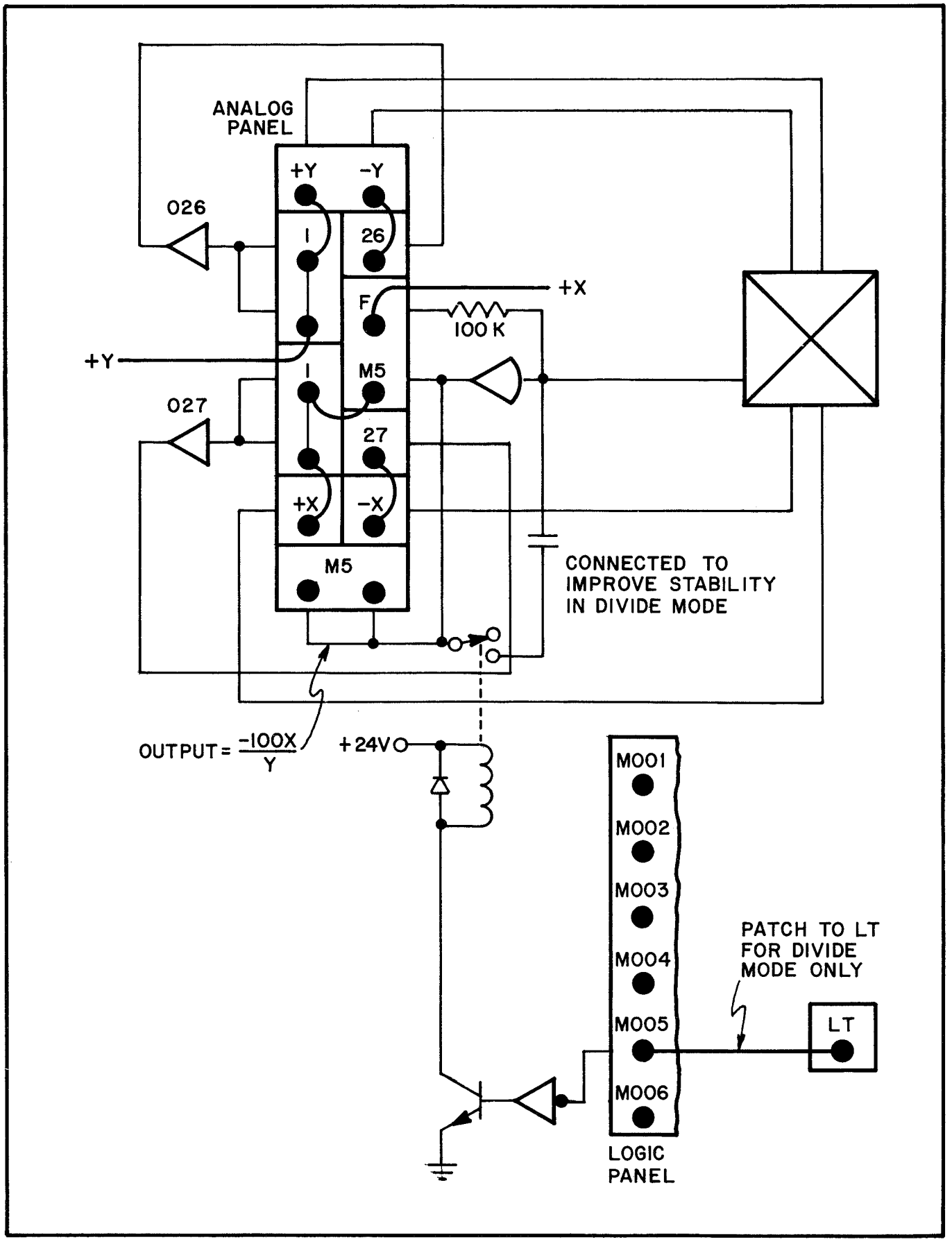


FIGURE 2-6

2.1.7 Square Root

The square root of a signal is found with the multiplier circuit patched as shown in Figure 2-7. One part of the squaring circuitry is connected so that it is the feedback element for the output amplifier. The input X is applied to the 100K resistor at F and may be either positive or negative. The output is equal to -10 multiplied by the square root of the absolute value of X.

The input amplifier 026 and 027 may be eliminated from the patching by connecting the M5 output directly to both +X and +Y. Remove the 026 to -Y and 027 to -X connections.

In this mode, a small feedback capacitor may be needed on the output amplifier for stability. Patching the appropriate hole on the logic panel to logic true (LT) will connect this capacitor via a relay as shown.

SQUARE ROOT PATCHING

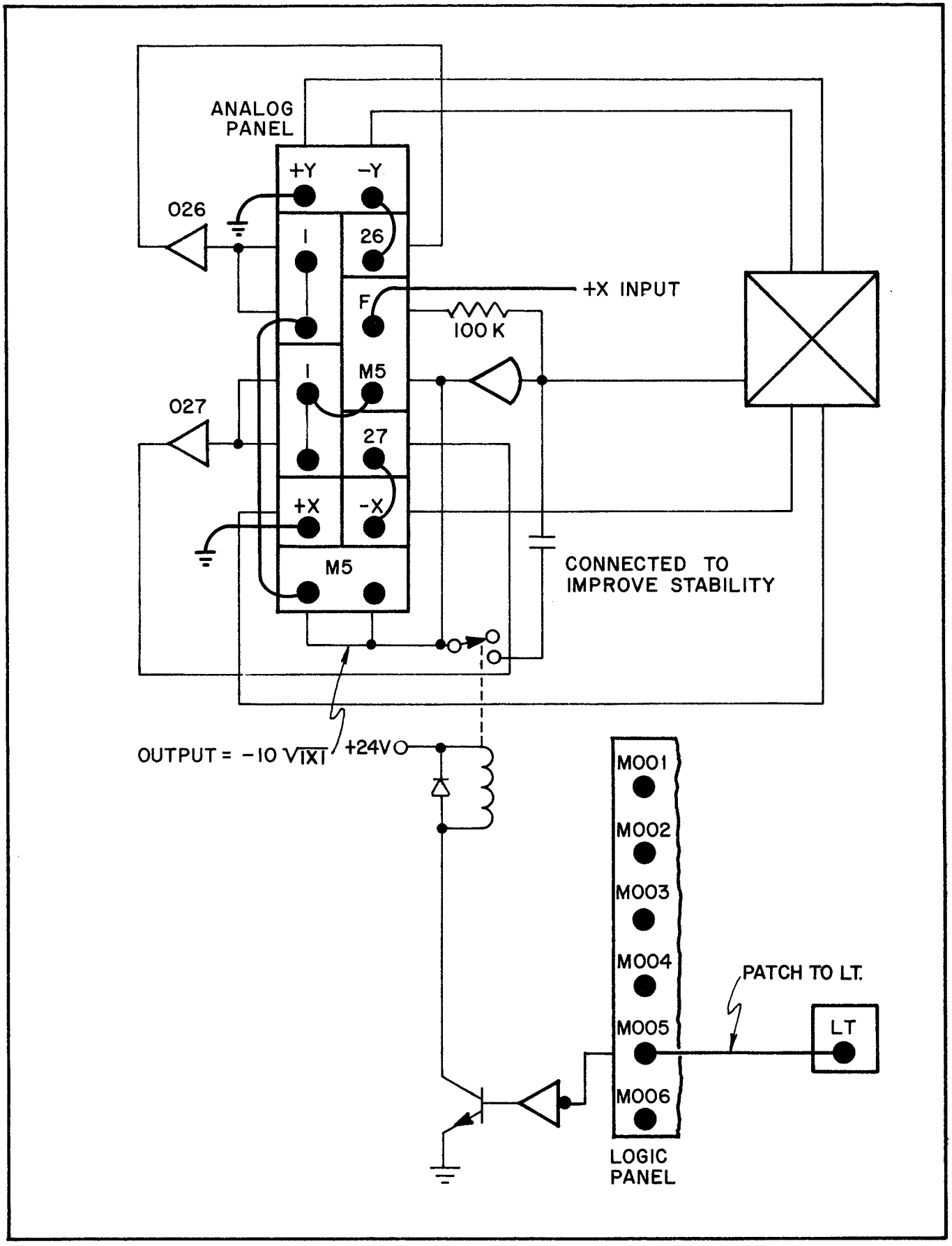


FIGURE 2-7

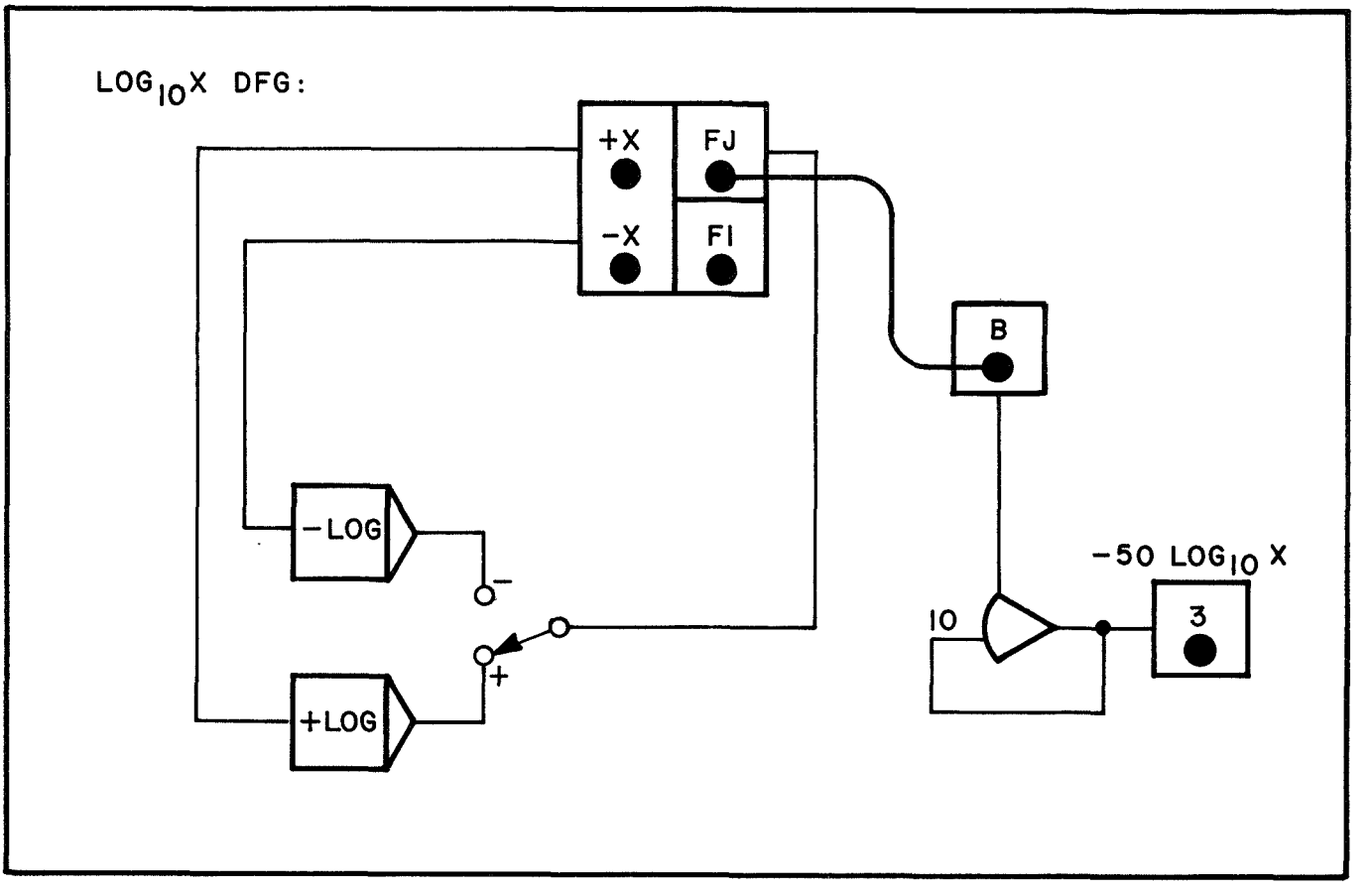
2.1.8 Fixed Diode Function Generator

The FDFG's produce a current output that varies in accordance with the mathematical functions: X^2 or log X. The log X DFG's generate the logarithm to the base 10 of an input X. Two log X function generators are provided, one for negative inputs and the other for positive inputs as shown in Figure 2-8. The inputs are labeled +X and -X on the analog patch panel. The outputs are selected with a SPDT switch, and the switch output appears at the FJ hole on the panel. The switches are located within the log card module beneath the SS-100 mainframe writing shelf. Since this is a current output, it should be patched to the base of an operational amplifier that is provided with a 100K feedback resistor. Any summer/inverter patched with a 100K feedback resistor can be used. The SJ to B patch on the amplifier should also be made. A convenient place to connect the function generator output, FJ to the amplifier, is via the SJ hole adjacent to SN (IJ of an integrator during initial condition). The output of the amplifier with the DFG input is $-50 \log_{10} X$.

The X^2 function generators are connected in a similar manner. Again both positive and negative input units are provided with inputs +X and -X at the patch panel. As indicated in Figure 2-8, their outputs are connected together at the FJ output hole. As in the log X connection, the FJ output connects to the base of an operational amplifier with a 100K feedback resistor. The output from this combination is $-X^2/100$.

The F1 and F5 holes below the FJ output holes on the DFG's are not used. By connecting the FDFG's in the feedback path of the operational amplifier, the implicit functions--antilog (X/50) with log X cards and $-10\sqrt{X}$ with X^2 cards can be performed. To patch these configurations, connect FJ of the DFG to the SJ of the operational amplifier. Patch the amplifier output to either the +X or -X input of the DFG depending upon the desired polarity. Do not patch a 100K feedback resistor on the amplifier. Patch only the B to SJ connection. The input will be a gain 10 input on the operational amplifier.

FIXED DIODE FUNCTION GENERATOR PATCHING



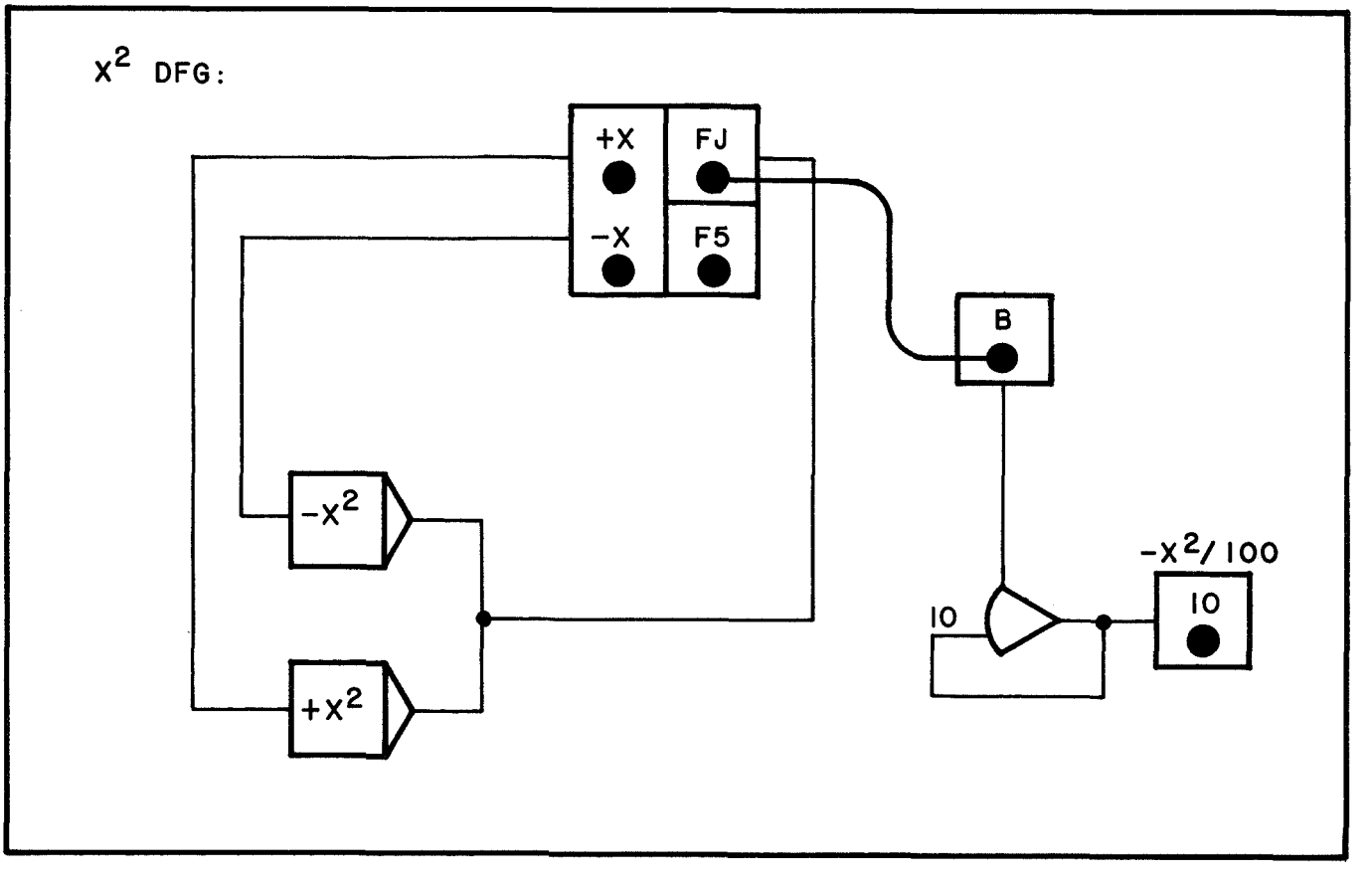


FIGURE 2-8

2.1.9 Log X Diode Function Generator

In some SS-100 configurations, the log X DFG patch panel connections are as shown in Figure 2-9. The +X output appears at FJ and the -X output at F1. No polarity switches are used. All other patching instructions are the same. (This is also true for some X^2 DFG patch panel connections.)

LOG X DIODE FUNCTION GENERATOR PATCHING

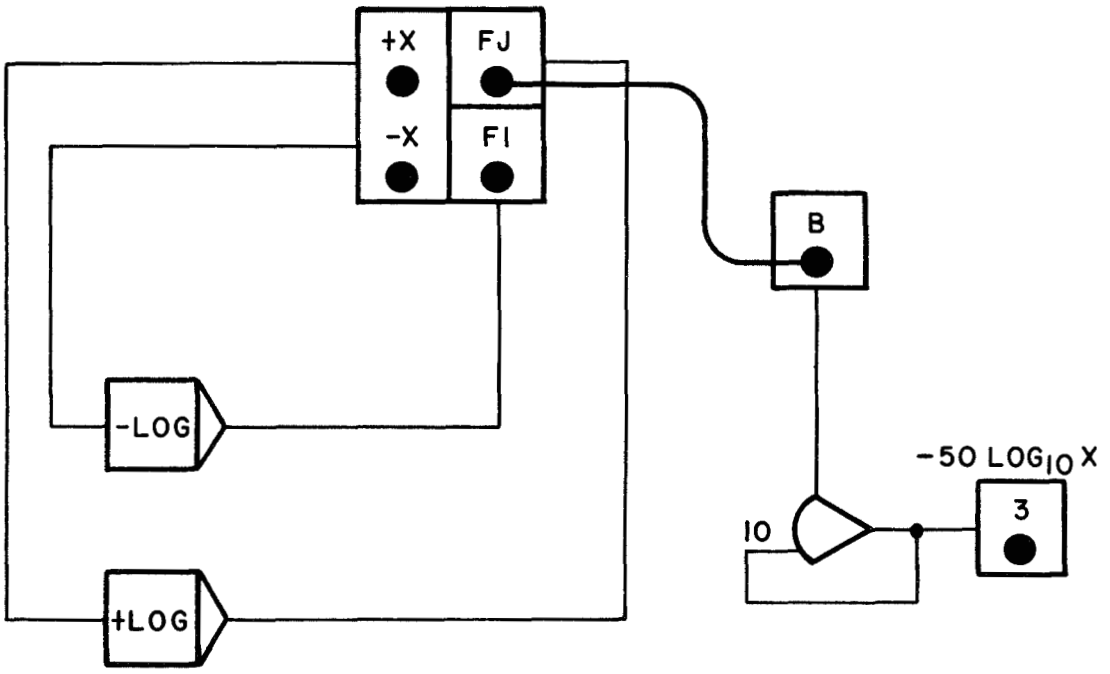


FIGURE 2-9

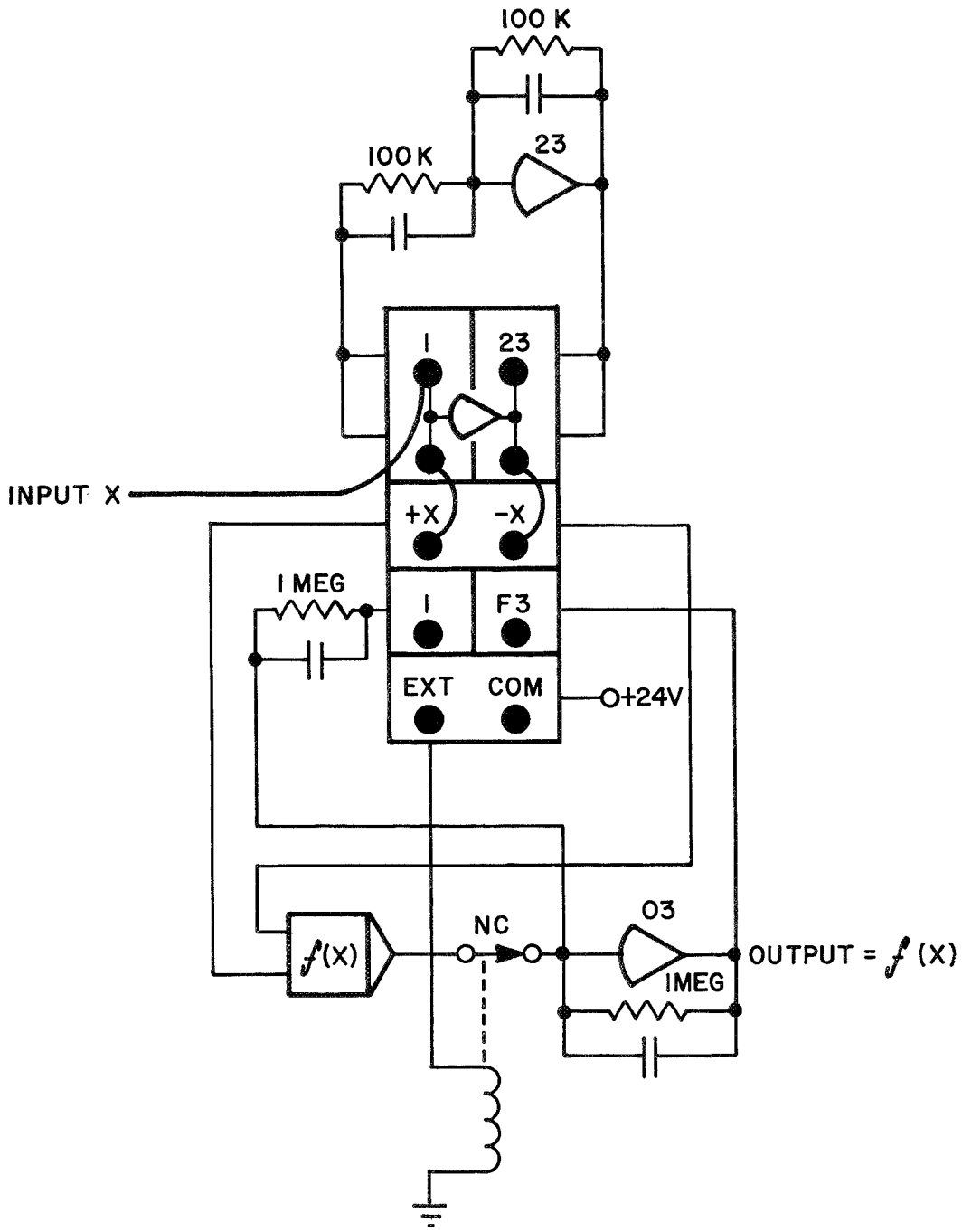
2.1.10 Manual Diode Function Generator

The MDFG is patched as shown in Figure 2-10. The input X is applied to inverter 023 to generate -X. Both +X and -X are applied to the function generator. The output of the MDFG is applied to the base of the output amplifier 003.

When the MDFG is not used, the input amplifier 023 may be used freely as an inverter by removing the connections to +X and -X. The output amplifier can also be used as a free 1 meg/1 meg inverter by patching EXT to COM. This pulls in the normally closed relay opening the contacts and breaking the connection between the function generator output and the amplifier base.

There are two MDFG's and associated amplifiers located together, one above the other, on the patch panel. Patching EXT to COM frees both output amplifiers simultaneously.

MANUAL DIODE FUNCTION GENERATOR PATCHING



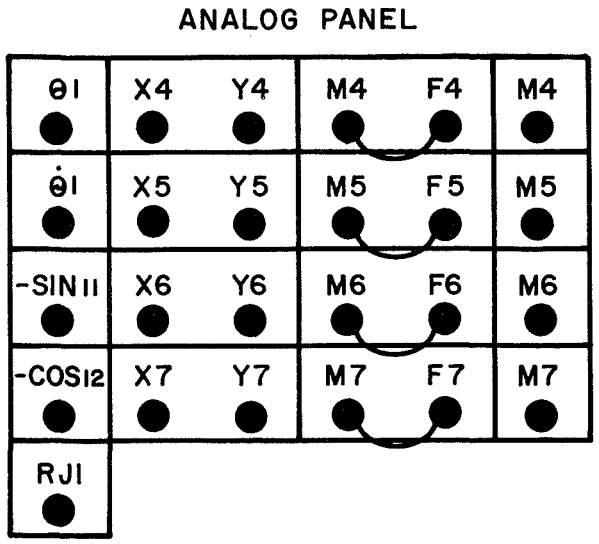
NOTE: PATCHING EXT TO COM FREES AMPLIFIER 3 FOR USE AS A 1 MEG/1 MEG INVERTER AND AMPLIFIER 2.

FIGURE 2-10

2.1.11 Resolver

The basic resolver patching on the analog panel is as shown in Figure 2-11. These M to F patches connect 100K feedback resistors to the multiplier output amplifiers. The other holes are the inputs and outputs to the various resolver components (multipliers, sine-cosine generators, etc.). The logic patching determines the resolver mode. In all resolver modes, the sine and cosine values are available for use at the patch panel. Figure 2-11 is a typical resolver in this case 401.

RESOLVER PATCHING



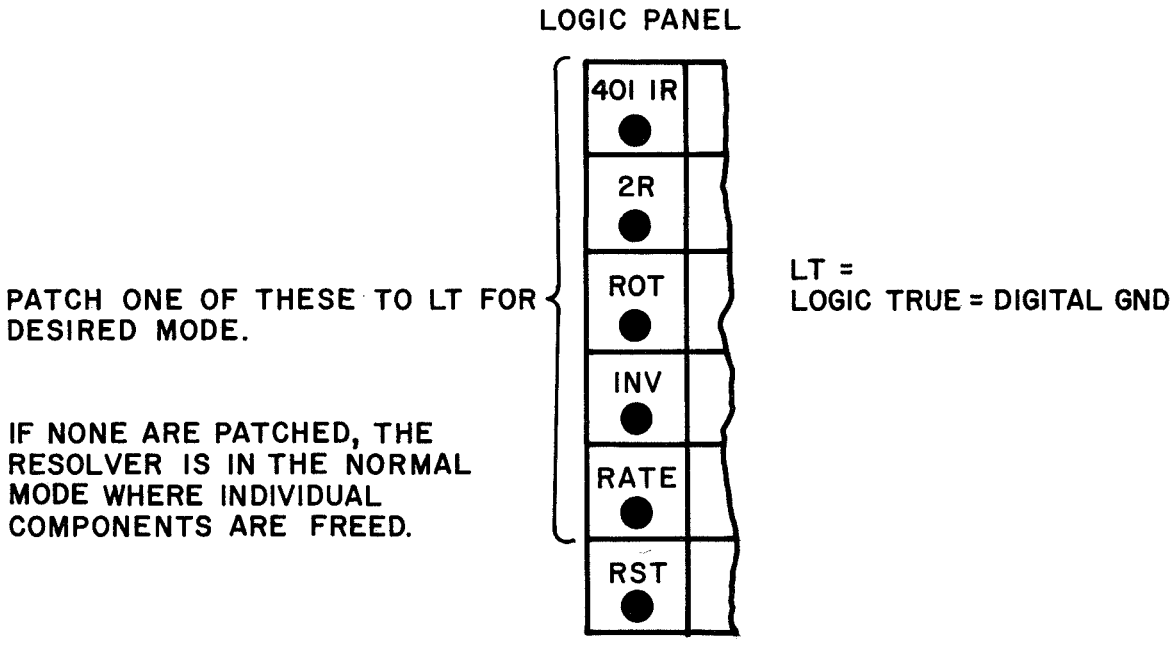


FIGURE 2-11

2.1.11.1 Resolver-Normal Mode

With no logic patching, all of the individual components can be used separately. No resolver functions will be performed. This provides the programmer with one sine-cosine generator and four quarter square multipliers that can be used freely.

The quarter square multipliers produce the output -XY/100. By connecting the X and Y inputs together as shown in Figure 2-12, the squaring function can be performed. The output is $-X^2/100$ with this connection. These multipliers can also be used for division by removing the M to F patch and patching M to Y. The X input will be denominator, while the F input will receive the numerator Y. The output at M will be -100Y/X. For this connection, X should always be positive, greater than zero and greater than Y. For square root operations, patch both X and Y to M with an off the board multiple and apply the input to F. The output will be $-10\sqrt{|X|}$.

RESOLVER - NORMAL MODE PATCHING

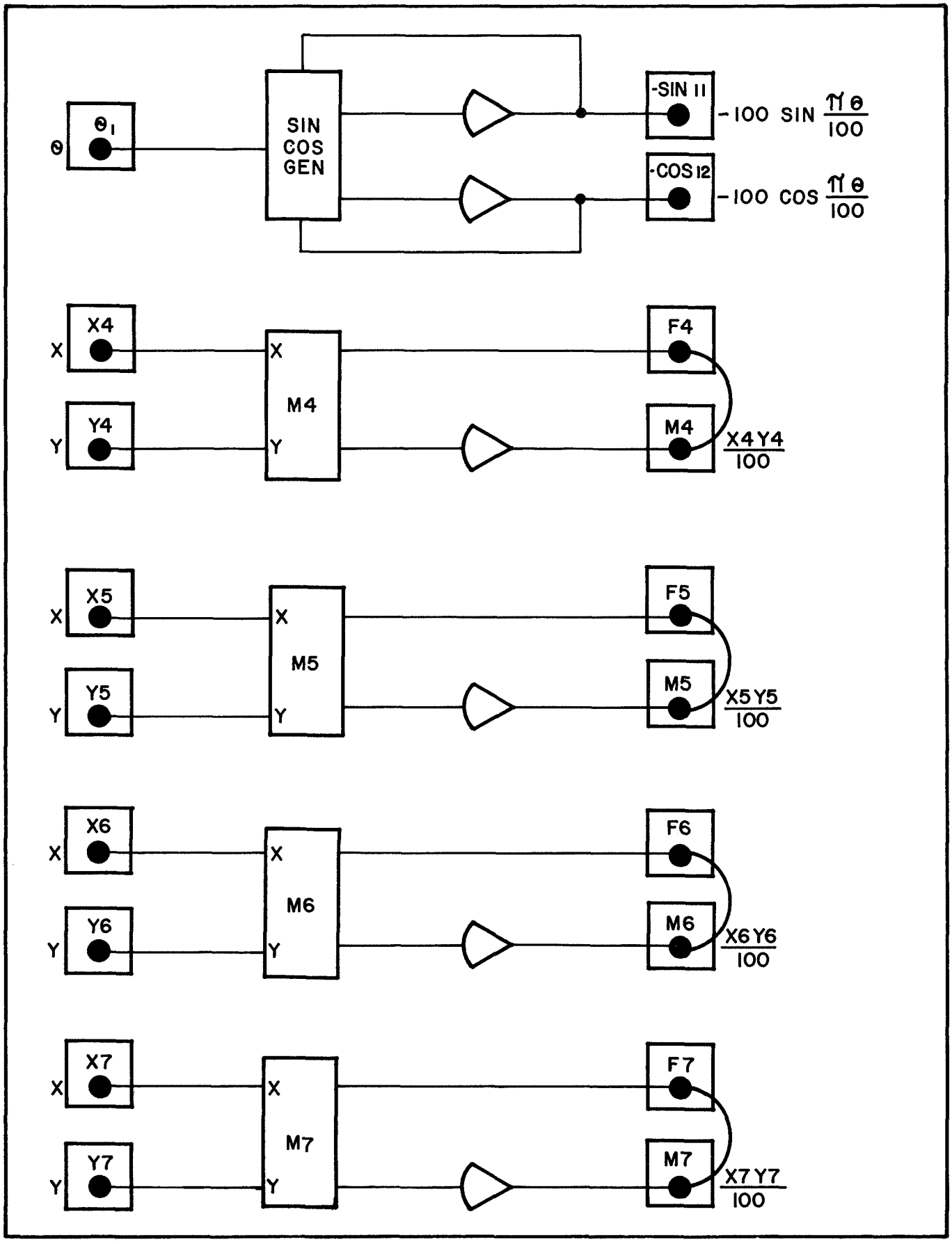


FIGURE 2-12

2.1.11.2 Forward Resolution (1R)

In this mode the resolver performs polar to rectangular coordinate transformation. The voltage representing the angle is patched to θ, and the voltage representing the radius r_o is connected to Y4 as shown in Figure 2-13. The X and Y outputs appear at M4 and M5, respectively.

To operate the resolver in this mode, the following patching is required.

Analog: a. M to F patches to provide multiplier output amplifiers with feedback resistors.

b. Inputs θ and r_o, outputs X and Y.

Logic: Patch 1R to logic true (LT).

Patching 1R to LT on the logic panel actuates relays in the resolver module that interconnect the components in the proper configuration as shown in Figure 2-13. These are strictly internal connections made automatically and no patching is necessary.

In this mode multipliers M6 and M7 are not used so they are free to be used as required.

FORWARD RESOLUTION (1R) PATCHING

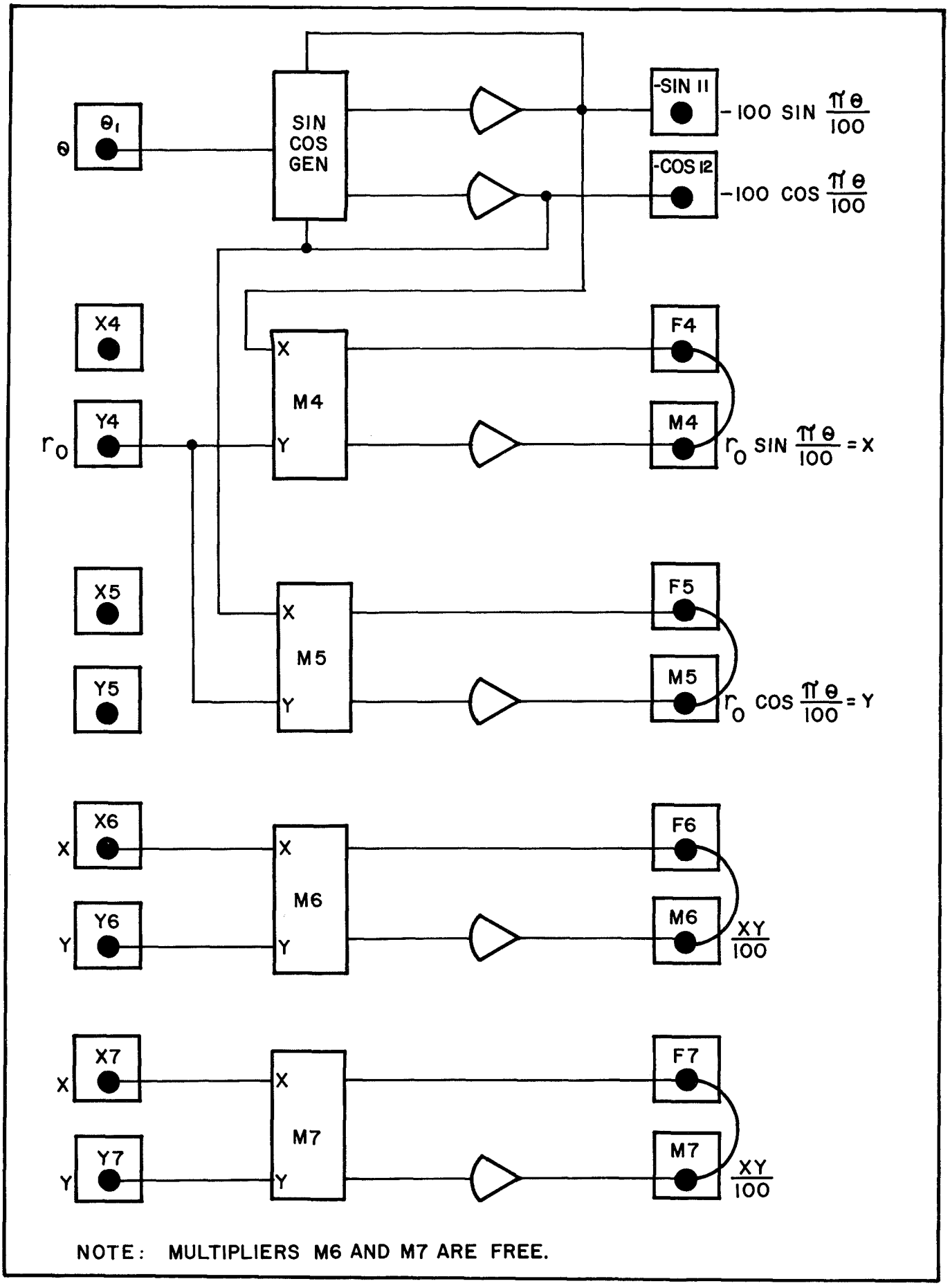


FIGURE 2-13

2.1.11.3 Double Forward Resolution (2R)

This mode provides polar to rectangular coordinate conversion for two input radii, r_0 and r_1, at the same angle Θ. The outputs are X, Y, and X_1, Y_1; they appear at M4, M5, M6 and M7.

The patching required for this mode is as follows:

Analog: a. Standard patching as shown in Figure 2-14.

b. Inputs Θ, r_0, r_1, and outputs X, Y, X_1 and Y_1.

Logic: Patch 2R to LT.

The logic patching operates relays that internally connect the resolver components as shown in Figure 2-11.

DOUBLE FORWARD RESOLUTION (2R) PATCHING

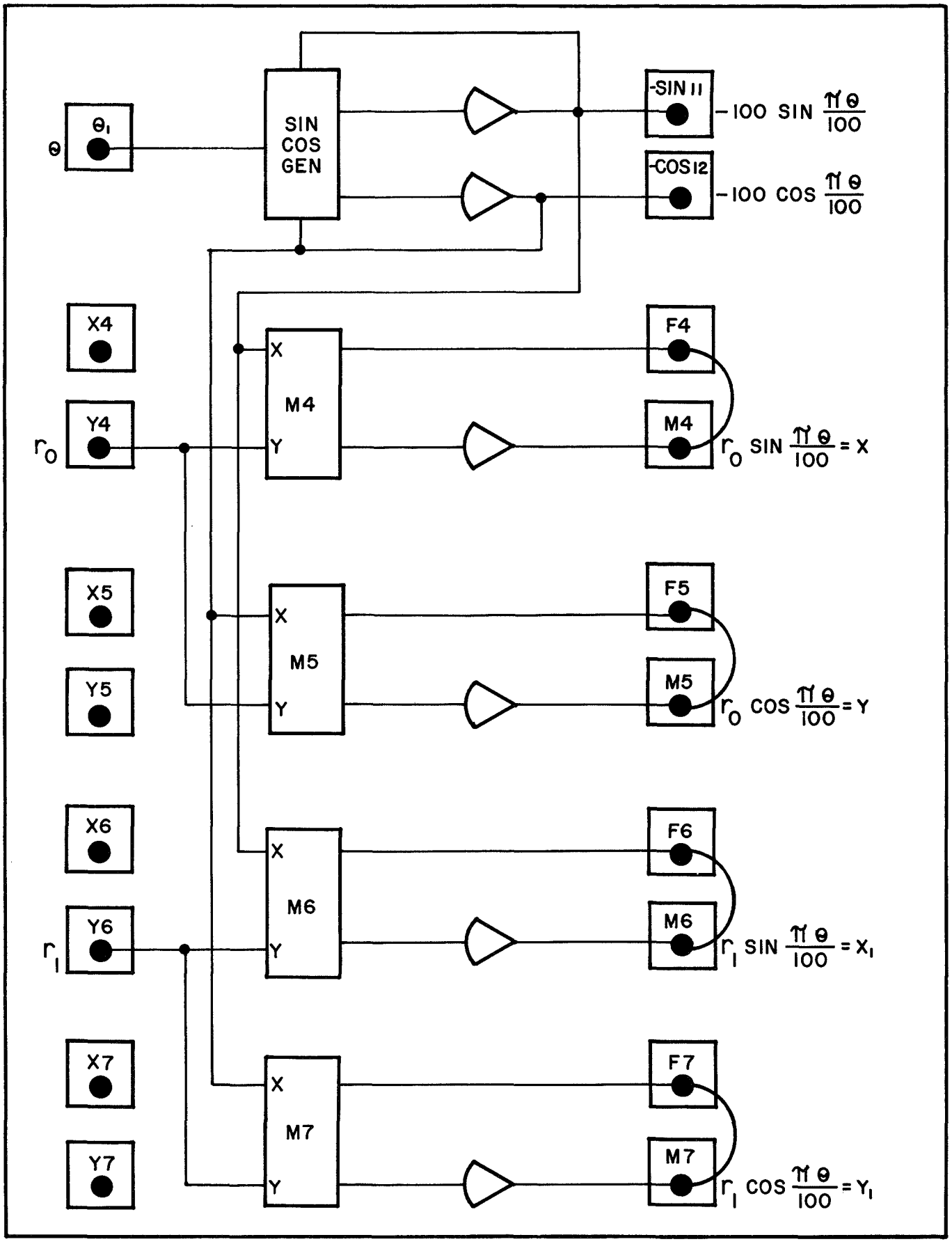


FIGURE 2-14

2.1.11.4 Inverse Resolution (INV)

This resolver mode provides rectangular to polar conversion. The inputs are voltages representing X and Y, and the outputs are voltages proportional to r and Θ.

Patching required:

Analog: a. Standard resolver patching as shown in Figure 2-15.
b. Inputs X and Y, outputs r and θ.

Logic: INV to LT.

The component configuration is as shown in Figure 2-15. In this mode, do not patch to the θ input.

INVERSE RESOLUTION (INV) PATCHING

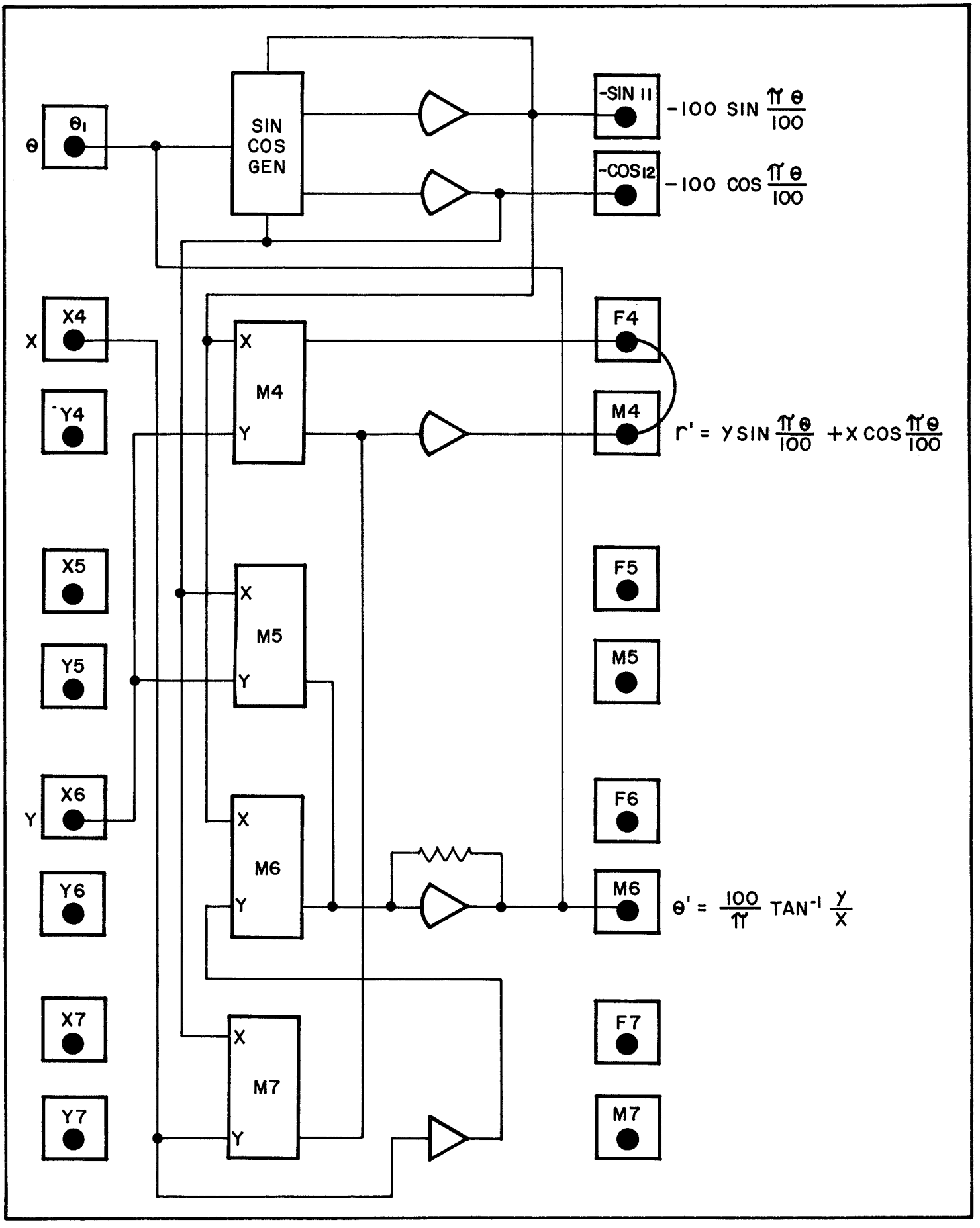


FIGURE 2-15

2.1.11.5 Rotation of Axis (ROT)

This mode of resolver operation provides for rotation of rectangular coordinate axis. Given coordinates X and Y and a desired angle of rotation Θ as inputs, the new position of the coordinates X^1 and Y^1 are generated as outputs.

The required patching is given below:

Analog: a. Standard resolver patching as shown in Figure 2-13.
b. Inputs X, Y, Θ and outputs X^1 and Y^1.

Logic: ROT to LT.

The component configuration is shown in Figure 2-16.

ROTATION OF AXIS PATCHING

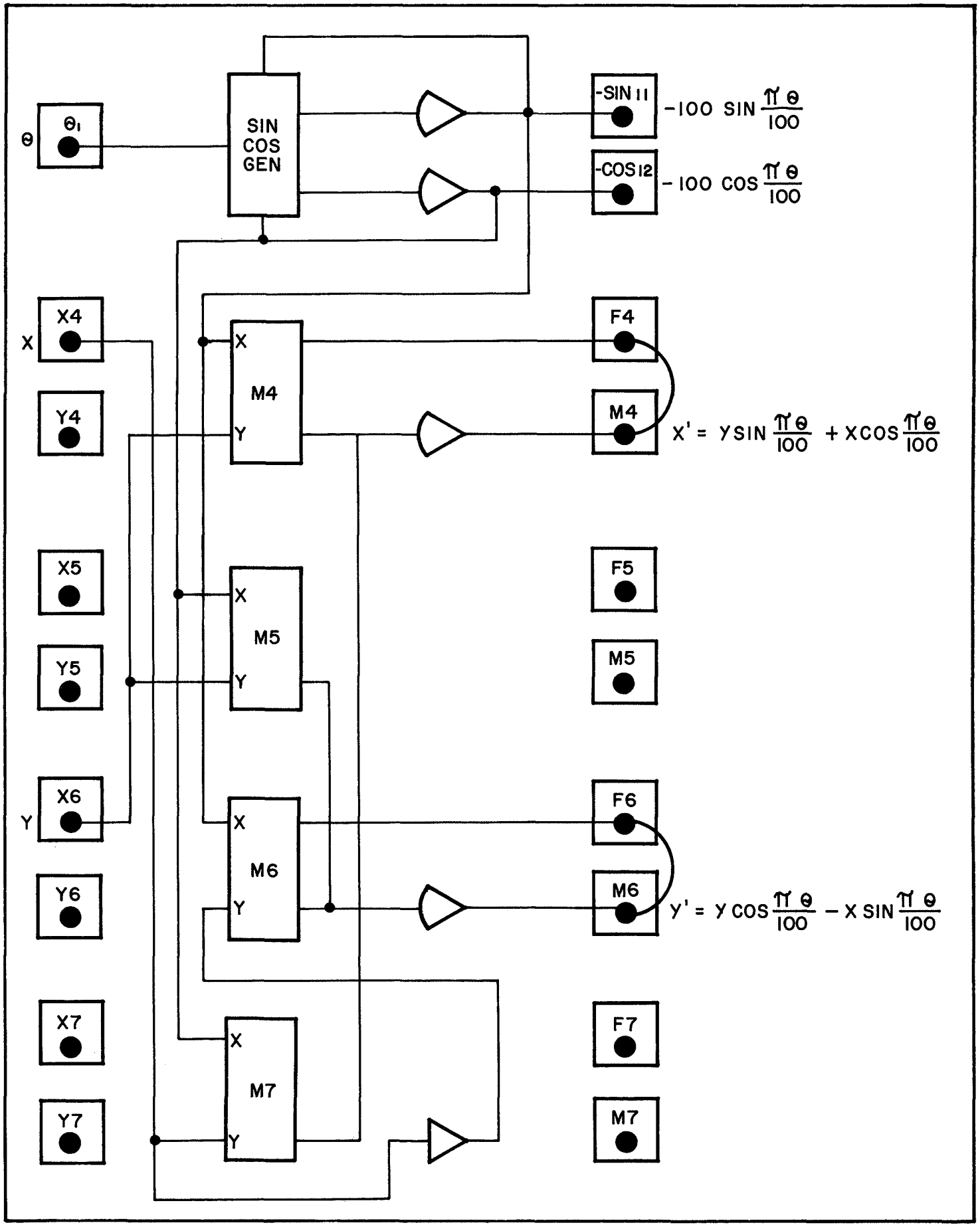


FIGURE 2-16

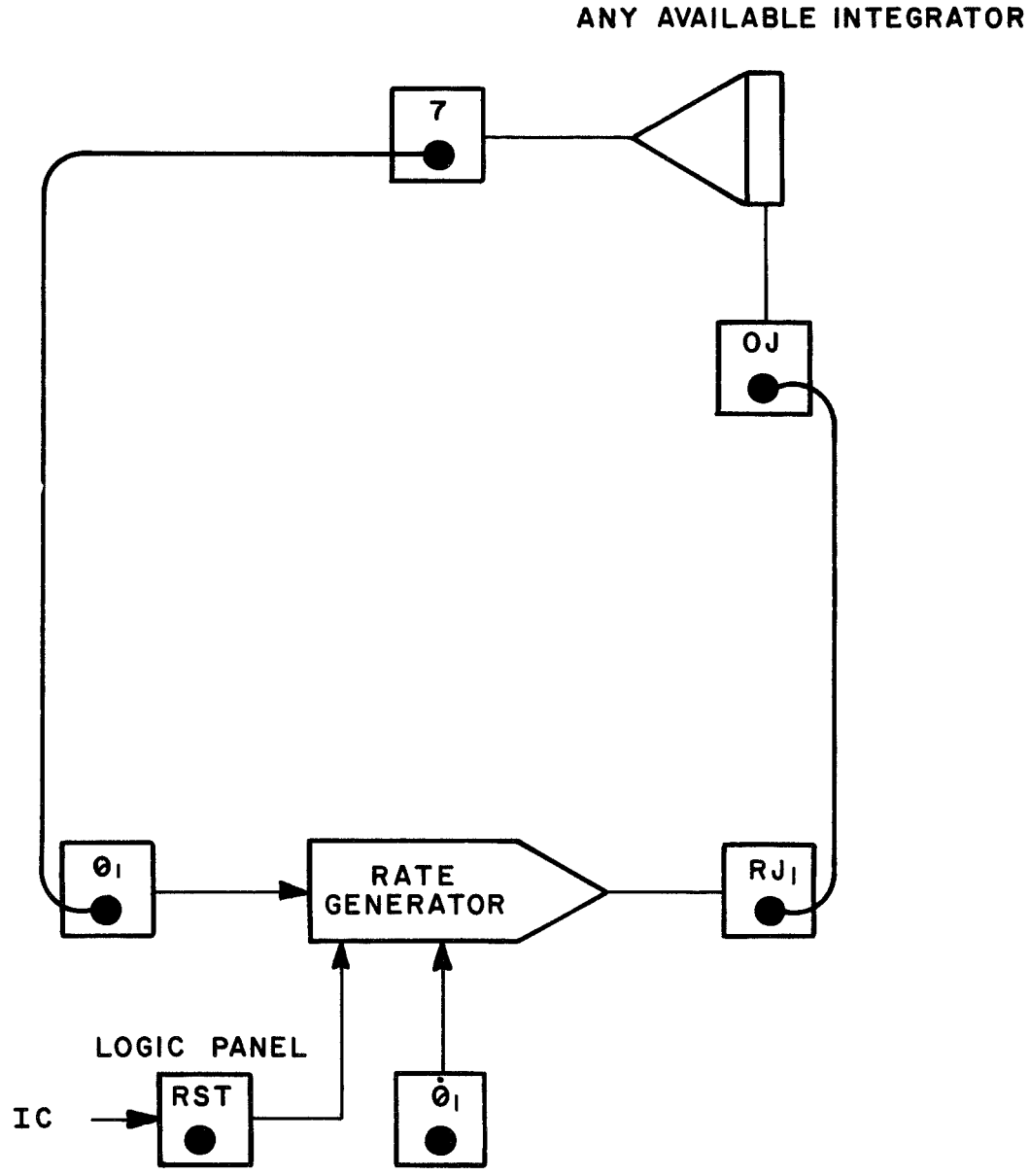
2.1.11.6 Resolver Rate Generator

The rate generator is used for continuous resolution in the 1R, 2R, and rotation of axis modes. Inverse (INV) mode operation is not used with the rate generator.

Any free integrator can be used in the rate mode. The mode control and capacitor select patching is similar to that for other integrators. The SJ to OJ connection on the analog panel is not made for the selection integrator. Instead the OJ hole is patched to the RJ hole in the resolver area. The B to B' patch is made; however, the input to the rate generator is $\dot{\Theta}$, the rate of change of theta. The integrator output, Θ, is patched back into the rate generator. The Θ output is also used to feed the sine-cosine generator for the selected modes. The configuration is shown in Figure 2-17.

On the logic panel, the desired mode is patched to LT. LT is also patched to the RATE hole to enable the rate generator. The RST (reset) hole is patched to the same logic source driving the integrator IC input.

RESOLVER RATE GENERATOR PATCHING



NOTE: CAN BE USED IN ALL RESOLVER MODES.

FIGURE 2-17

2.1.12 Free Components/Bridge Limiter

2.1.12.1 Free Components

Some SS-100's have additional 100K and 1 megohm resistors provided on the analog patch panel at various places. These can be used as extra input or feedback resistors, voltage dividers or for other required applications. These 0.005% units are the same as those used in the summing networks of the operational amplifiers. Mounted in the capacitor oven, they are very temperature stable. The resistors are wired with coax; no padding capacitors are used. Refer to Figure 2-18.

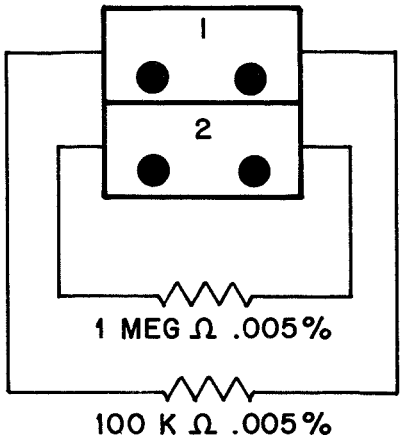
2.1.12.2 Bridge Limiter

The bridge limiter provides independent positive and negative limit levels. Both are adjustable over the 0 to 100 volt range with pots K1 and K2. These are precision 10-turn pots with built-in calibrated dial.

The limiter is simply a biased diode bridge as illustrated in Figure 2-18. It should be driven from an amplifier and it may drive any other component.

FREE COMPONENTS AND BRIDGE LIMITER PATCHING

FREE RESISTORS:



BRIDGE LIMITER:

LIMIT RANGE: ±100V
UPPER (+) AND LOWER (−) LIMITS INDEPENDENTLY ADJUSTABLE WITH K1 AND K2.

$0 < K < 1$

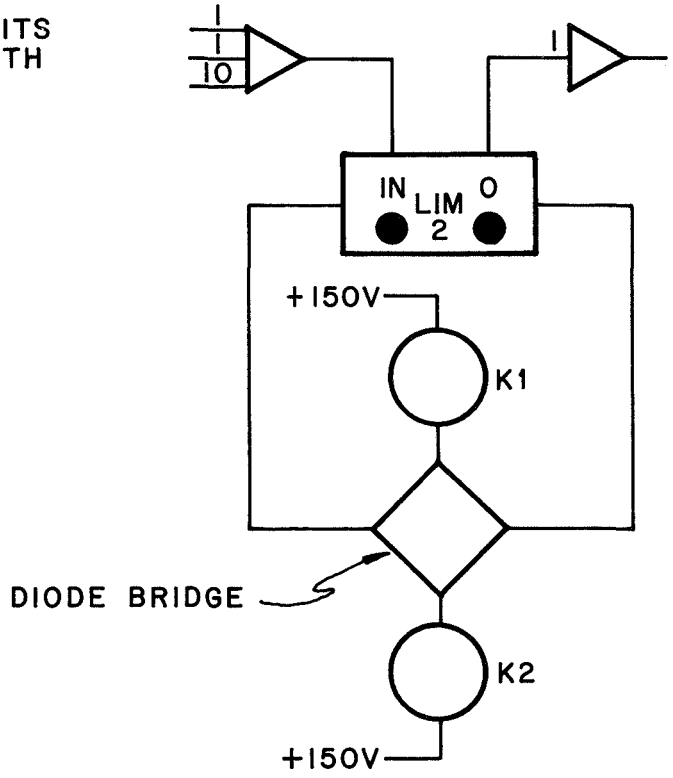


FIGURE 2-18

2.1.13 D-R Converter/Sample-Hold Amplifier

2.1.13.1 D-R Converter

The D-R (digital-to-resistance) converter receives a 13-bit plus sign 2's complement binary number from the digital computer through the interface and generates an analog voltage proportional to the digital input. The output appears at the D-R holes on the analog panel as illustrated in Figure 2-19. The D-R converter is essentially a potentiometer whose voltage division ratio is set digitally. Both the low and high sides of the resistance element are terminated at the analog panel and are marked LO and HI. The pot arm is the D-R output.

For standard digital-to-analog conversion, the HI and LO inputs are patched to +100 and -100 volt reference with bottle plugs. This gives an output voltage range of ±100 volts. For digital-to-analog multiplication, the analog signal to be multiplied by the digital value is patched into HI. The LO input is grounded. To multiply by a bipolar analog variable, patch +X to HI and -X to LO.

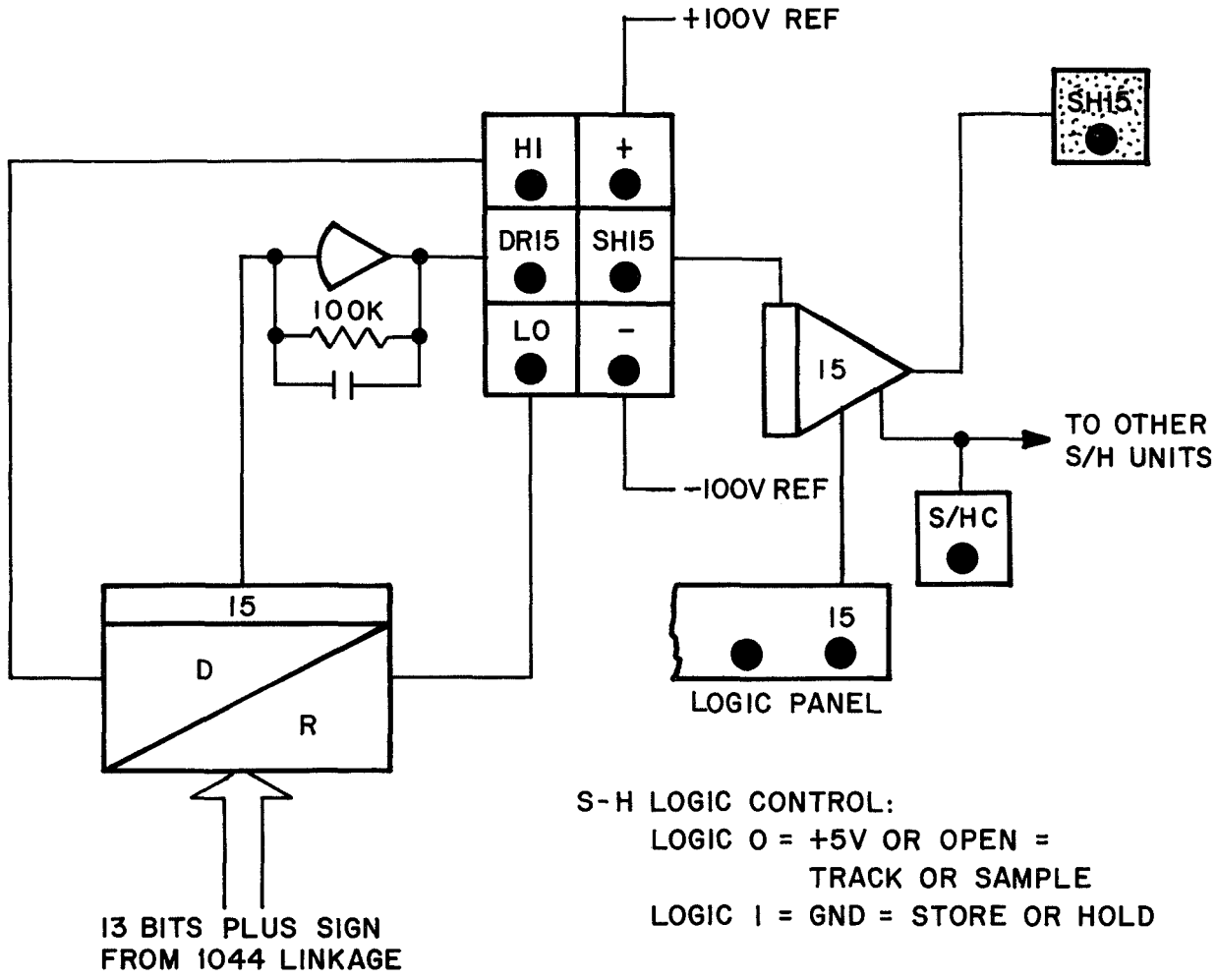
2.1.13.2 Sample-Hold Amplifier

The sample-hold (S/H) amplifier is essentially a two-mode integrator (reset and hold or track and store) with 100K/100K input/feedback resistors and a 0.01 mfd hold capacitor. High speed electronic switches place the amplifier in either mode under logic level control.

With the logic panel input hole LF (logic 0) or at +5 volts, the amplifier is in track mode, and the output is simply an inverted version of the input. Refer to Figure 2-19. However, making the logic input LT (logic 1) causes the amplifier to switch to the hold mode. The signal on the input at the time of switching from sample to hold is stored by the capacitor.

In addition to the individual S/H control inputs on the logic panel, there

D-R CONVERTER AND SAMPLE-HOLD AMPLIFIER PATCHING



S-H LOGIC CONTROL:
LOGIC 0 = +5V OR OPEN = TRACK OR SAMPLE
LOGIC 1 = GND = STORE OR HOLD

NOTES:

1. FOR D-A OPERATION, PATCH HI TO + AND LO TO −. (FOR CONTROL BY 1044)
2. FOR D-R MULTIPLIER OPERATION, PATCH LO TO GND AND ANALOG INPUT TO HI.

FIGURE 2-19

is a single master control for all of the S/H amplifiers. It is white and labeled S/HC. Refer to Figure 2-19.

Notice that the D-R output is directly adjacent to the S/H input. This permits a simple bottle plug connection to implement a loop test of interface when the S/H amplifiers are used ahead of the multiplexor channels.

2.1.14 Sample-Hold Amplifier/High Speed Integrator

This unit has a dual purpose. It can perform as either a sample-hold amplifier or a high-speed two mode integrator. For S/H application, use the IC input and S/H output as illustrated in Figure 2-20. For integrator use, one gain of 10 and gain of 1 inputs are provided as well as the IC input. A 0.001 mfd hold/rate capacitor is used.

For S/H operations the individual logic input lines are used for control as follows:

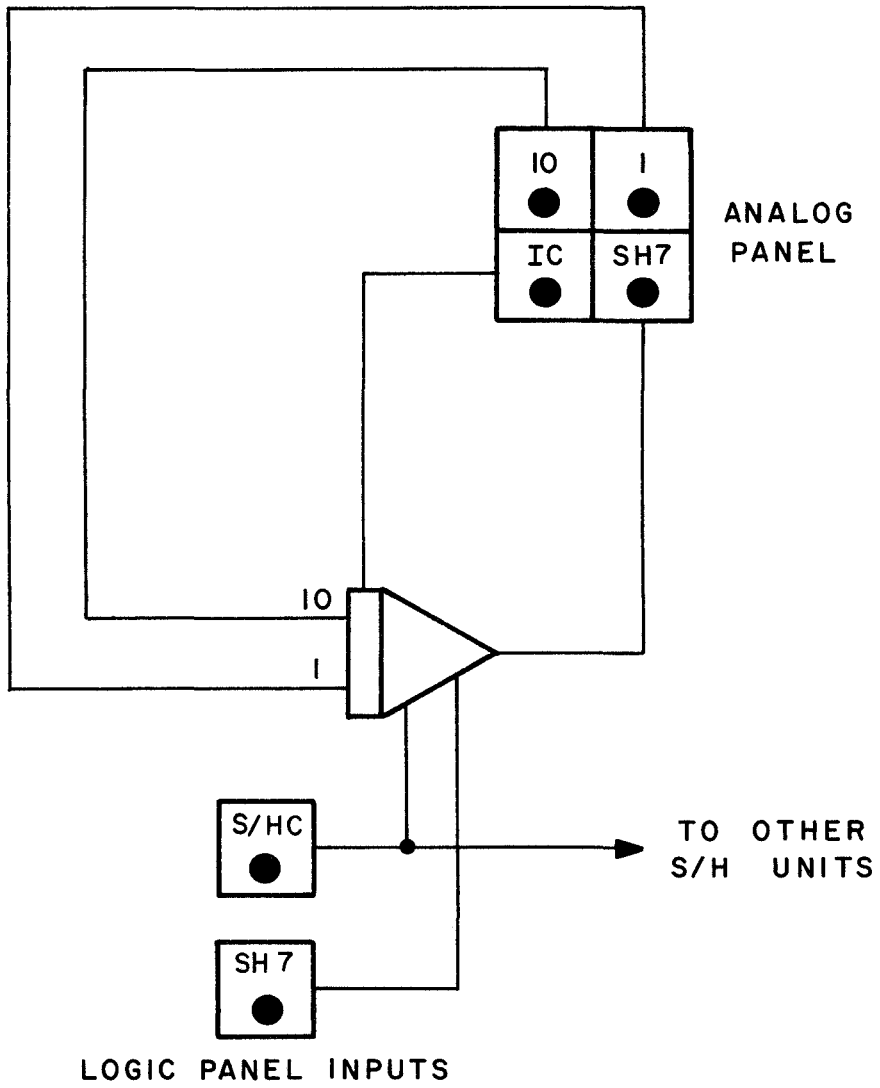
Logic 0 = +5 volts or open = sample (track)
Logic 1 = ground = hold (store)

For integrator operations the logic control inputs are:

Logic 0 = +5 volts or open = IC
Logic 1 = ground = operate

The S/HC input on the logic panel controls all units simultaneously.

SAMPLE-HOLD AMPLIFIER / HIGH SPEED INTEGRATOR PATCHING



S/HC-SAMPLE/HOLD CONTROL LINE COMMON TO ALL S/H AMPLIFIERS

SH 7-INDIVIDUAL LOGIC CONTROL LINE FOR S/H AMPLIFIER

FIGURE 2-20

2.1.15 Function Switch/Function Relay

2.1.15.1 Function Switch

This is a single pole three-position toggle switch that can be used to switch the analog signal from one place to another. When the switch is in the center position, the arm at FS2 is connected to the C hole on the patch panel. In the left or right position, the arm is connected to holes L or R. Refer to Figure 2-21. The switches are located on the center panel of the SS-100 console with the hand set pots and are wired to the patch panel with coax.

2.1.15.1.1 Desk Function Switch

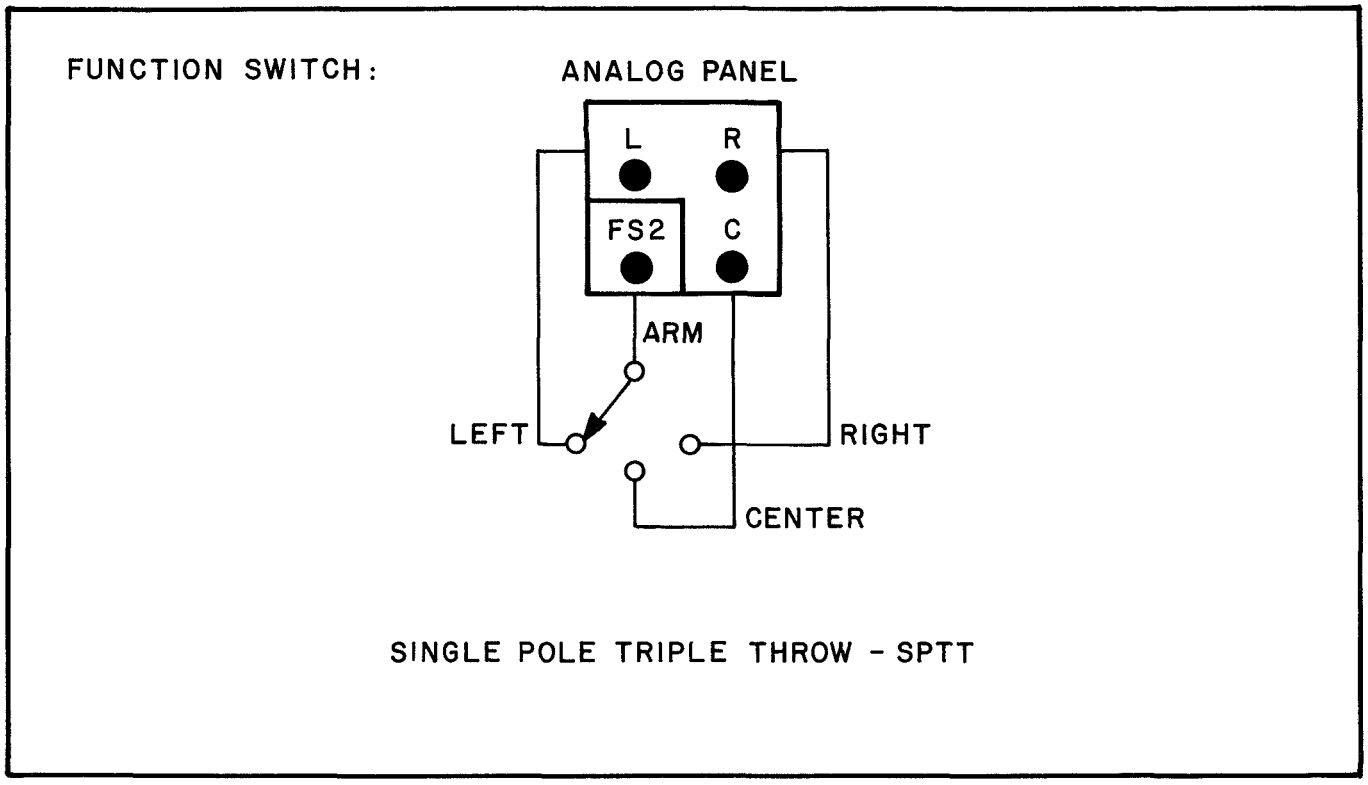
In some versions of the SS-100, the function switches are located remotely in the operator's control desk. To avoid the detrimental effects of long cables, the switches themselves control relays which actually perform the switching in the computer mainframe. The relay contacts are connected as shown in Figure 2-21 and terminate at the analog patch panel.

The relays are SPDT mercury wetted units that are connected to produce a SPTT function. Each relay is actuated by a driver circuit that is controlled by the SPTT desk switch. With the switch in its center-off position, both relays are de-energized and rest in their normally closed position connecting the arm FS2 to the C (center) hole. Placing the switch in the left position (up on this drawing) causes K1 to be energized. This connects the arm FS2 to the L (left) hole through energized K1 and still de-energized K2. With the switch in the right position (shown down here), relay K2 becomes actuated, K1 is de-energized. This produces continuity between FS2 and R (right) hole. Relay pull-in and drop out time is 1.5 milliseconds. There is no bounce.

2.1.15.2 Function Relay

The function relays are single pole double throw (form C, break before make) mercury wetted relays that are used to switch analog signal under logic control.

FUNCTION SWITCH AND FUNCTION RELAY PATCHING



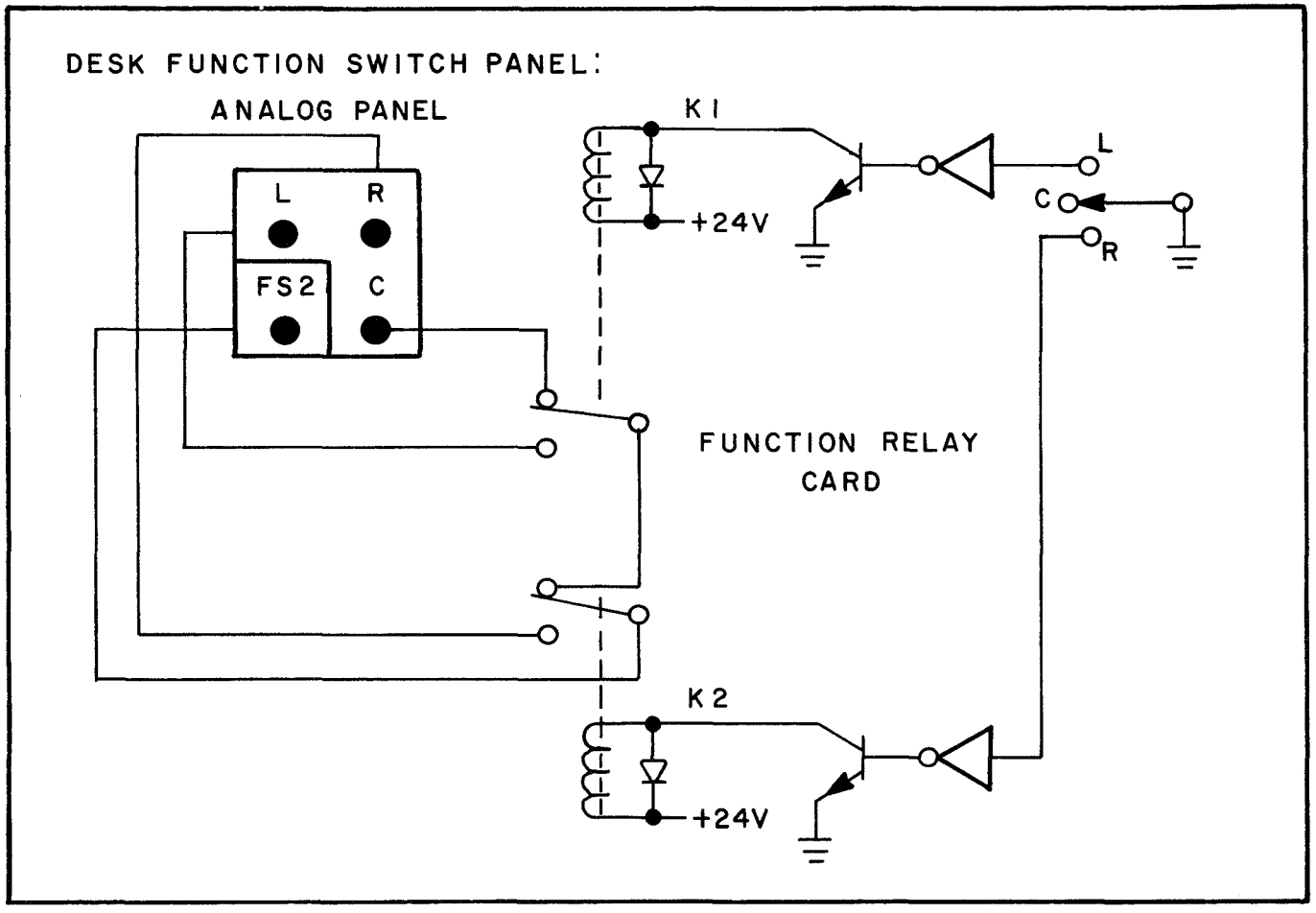


FIGURE 2-21

When the digital input at the logic panel is open or a +5 volts, the relay is at rest, and continuity exists between the relay arm FR4 and the normally closed contact (NC) as indicated in Figure 2-21. When LT is applied to the input, the relay pulls in causing the curcuit between the arm and the normally open contact (NO) to close. The arm will remain in this position as long as LT is applied to the input.

The relay switches in approximately 1.5 miliseconds thus permitting logic driving frequencies of up to about 500 Hz with reliability. The mercury wetted contacts produce no bounce.

FUNCTION SWITCH AND FUNCTION RELAY PATCHING

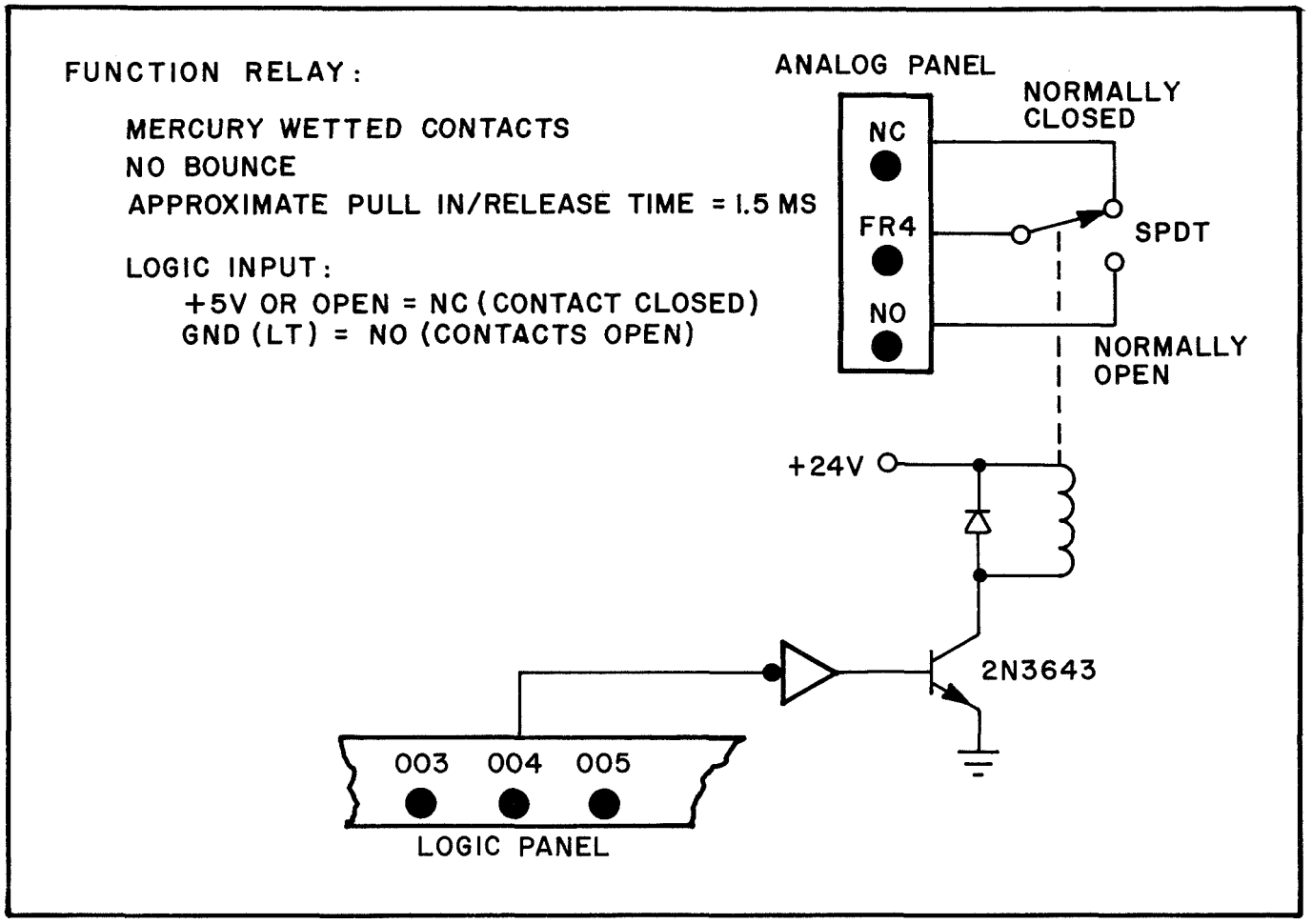


FIGURE 2-21 (CONT'D)

2.1.16 Comparator

The comparator accepts two analog input signals at holes a and b on the analog panel. It compares the two inputs and generates a logic output level that designates whether the sum of the inputs is either positive or negative. The relationships below express the operation of the circuit.

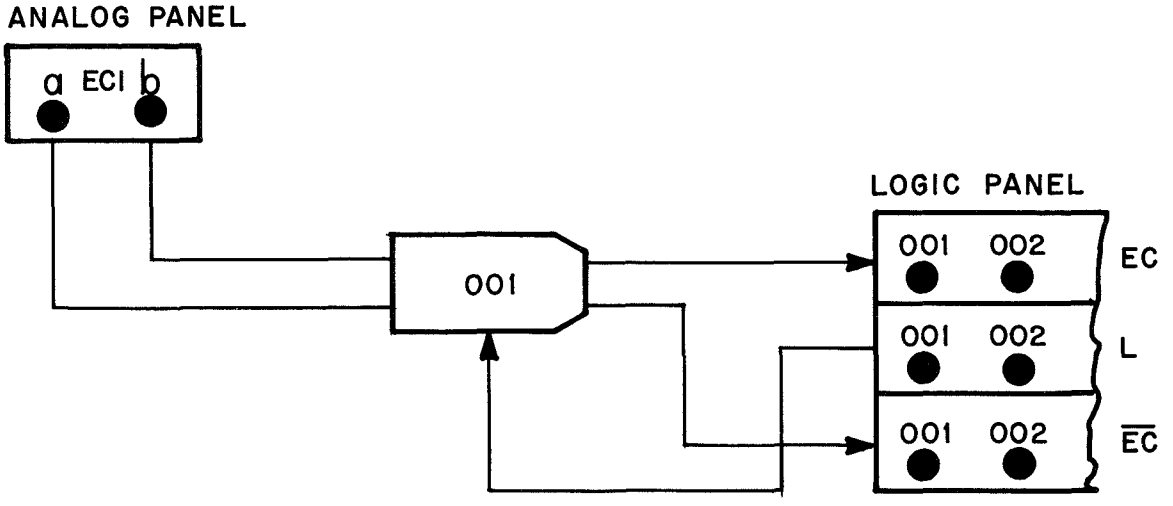
$a + b > 0$ then the logic output EC = logic 1 = ground

$a + b < 0$ then the logic output EC = logic 0 = +5 volts

The comparator sensitivity is normally set so that the comparator switches when the input signals are within 10 millivolts of one another. The switching time is less than 1 microsecond.

The latch input permits control over the output levels. With the latch (L) input open or at +5 volts, the logic output levels switch in accordance with the changes in the analog inputs. Bringing L to logic true locks the comparator outputs in their current state regardless of input changes. Releasing the latch will immediately cause the comparator output to remain as is or to switch as determined by the state of the inputs at the time of release. Refer to Figure 2-22.

COMPARATOR PATCHING



a + b > o - - - - - EC = LOGIC 1 = GND

a + b < o - - - - - EC = LOGIC 0 = +5V

L = LATCH

L = LOGIC 0 = +5V OR OPEN - COMPARATOR RESPONDS TO ANALOG INPUTS.

L = LOGIC 1 = GND - COMPARATOR OUTPUT HOLDS LAST RESULT REGARDLESS OF INPUT CHANGES.

FIGURE 2-22

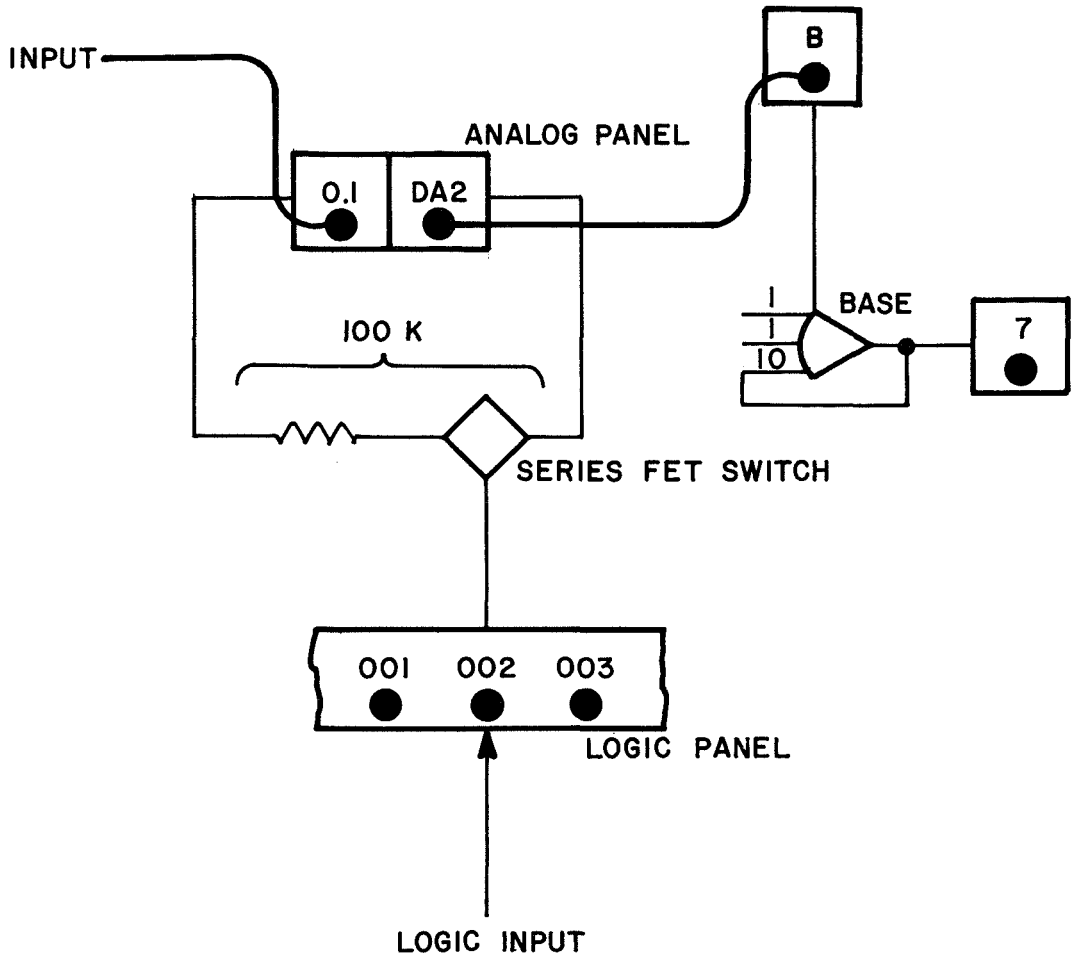
2.1.17 Digital-to-Analog Switch

The D-A switch is a field effect transistor used as a series SPST switch with a precision resistor for switching analog signals under logic control. It is designed to be connected to the base of an operational amplifier. The overall impedance is 100K ohms, therefore, the response is a gain of 10 input. Refer to Figure 2-23.

With the logic panel input open or at +5 volts, the switch is open. Patching LT to the logic input causes the switch to close. Any signal at the input will be applied to the base of the amplifier. The switching speed is less than 1 microsecond.

A single pole double throw (SPDT) switch is easily made with two D-A switches. Their outputs (D-A) are tied together at the base (B), and the two signals to be switched into the base are applied to the inputs. The logic inputs are then driven from complementary sources so that when one switch is on the other is off.

DIGITAL TO ANALOG SWITCH PATCHING



LOGIC INPUT:

LOGIC 0 (+5V OR OPEN) - SWITCH OPEN

LOGIC 1 (GND) - SWITCH CLOSED

FIGURE 2-23

2.2 LOGIC ELEMENTS

The basic function of each patchable logic element is described in the following paragraphs. However, their application is certainly not limited to only those functions described here. For example, the gates may be used to form additional set-reset flip-flops. This is done by patching the $\overline{Gn}$ outputs of two gates back into the opposite inputs of these gates. Another example is the ability to patch virtually any type of counter, register or divider with the JK flip-flops. Only the most popular and most often used counter/register configurations are provided, but any other configuration needed can be easily made. It is recommended that a good text on Boolean algebra, logic design and applications be read in order to achieve maximum usefulness from the patchable logic elements.

2.2.1 Integrator Mode and Time Scale Patching

The rate and hold capacitors for the integrators can be selected on a sector or quadrant basis or individually. The sector basis is more convenient and saves patching effort. Each group of eight integrators is provided with a set of capacitor select holes as shown in Figure 2-24. Three holes are for selecting rate capacitors, and the other three select hold capacitors. The holes are labeled with time scale factors. These holes are patched to logic true (LT) to select the capacitors as designated in the following table. (A 1 megohm input resistor is assummed.)

Capacitor	Time Scale	Patch to LT
10.0 mfd	0.1	0.1
1.0	1	0.1 & 10
0.1	10	10
0.01	100	0.1 & 10^3
0.001	1,000	10^3
0.0001	10,000	10 & 10^3

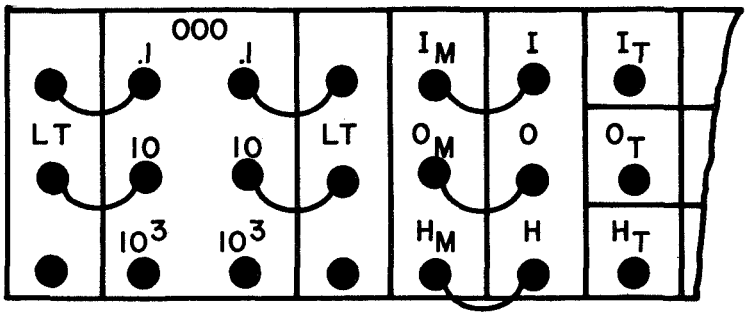
Several examples of rate and hold capacitor patching are illustrated in Figure 2-24.

Integrator capacitors may also be selected on an individual basis. Each integrator is provided with a set of holes similar to those shown for sector or quadrant patching, and all patching rules are similar. Remember, individual capacitor selection predominates over quadrant selection.

Integrator mode control can be accomplished from the rep-op timer, the manual pushbuttons on the keyboard, a slaved analog computer, or the digital computer through an interface. In addition, like time scale selection, the mode can be controlled on a sector/quadrant basis or individually. Mode control via the quadrant inputs is shown in Figure 2-24. The inputs are I, O and H, and patching to these holes will control the eight integrators in the quadrant. The upper illustration shows the patching used for controlling the integrators

INTEGRATOR MODE & TIME SCALE PATCHING

LOGIC PANEL



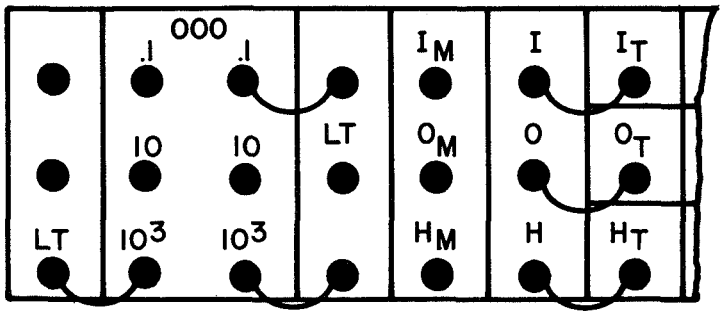
QUADRANT CONTROL

PATCHING EXAMPLE:

RATE CAPACITOR - 1 MFD
HOLD CAPACITOR - 1 MFD
} TIME SCALE FACTOR HOLES .1 AND 10 PATCHED TO LT

MODE CONTROL VIA MANUAL PUSHBUTTONS (OR THE DIGITAL COMPUTER OR ANOTHER ANALOG COMPUTER IF THE REMOTE MODE IS ENABLED).

LOGIC PANEL



QUADRANT CONTROL

PATCHING EXAMPLE:

RATE CAPACITOR - .001 MFD
HOLD CAPACITOR - .01 MFD

MODE CONTROL VIA REP-OP INTERVAL TIMER.

FIGURE 2-24

page 2-54 blank

from the manual pushbuttons, the digital computer or a slave computer. The Im, Om, and Hm outputs come from one of these three sources. With this configuration, the keyboard buttons will normally be in control. However, pushing the REMOTE MODE button on the operators desk disables the keyboard buttons and gives control to a remote external source. A 10-position (0-9) thumbwheel switch selects the source. Position 0 designates the digital computer while, positions 1-9 indicate control from one of nine slaved analog computers.

Mode control from the rep-op timer is shown in the lower illustration. Timer outputs I_t, O_t, and H_t feed the I, O, H inputs. The auxiliary timer outputs I_A, O_A, H_A may also be used. A total of three auxiliary timers may be provided; they are labeled with subscripts A, B and C on logic panel.

Each integrator has its own separate I, O, H control inputs so that individual control of mode can be obtained. As with capacitor selection, the individual inputs override quadrant control.

The patching instructions given here are for conventional operation. However, keep in mind that there is no reason why the mode or capacitor selection cannot be controlled from any other logic source on the panel (gates, flip-flops, control lines, etc.).

2.2.2 Gates

The gates used in the SS-100 logic system are integrated circuit DTL units with logic levels of 0 and +5 volts. There are 2, 4 and 6 input units available, and both the normal and complement (inverted) outputs are available. Both inputs (labeled 1, 2, etc.) and outputs (G1, $\overline{G1}$, etc.) appear at the logic panel.

With no inputs or outputs patched, the $\overline{G1}$ output will be low and the G1 output will be at +5 volts. The result will be the same if both inputs are at +5 volts. With this type of logic gate, an open input circuit produces the same result as a +5 volt input. Grounding any one or more of the inputs will cause the $\overline{G1}$ output to switch to +5 volts and G1 output to ground. The circuit performs the positive logic AND/NAND and negative logic OR/NOR function.

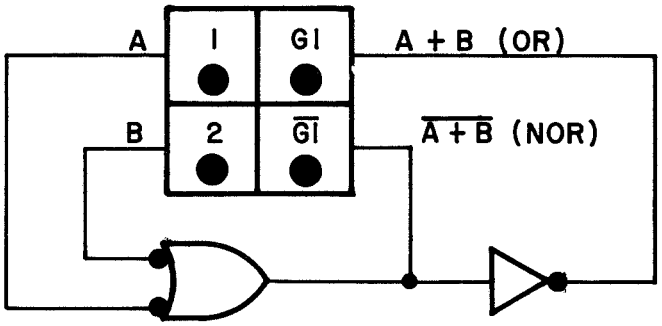
The circuits, symbols, and truth tables in Figure 2-25 define the logic operations performed. Two input gates are shown, but the information also applies in principle to the 4 and 6 input gates.

The gates may also be used for logic inversion. Use any one gate input and the $\overline{G1}$ output. Leave the other inputs open.

The gate inputs represent one unit load, and each output can drive 10 unit loads.

NEGATIVE LOGIC:

LOGIC 0=+5 VOLTS = H
LOGIC 1= 0 VOLTS = L

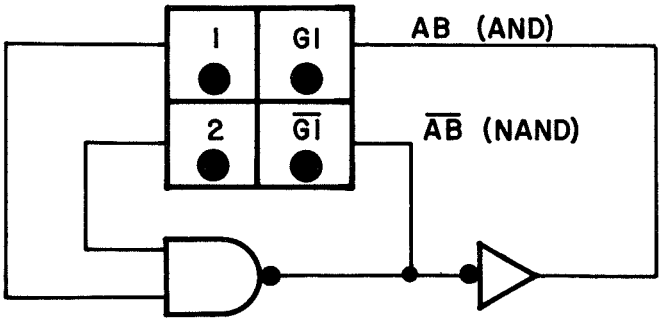


TRUTH TABLE

A	B	G	$\overline{G}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

POSITIVE LOGIC:

LOGIC 0 = 0 VOLTS = L
LOGIC 1 = +5 VOLTS = H



TRUTH TABLE

A	B	G	$\overline{G}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

FIGURE 2-25

2.2.3 Flip-Flops

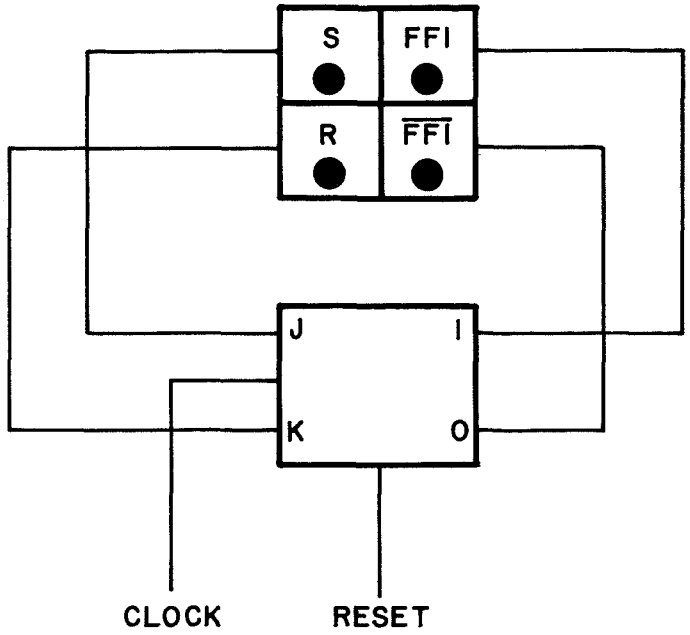
The patchable logic flip-flops may be used in two different configurations, as a JK/clocked flip-flop or as a set-reset/latch flip-flop. The same flip-flops are used for both operations. The JK mode is used when synchronous or clocked operation is selected at the logic panel or by the external clock, if in Ext CLK mode. In this mode the flip-flop is operated by the 1 MHz SS-100 system clock. The flip-flops operate as set-reset units or latches in the asynchronous mode. The circuits and truth tables in Figure 2-26 define the operations in the two modes.

In the JK mode with both S and R (J and K) inputs open or at +5 volts, the flip-flop will simply toggle or complement itself at the selected clock rate, usually 1 MHz. With the S (or J) input at ground (LT) and the R (or K) input high or open, the flip-flop will reset on the next clock pulse making output FF1 ground (logic 1) and $\overline{FF1}$ +5 volts (logic)). With the R input at ground and the S input open or high, the flip-flop will set on the next clock pulse. FF1 will be +5 volts (logic 0) and $\overline{FF1}$ will be 0 volts (logic 1). Making both inputs S and R ground will cause flip-flop to remain in its current state. The clock trigger will have no effect.

In the JK truth table, J corresponds to S and K to R. The designation (t) represents the time period before the clock pulse, and (t + 1) is the time period after the next clock pulse. FF1 and $\overline{FF1}$ are the flip-flop outputs during these time periods. Remember, negative logic is used.

In the asynchronous or non-clocked mode, the flip-flops act as latches. A ground at S sets the flip-flop making FF1 0 volts and $\overline{FF1}$ +5 volts. A ground at R resets the flip-flop, causing FF1 to be +5 volts and $\overline{FF1}$ to be 0 volts. With both inputs open or high, the flip-flop simply remains in its current state. An ambiguous condition occurs when both inputs are grounded. Here both outputs become 0 volts. This condition should be avoided.

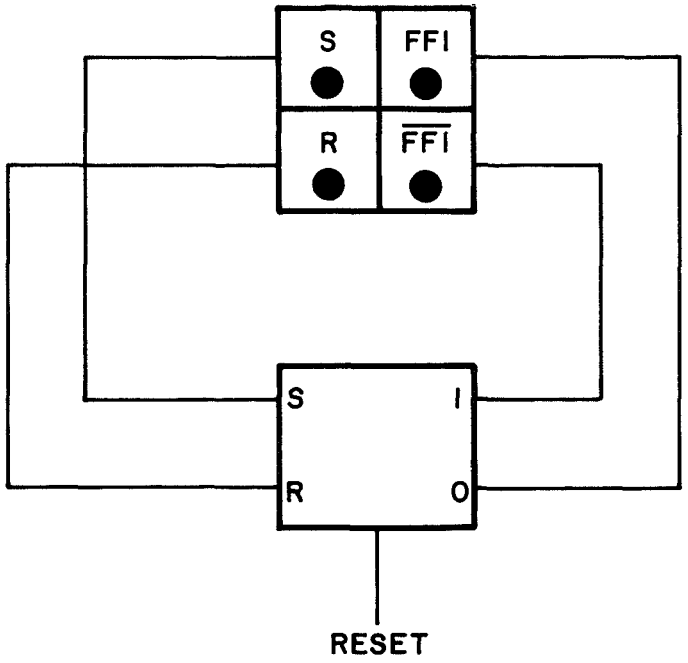
FLIP-FLOPS PATCHING



TRUTH TABLE

J	K	(t) FFI	(t+1) $\overline{FFI}$
0	0	TOGGLES	
0	1	X	0
1	0	X	1
1	1	X	NC

JK (SYNCHRONOUS)



TRUTH TABLE

S	R	FFI	$\overline{FFI}$
0	0	X	$\overline{X}$
0	1	0	1
1	0	1	0
1	1	1	1

SET - RESET
(ASYNCHRONOUS)

NEGATIVE LOGIC:

LOGIC 0 = +5 VOLTS
LOGIC 1 = 0 VOLTS

X = EITHER 1 OR 0
NC = NO CHANGE

FIGURE 2-26

2.2.4 Counter/Register

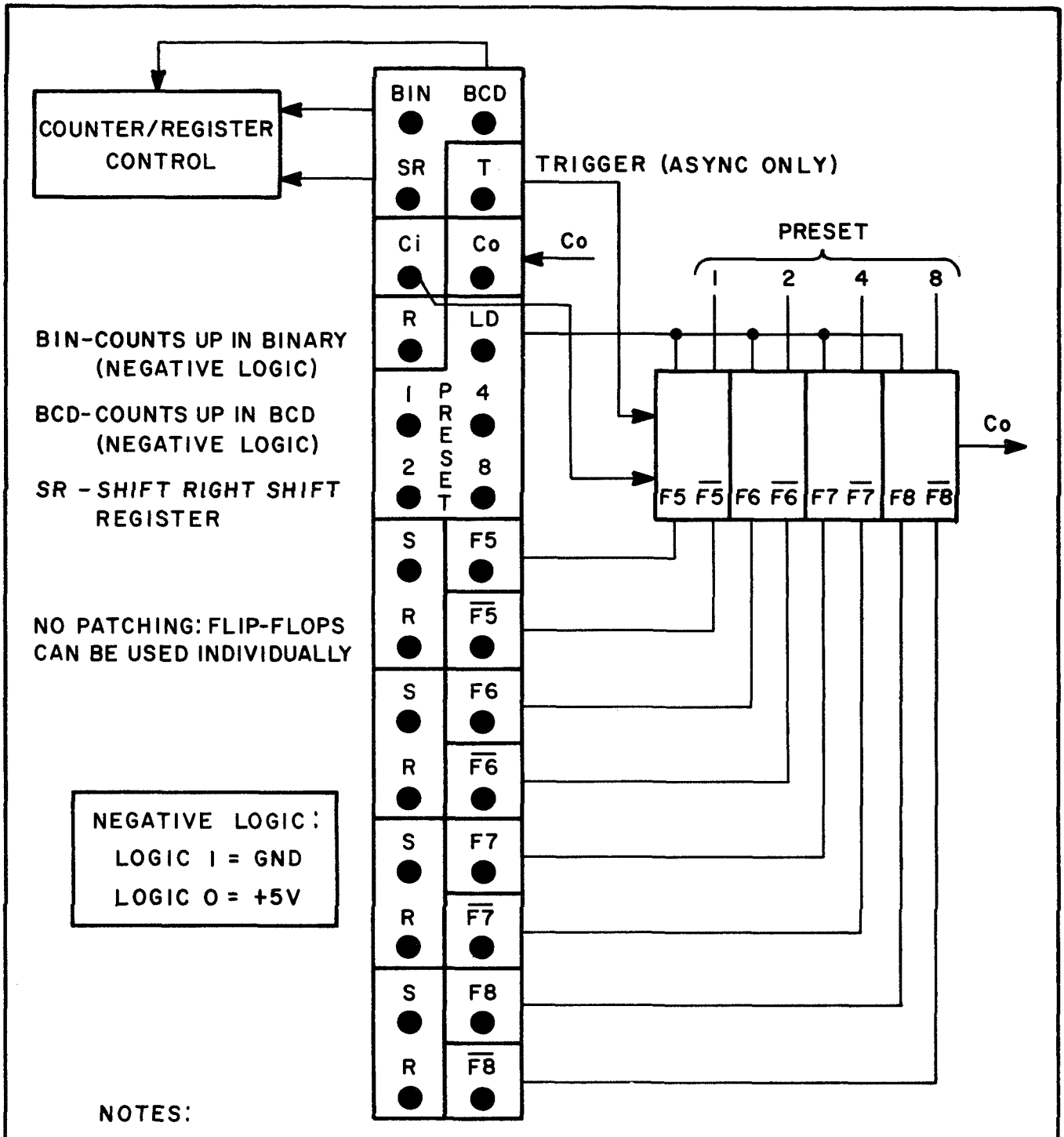
Each patchable logic counter/register is a 4-bit unit that can be used in a variety of modes. The holes BIN, BCD, SR at the top of the counter patching sector on the logic panel as illustrated in Figure 2-27 are used to select the desired operation. With all three holes open, flip-flops act as a shift register. Instead, the flip-flops are free and can be used individually as desired. In addition, the four together may be used as a parallel loaded storage register. The 4-bit number to be stored is entered at the 8421 PRESET inputs. A +5 volt to ground transition at the LD (load) hole will store the input number.

Patching BIN to LT causes the counter to operate in the binary mode. It counts up in negative logic, down in positive logic with a 8421 format. Patching BCD to LT will produce BCD counting in the same format. To use the unit as a shift register, patch SR to LT.

The rules given below apply to using the counter/register units.

a. When using any of the counter/register modes, do not patch to the S and R input holes for the flip-flops.
b. The counters and registers are normally used in the synchronous mode under system clock control. Asynchronous operation of the counter modes is accomplished by triggering at the T input. No asynchronous operation in the shift register mode is permitted.
c. To cascade the counters and register, use the Co and Ci holes. Simple patch the Co of one to the Ci of the next.
d. The R input resets all flip-flops in any mode of operation. A +5 volts to ground transition resets.
e. Negative logic is assumed. Logic 1 = 0 volts, logic 0 = +5 volts. For example, if a flip-flop is set or is storing a logic 1, its F output is 0 volts and its $\overline{F}$ output is +5 volts. At the preset inputs, a logic 1 is a 0 volt level.

COUNTER/REGISTER PATCHING



NOTES:

1. WHEN OPERATING IN A COUNTER/REGISTER MODE, ALL S&R INPUTS TO FLIP-FLOPS SHOULD BE UNPATCHED. THESE INPUTS ARE TO BE USED ON THE INDIVIDUAL FLIP-FLOPS ONLY.
2. TO CASCADE COUNTERS, PATCH Co OF FIRST COUNTER TO Ci OF SECOND COUNTER, ETC.
3. FOR DOWN COUNTING, USE POSITIVE LOGIC. A NEGATIVE LOGIC UP COUNTER IS A POSITIVE LOGIC DOWN COUNTER.

FIGURE 2-27

2.2.5 One Shot

The one shot, also known as a monostable, a pulser or a single shot, is a pulse generator that produces pulses of selectable width. Each time the one shot receives a negative going level transition (+5 volts to 0) a pulse will be generated. The output amplitude switches between 0 and +5 volts. The width is continuously adjustable over the 1 microsecond to 1 second range. A seven-position switch selects the coarse pulse width range, and a single-turn potentiometer is used for continuous resolution. Refer to Figure 2-28.

The input T1 can be driven from any logic source on the panel. The outputs OS1 and $\overline{OS1}$ can drive 10 unit loads. In the stable state before triggering, the OS1 output is at +5 volts and $\overline{OS1}$ is at ground. When triggered, the circuit generates a pulse of the selected duration where OS1 is ground and $\overline{OS1}$ is at +5 volts.

Care should be taken in using the one shot to ensure that the duty cycle (the ratio of the on time to the period of triggering) is less than 60%, otherwise erratic operation may occur.

ONE SHOT PATCHING

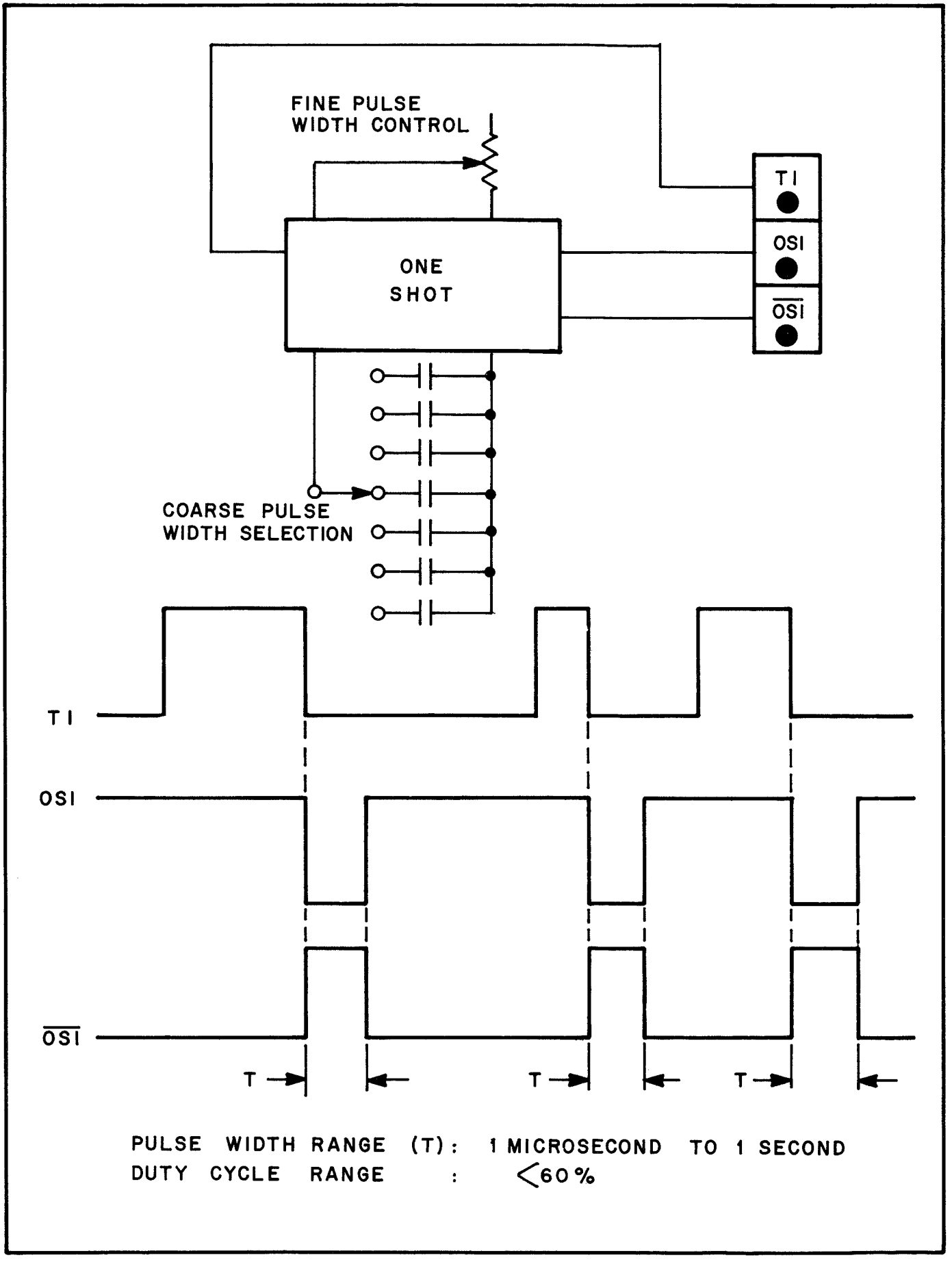


FIGURE 2-28

2.2.6 Differentiator

The differentiator generates a fixed width pulse and its complement when triggered. The pulse width is 500 nanoseconds. Triggering occurs on the trailing edge of the input signal at IN1. The complementary outputs appear at D1F1 and $\overline{\text{D1F1}}$. The input represents one unit load and the outputs can each drive 10 unit loads. Refer to Figure 2-29.

DIFFERENTIATOR PATCHING

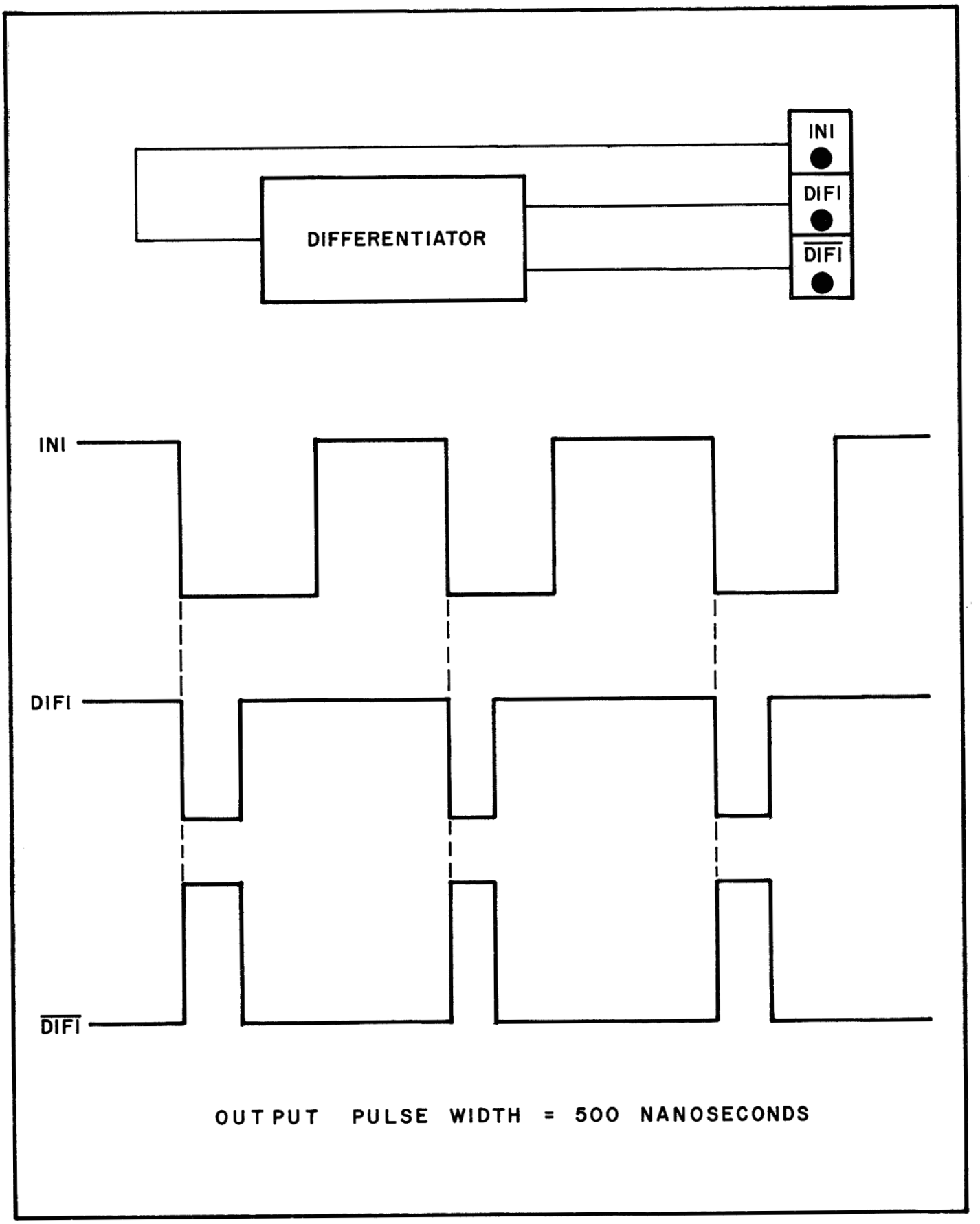


FIGURE 2-29

2.2.7 Logic Pushbutton and General Purpose Indicators

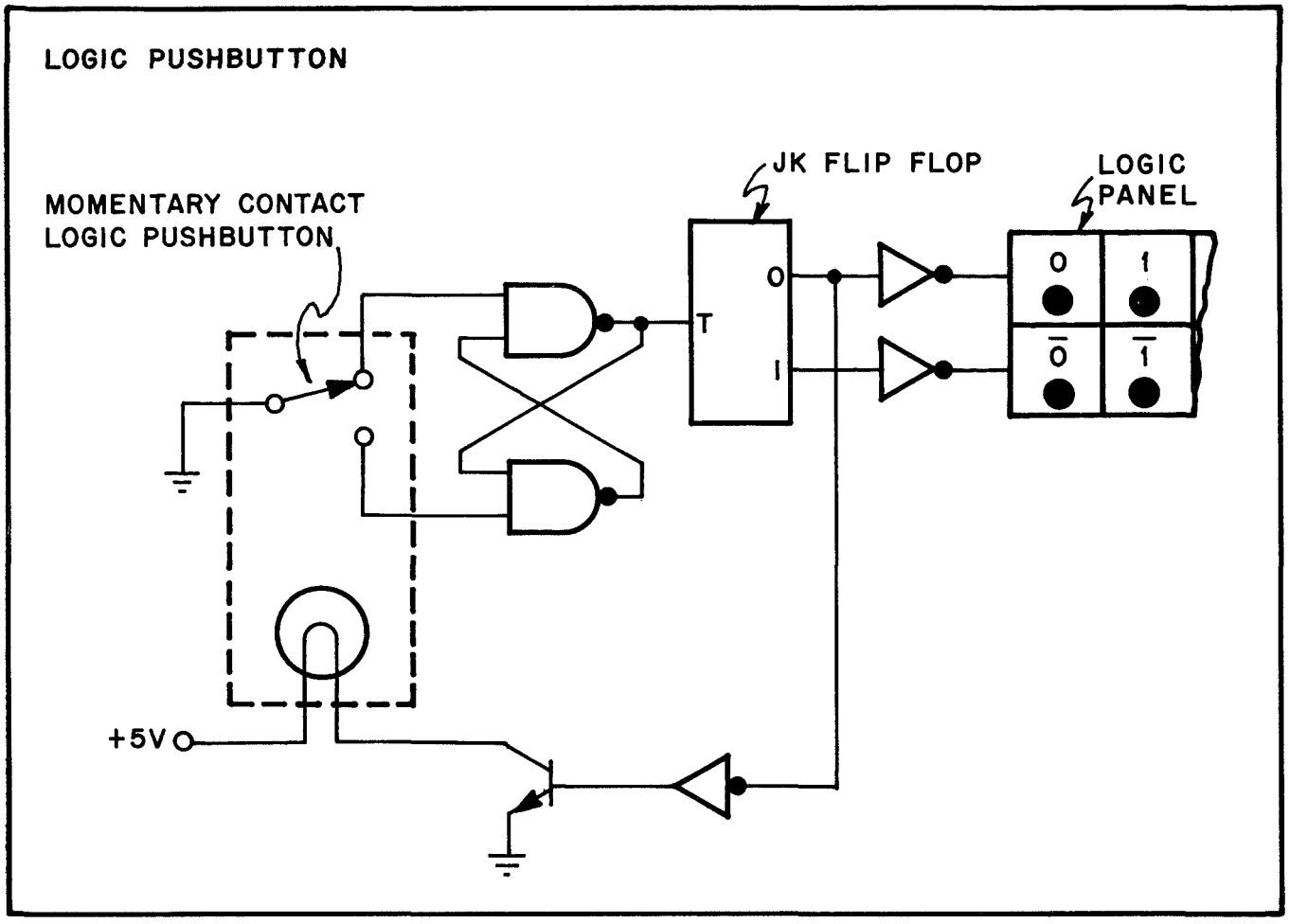
2.2.7.1 Logic Pushbutton

The logic pushbutton is an electronic alternate action type pushbutton switch for generating logic levels at the panel for manual control purposes. The lighted pushbutton switches is the momentary contact type. It is buffered with a latch and used to toggle a JK flip-flop whose outputs appear at the logic panel. Depressing the button once toggles the flip-flop. The outputs remain constant until the button is again depressed. The flip-flop toggles and the complement outputs appear. Each output can drive 10-unit loads. Refer to Figure 2-30.

2.2.7.2 General Purpose Indicators

The general purpose indicators are patchable light driver circuits used for monitoring selected logic functions. With the input line at the patch panel open or at +5 volts, the light is off. Patching the input to ground (LT) causes the light to turn on. The input represents a one unit load, and it can be driven from any logic source on the panel. Refer to Figure 2-30.

LOGIC PUSHBUTTON & GENERAL PURPOSE INDICATOR PATCHING



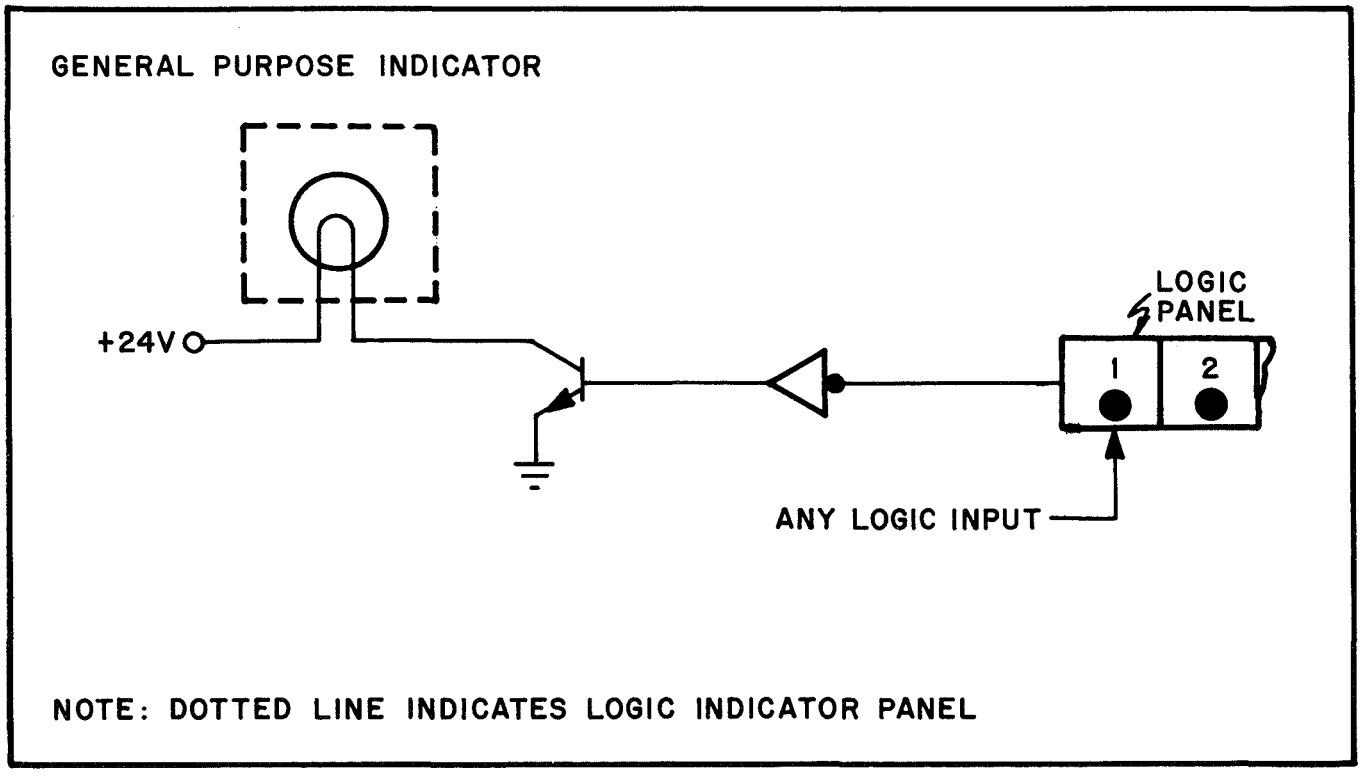


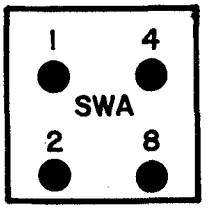
FIGURE 2-30

2.2.8 Preset Thumbwheel Switches

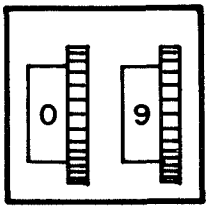
The preset thumbwheel switches provide a convenient means of generating 4-bit 8421 binary or BCD numbers at the logic panel. The outputs, labeled 8, 4, 2, 1, can be used to preset counters and registers and provide a convenient source of data input for digital comparators or other logic circuits.

The thumbwheels are used to set in the decimal value desired, and the equivalent 4-bit binary code appears at the four output holes. The logic levels are 0 volts for logic 1 and +5 volts for logic 0. The output codes for each thumbwheel setting are shown in the table in Figure 2-31. Since four are used, only decimal settings of 00 to 15 are valid.

PRESET THUMBWHEEL SWITCHES PATCHING



LOGIC PANEL



THUMBWHEEL SW - FRONT VIEW
(ONLY SETTINGS OF 00 TO 15 ARE VALID)

4-BIT DECIMAL-TO-BINARY TABLE

THWH. SETTING	OUTPUT HOLES 8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

(0 = +5 VOLTS, 1 = GND)

APPLICATIONS:

1. PRESETTING REGISTERS AND COUNTERS
2. INPUT TO COMPARATORS
3. MANUALLY ENTERING 4 BIT NUMBERS TO LOGIC CIRCUITS.

FIGURE 2-31

APPENDIX A

PATCH PANELS TERMINATION

CODING AND FUNCTIONS

TABLE A-1

ANALOG PATCH PANEL COLOR SCHEME

BASIC COLOR	DOTS	FUNCTION
1. White		summing junctions, DR outputs
2. Gray		feedback circuits, decoupling network, MDFG control, logic trunks
3. Red		pulse reference, multiplier outputs
4. Orange		minus reference, passive elements
5. Yellow		inputs to amplifiers, inputs to IC
6. Blue		amplifier outputs
7. Black		computer ground
8. Purple		function relay
9. Brown		D/A inputs, limiters
10. Green		pots
11. Black	white	multiples
12. Red	white	multiplexor plus switched reference
13. Purple	white	function switch
14. Green	white	D/R high and low inputs, comparator inputs
15. Orange	white	minus switched reference, display
16. Blue	white	MDFG amplifier output, sample/hold output
17. Brown	white	nonlinear inputs
18. Green	black	direct readout
19. Orange	black	select readout
20. Yellow	black	EXT trunks, delay line input, sample/hold inputs
21. White	black	recorders

TABLE A-2

ANALOG PATCH PANEL TERMINATION CODING AND FUNCTIONS

DESIGNATION	COLOR	FUNCTION
HI	Green	High side of hand set or servo set pots.
N	Green	Arm of hand set or servo set pots.
LO	Green	Low side of hand set pots.
IN (DVM)	Green with black dots	Input to DVM through unloading amplifier when in PP DVM mode.
IN (VM)	Green with black dots	Input to fluke transistor voltmeter when in PP TVM mode.
X	Green with black dots	X input to X-Y plotter.
Y	Green with black dots	Y input to X-Y plotter.
HI	Green with white dots	High side of DR converter resistance element.
LO	Green with white dots	Low side of DR converter resistance element.
ECn (a,b)	Green with white dots	Comparator inputs.
-	Orange	-100 volts reference.
1,3	Orange	Passive element 100K ohm resistor.
2,4	Orange	Passive element 1 megohm resistor.
NG	Orange	Noise generator output.
RJ_n	Orange	Output of rate resolver network to SJ of an integrator.
SEL (DVM)	Orange with black dots	Master readout bus connected to DVM input in SEL DVM mode.
SEL (VM)	Orange with black dots	Master readout bus connected to TVM input in TVM SEL mode.

TABLE A-2 (Cont'd)

DESIGNATION	COLOR	FUNCTION
a,b	Orange with black dots	Outputs of X-Y plotter used as a function generator.
-	Orange with white dots	-100 volts static test reference.
Display 1 thru 10	Orange with white dots	Inputs to CRT and memory scope displays.
+	Red	+100 volts reference.
Mn	Red	Multiplier output.
+	Red with white dots	+100 volts static test reference.
TnO	Red with white dots	Delay line output.
MXn	Red with white dots	Multiplexor inputs.
DRn	White	DR converter output.
SJ	White	Summing junction of input network.
SN	White	Summing junction of auxiliary network.
B	White	Base input of operational amplifier.
B^1	White	Common point for FET mode control switches. Connects to B to form an integrator.
OJ	White	Operate junction. Junction of rate capacitor and mode control switch.
IJ	White	Junction of 100K initial condition resistors.
DAn	White	DA switch output to SJ of op amp.
Recorder 1 thru 8	White with black dots	Strip chart recorder inputs.

TABLE A-2 (Cont'd)

DESIGNATION	COLOR	FUNCTION
1	Yellow	1 megohm input summing resistor.
10	Yellow	100K ohm input summing resistor.
IC	Yellow	Initial condition input.
Θn	Yellow	Theta angle input to sine-cosine generator.
SHn	Yellow with black dots	Sample/hold amplifier inputs.
EXT TRUNKS 1 thru 100	Yellow with black dots	External trunks.
IN	Yellow with black dots	Delay line input.
$\dot{\Theta}$n	Yellow with black dots	First derivative of theta angle input to sine-cosine generator.
GND	Black	Signal ground.
Xn	Black with white dots	Multiple or commoning connection.
n	Blue	Outputs of operational amplifiers (summers, integrators, inverters).
-SINn	Blue	Sine output of sine-cosine generator.
Fn	Blue with white dots	MDFG output.
SHn	Blue with white dots	Sample/hold amplifier outputs.
-COSn	Blue with white dots	Cosine output to sine-cosine generator.
n	Gray	Operational amplifier output used for patching a feedback resistor.

TABLE A-2 (Cont'd)

DESIGNATION	COLOR	FUNCTION
F	Gray	Feedback resistor to op amp output.
Mn	Gray	Multiplier output used for patching a feedback resistor.
EXT	Gray	MDFG relay coil that frees output amplifiers from MDFG.
COM	Gray	+24 volts to energize MDFG relay.
Logic Panel 1 thru 8	Gray	Trunks to logic panel. (Fused and voltage limited).
FJ	Gray	Function junction. Output of FDFG network to base of an op amp.
Fn	Gray	Stability network output for FDFG or multiplier feedback resistor.
IC Trunks 1 thru 100	Gray with black dots	Inter-console trunks.
0.1	Brown	DA switch 100K input resistor.
LIM 1, 2 IN	Brown	Limiter input.
0	Brown	Limiter output
+X, -X, +Y, -Y	Brown with white dots	Multiplier or FDFG inputs.
FRn	Purple	Arm of function relay.
NC	Purple	Normally closed contact of SPDT function relay.
NO	Purple	Normally open contact of SPDT function relay.
FSn	Purple with white dots	Arm of function switch n.
L, R, C	Purple with white dots	Left, right and center poles of SPTT function switch.

TABLE A-3

LOGIC PATCH PANEL TERMINATION CODING AND FUNCTIONS

DESIGNATION	COLOR	FUNCTION
I, O, H	Yellow	Mode control inputs for individual integrators whose number appears at the top of the next two columns to the right.
I, O, H	Yellow	Between IM, OM, HM, and IT, OT, HT. This is the I, O, H control for the entire quadrant.
Tn	Yellow	Trigger input to one shots with pulse width controlled from one shot panel above logic patch panel; logic true initiates pulse.
INn	Yellow	Trigger inputs to differentiators Logic true initiates output pulse.
RST	Yellow	Input. Logic true resets all patchable flip-flops on digital patch panel when in sychronous mode.
RUN	Yellow	Input. Logic true allows all patchable flip-flops to run with clocked inputs in synchronous mode.
STOP	Yellow	Input. Logic true stops the clock to patchable flip-flops without resetting when operating in synchronous mode.
1, 2, 3, 4, 5, 6	Yellow	Inputs to six input AND/OR gates. For low level inputs (logic true) the gate functions as an OR gate and for high level inputs as an AND gate.
1, 2, 3, 4	Yellow	Inputs to four input AND/OR gates. For low level inputs the gate performs an OR function and for high level inputs, an AND function.
1, 2	Yellow	Inputs to two input AND/OR gates. Low levels cause the gate to act as an OR gate, while, for high level signals, the gate performs an AND function.

TABLE A-3 (Cont'd)

DESIGNATION	COLOR	FUNCTION
T	Yellow	Toggle input for individual clocking of the patchable flip-flop in groups of four.
Ci	Yellow	Carry-in input to counter/register for a digital carry from another digital component.
R (to left of LD)	Yellow	Reset input. Logic true resets the four individual flip-flops in the column below the patch point.
S, R	Yellow	Set and reset inputs to individual flip-flops. J and K inputs when in the synchronous mode.
G. P. $IND_{(n)}$ 1 - 16	Yellow	Input to general purpose indicator lamps on desk indicator panel for monitoring logic not provided with indicators.
$FR_{(n)}$	Yellow	Inputs for control of function relays, logic true provides a relay closure.
$L_{(n)}$	Yellow	Input for setting comparator latch. Logic true locks comparator in the existing condition.
TS	Green	Timer start. Timer starts on transition of logic input level from true to false.
EXT CLK	Green	Input for operating patchable flip-flops from an external clock.
LD	Green	Load input for presetting the patchable flip-flop with the inputs patched in the preset blocks. Logic true loads preselected binary value into each patchable flip-flop in the same column.
1, 2, 4, 8	Green	Binary inputs for presetting the flip-flop in each respective column. Value is loaded only when LD is at logic true level.

TABLE A-3 (Cont'd)

DESIGNATED	COLOR	FUNCTION
AD	Green	Input to hybrid linkage for the external control of the A-to-D converter.
DA	Green	Input to hybrid linkage for the external control of the D-to-A converter.
PNCH, PRINT	Green	Logic input lines for controlling the start and stop of external paper tape punch and printer.
ADV, REC	Green	Start/stop/record controls for external strip chart recorder.
$DA_{(n)}$ 001-004 101-104 201-204 301-304	Green	DA switch logic control inputs with no patching or logic false (+5 volts); the DA switch is open.
PS, RT, ST	Green	Inputs to set pot set, rate test, static test, or remote modes from any logic signal.
$Sample/Hold_{(n)}$ 0-31	Green	Input for individual control of the sample/hold amplifiers. Logic true puts amplifier in hold.
Sense Lines 0-15	Green	Input lines to hybrid linkage for indication of analog logic condition.
Interrupts 0-15	Green	Inputs to interrupt lines to digital computer.
.1, 10, 10^3	White	The two columns to the immediate right of the yellow mode control inputs which contain the integrator number at the top. The left column of the pair selects the Rate capacitor value and the right hand column selects the Hold capacitor value.
ASYNC	White	Patching to logic true allows the patchable flip-flops to operate in the asynchronous mode.

TABLE A-3 (Cont'd)

DESIGNATION	COLOR	FUNCTION
BIN	White	Logic true patched to this point sets the selected patchable flip-flop to operate in the binary counter mode.
BCD	White	Logic true establishes the selected patchable flip-flops to the BCD counter mode.
SR	White	Patching logic true to SR sets the chosen patchable flip-flops to operate as a shift register.
Signal Panel 1-8	White	Eight protected lines to/from the analog patch panel.
DIG IN	White	Output digital control for S/H amplifiers.
S/H C	White	Input for controlling all sample/hold amplifiers simultaneously.
$M_{(n)}$ 001-006 201-206 301-306 401-403 501-503	White	Input for the control of the electronic multiplier operational mode. Logic true causes a capacitor to be added across the feedback resistor of the output amplifier for stability. Used only in the divided mode.
Logic Trunks 1-48	White	Lines for use with external equipment and available on a connector at the rear of the console.
1R, 2R, ROT INV, RATE, RST	White	Inputs for setting the operational mode of the resolvers.
.1, 10, 10^3	White	Two columns that have 000, 100, 200, 300 at the top are used to select the integrator capacitors in the entire quadrant.
IM, OM, HM	Brown	Outputs that can be remotely set from the desk keyboard or the digital computer to control the integrator mode of operation.

TABLE A-3 (Cont'd)

DESIGNATION	COLOR	FUNCTION
PS	Brown	Pot set output. Logic true indicates the analog is operating in the pot set mode.
RT	Brown	Rate test output. Logic true indicates the analog is operating in the rate test mode.
ST	Brown	Static test output. Logic true indicates the analog is operating in the static test mode.
REM	Brown	Remote output. Logic true indicates the analog mode is being controlled remotely.
RST M	Brown	Output manually set from desk keyboard for resetting patchable flip-flops.
RUN M	Brown	Manually set output to start patchable flip-flops operating in synchronous mode.
STOP M	Brown	Manually set output to stop patchable flip-flops operating in synchronous mode.
$G_{(n)}$	Brown	Output of a logic gate whose inputs appear in the yellow blocks immediately to the left. N is the gate number designation; e.g., G1, G2, ...G12, etc.
$SW_{(n)}$	Brown	Binary coded outputs from thubmwheel switches located on panel above the logic patch panel. The numbers 0 thru 15 may be set manually with these switches: SWA, SWB, SWC, and SWD.
Control Lines 0-15	Brown	Digital outputs from the hybrid linkage for analog logic control.

TABLE A-3 (Cont'd)

DESIGNATION	COLOR	FUNCTION
LOGIC PB	Brown	Output logic levels from logic pushbuttons located on desk panel.
$\overline{IM}$, $\overline{OM}$, $\overline{HM}$	Brown with white dots	Complements of IM, OH, HM outputs.
$\overline{G}_{(n)}$	Brown with white dots	Complement of gate output.
$\overline{LOGIC\ PB}$	Brown with white dots	Complementary output from logic push-buttons.
LT	Red	Logic true for patching (ground).
LF	Red with white dots	Logic false for patching (+5 vdc).
IT, OT, HT	Blue	Initial condition, operate, hold outputs from the master timer, which may be patched into the individual IC, operate, hold or quadrant initial condition, operate, hold.
SET	Blue	Set output. Logic true indicates the addressed servo set pot is set.
NULL	Blue	Null output. Logic true indicates the addressed servo set put has not reached a null condition.
OS1-OS2	Blue	Output of one shots 1 and 2.
DIF1-DIF2	Blue	Output of differentiators 1 and 2.
OVLD	Blue	Output from overload OR gate logic true indicates a component is in overload condition.
C_o	Blue	Carry out output is for a digital carry from the binary or BCD counters.
$F_{(n)}$	Blue	Flip-flop F(n) output.

TABLE A-3 (Cont'd)

DESIGNATION	COLOR	FUNCTION
I_A, O_A, H_A I_B, O_B, H_B I_C, O_C, H_C	Blue	IC, operate and hold outputs of auxiliary rep-op timers A, B, and C.
$EC_{(n)}$ 001-004 101-104 201-204 301-304 401-402 501-502	Blue	Output of electronic comparator whose inputs are on the analog patch panel.
$\overline{IT}$, $\overline{OT}$, $\overline{HT}$	Blue with white dots	Complements of IT, OT, HT outputs.
$\overline{OS1}$-$\overline{OS2}$	Blue with white dots	Complements of one shot output.
$\overline{DIF1}$-$\overline{DIF2}$	Blue with white dots	Complements of differentiator 1 and 2 outputs.
$\overline{F\ (n)}$	Blue with white dots	Complement of flip-flop $F_{(n)}$ output.
$\overline{I_A}$, $\overline{O_A}$, $\overline{H_A}$ $\overline{I_B}$, $\overline{O_B}$, $\overline{H_B}$ $\overline{I_C}$, $\overline{O_C}$, $\overline{H_C}$	Blue with white dots	Complement outputs of auxiliary A, B, and C.
$\overline{EC}_{(n)}$	Blue with white dots	Complement of comparator output (EC).
1M Clock	Orange	Clock output - 1 MHz clock rate.
1M 100K 10K 1K 100	Orange	Clock outputs 1M = 1 MHz, 100K = 100 KHz, 10K = 10 KHz, 1K = 1KHz, 100 = 100 KHz
CLK SW	Orange	Switched 1 MHz clock output. Clock signal appears only in the run logic mode.
$\overline{CLK}$ SW	Orange with white dots	Complementary output of switched clock function.

TABLE A-3 (Cont'd)

DESIGNATION	COLOR	FUNCTION
$\overline{TM}$ Clock	White on Orange Background	Complement of 1 MHZ clock output.
(None)	Black with white lines inter-connecting holes	Multiples for common connection fan out.

READER COMMENT SHEET

Publication: SS-100 Analog/Hybrid Computer Patching Reference Manual, Publication No. HS/40001.0

From: Name _____

Company _____

Address _____

_____

Comments: (Describe errors, suggested additions or changes)

FIRST CLASS
PERMIT NO.
11175
HOUSTON,
TEXAS

BUSINESS REPLY MAIL
No postage necessary if mailed in the United States

POSTAGE WILL BE PAID BY ADDRESSEE

Hybrid Systems, Inc.

4301 DACOMA ROAD
HOUSTON, TEXAS 77018

Attn: Publications Dept.

HYBRID
SYSTEMS, INC.
a subsidiary of

scientific
resources
corporation

4301 Dacoma Road
Houston, Texas 77018
713/681-3581
TWX 910-881-1556

Hybrid Systems, Inc.

1044 HYBRID LINKAGE UNIT

1044 HYBRID LINKAGE UNIT

PROGRAMMING REFERENCE MANUAL

PROGRAMMING REFERENCE MANUAL
FOR THE 1044 HYBRID LINKAGE UNIT
WITH THE IBM SYSTEM/360 MODEL 44

Publication 6030.2
9/68

Hybrid Systems, Inc.

Copyright, 1968, by Hybrid Systems, Inc., Houston, Texas. Printed in the United States of America. This book, or parts thereof, may not be reproduced in any form without permission of Hybrid Systems, Inc.

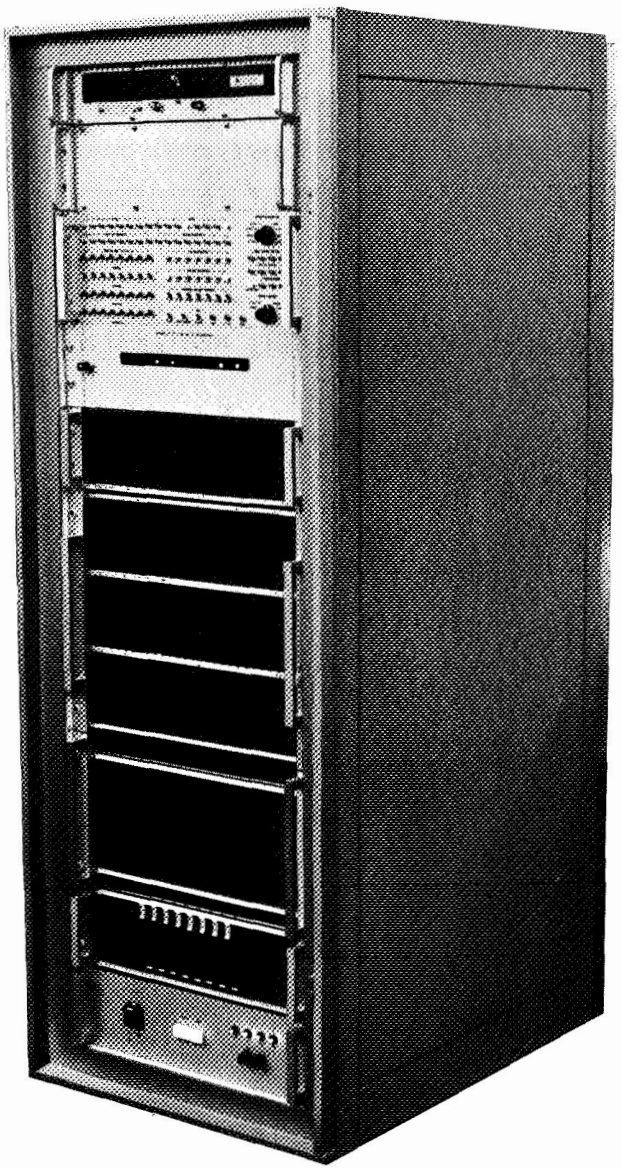
PREFACE

This publication describes the Hybrid Systems 1044 Hybrid Linkage Unit which connects the Hybrid Systems SS-100 Analog/Hybrid Computer and the IBM System/360 Model 44. It connects to the Model 44 via the High Speed Multiplexor Channel and the Priority Interrupt Feature. The manual describes the functional operations of the 1044 and command codes. For additional information, the reader is referred to the following publications.

Hybrid Systems 1044 Instruction Manual, Publication 6031

Hybrid Executive, Programming Guide, Publication 6033

Hybrid Systems SS-100 Analog/Hybrid Computer, Programming Reference Manual, Publication 6011



1044 LINKAGE CABINET

TABLE OF CONTENTS

Section		Page
1.	INTRODUCTION	1
	Analog/Digital Conversion	1
	Digital/Analog Conversion	2
2.	ADDRESSING THE 1044	4
	I/O Address	4
	Subchannel Address	5
	Analog Console Address	6
3.	SYSTEM/360 I/O INSTRUCTIONS	6
	Start I/O Instruction (SIO)	7
	Test I/O Instruction (TIO)	7
	Halt I/O Instruction (HIO)	8
	Execution of Input/Output Commands	9
4.	COMMAND STRUCTURE AND EXECUTION	10
	Subchannel Operation	11
	Analog/Digital Conversion	11
	Sequential Read A/D	11
	Random Read A/D	15
	Repetitive Read A/D	17
	Direct Read A/D	18
	Digital/Analog Conversion	19
	Sequential Write D/A	19
	Random Write D/A	22
	Repetitive Write D/A	24
	Write Addresses	25
	Digital Input	26
	1044 Sense Byte	26
	Sense Lines	28
	Analog Computer Status	28

TABLE OF CONTENTS (Continued)

Section		Page
	Digital Output	28
	Control Lines	29
	Analog Computer Control Functions . .	29
5.	INTERNAL PRIORITY OF THE 1044	30
	Programming for Channel Efficiency	30
6.	TERMINATING SEQUENCE	31
	1044 Unit Status Conditions	32
7.	TEST PANEL	34
8.	PRIORITY INTERRUPTS	41
9.	ANALOG COMPUTER INTERFACE	42

LIST OF FIGURES

		Page
Frontispiece	1044 Hybrid Linkage Unit	ii
Figure 1	Block Diagram of 1044	3
Figure 2	Test Panel	35
Figure 3	Interface Lines For Priority Interrupt Feature	42
Figure 4	Control Interface For SS-100	43

1. INTRODUCTION

The Hybrid Systems 1044 Hybrid Linkage Unit is a high performance conversion and control interface that connects the IBM System/360 Model 44 and the Hybrid Systems SS-100 to form an integrated hybrid computer system. The 1044 attaches to the Model 44 by the High Speed Multiplexor Channel with three subchannels, providing a multichannel communication path between the Model 44 and up to sixteen analog consoles.

Separate subchannels are used to perform analog/digital conversion, digital/analog conversion, and digital input/output. This allows high speed concurrent operation of the functions of the hybrid system. For example, the Model 44 could be programmed to perform a set-up operation on one analog console while a problem is being run on another console. This, of course, is dependent upon the operating system as well as the nature of the problem being run.

The 1044 is programmed similar to a standard peripheral device on the Model 44, with the additional consideration that the hybrid system requires quick response from the digital computer. Therefore, most 1044 operations will be treated as real-time operations by the Model 44.

The 1044 can be used to interface up to sixteen analog consoles. During an operation involving the transfer of data or control information, one console is addressed by the Model 44. More than one operation can be performed concurrently on the three High Speed Multiplexor Subchannels allowing concurrent operations with two analog computers.

Analog/Digital Conversion

The analog-to-digital converter in the 1044 is a high-speed, parallel converter which accepts an analog input signal and converts it to a digital value. The digital value is a binary word which is proportional to the amplitude of the input signal. The format of the data is binary, 13 magnitude bits and sign bit. Negative numbers are represented in two's complement format. The converter output is sign extended to a sixteen bit half word.

There are three modes of converting analog signals in the 1044: Random, Sequential, and Direct. The Random and Sequential modes of conversion are used for converting the analog signals on the multiplexer inputs to the converter which are the problem variables. A Direct Read is a special mode of converting an analog signal connected directly to the input of the analog-to-digital converter. This is used for readout of components in the analog computer.

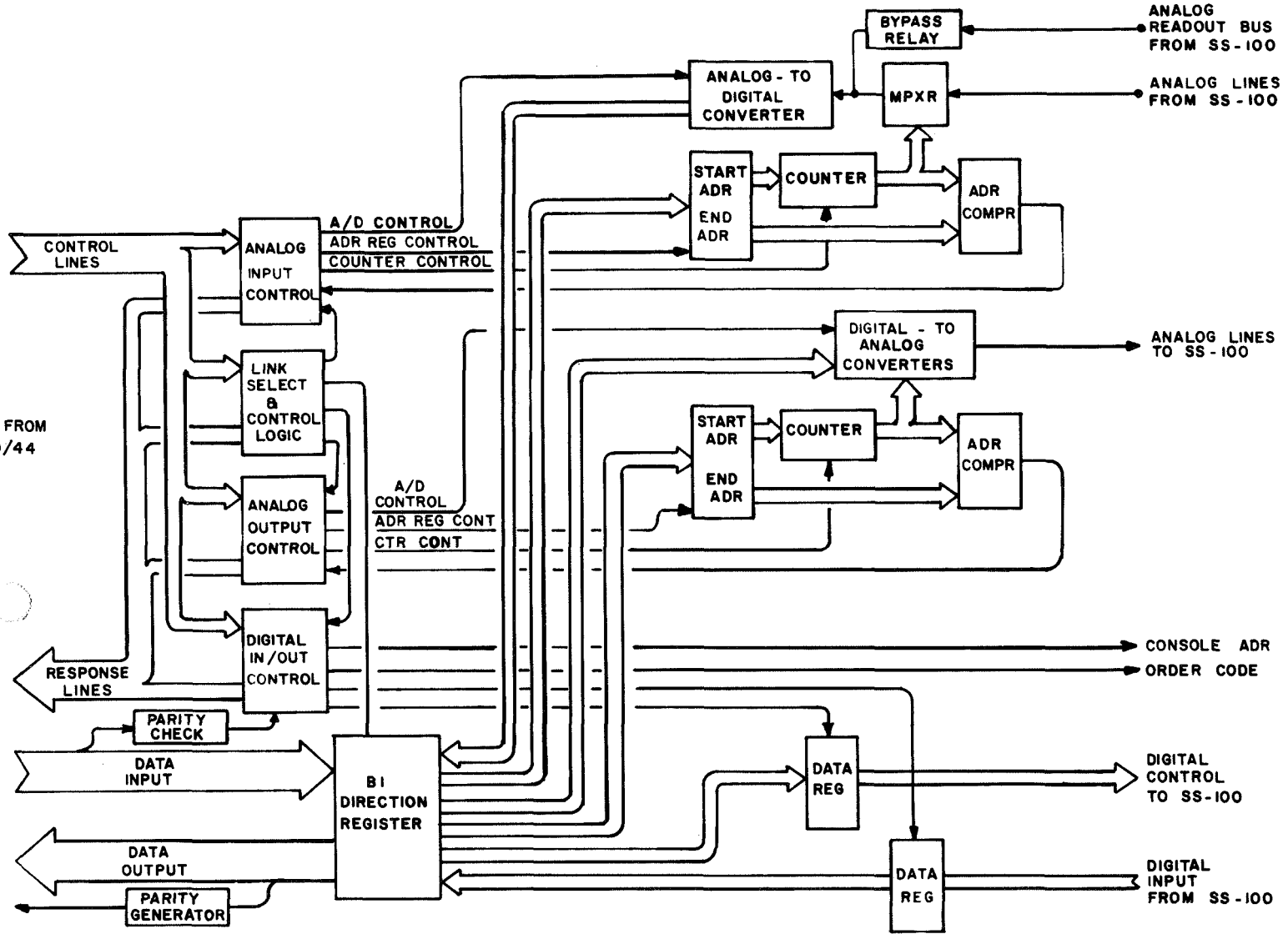
The analog-to-digital converter in the 1044 which has a built in sample and hold amplifier, achieves high speed by using a high degree of parallel operation. The unit contains several sections of level detectors which convert the analog signal. A special error correcting circuit compensates for errors due to uncertainties in detector threshold levels. The output of the level detectors is connected to the decode logic which translates the output voltage into a binary coded signal. The timing of the converter and the analog multiplexer are interlocked to assure receiving valid information.

Digital/Analog Conversion

The 1044 is available with digital/analog converters and digital/resistance converters (also called digital/analog multipliers). Both devices convert a binary word into an equivalent analog signal. The digital/resistance converter also accepts a variable analog input signal and produces as an output the product of the digital and analog variables.

The first rank of a double input register for each converter is loaded from a common data bus. The second register is loaded from the first rank; all registers can be updated simultaneously or individually. The output of the converter is updated upon receiving the start of the conversion signal.

There are two modes of converting digital/analog signals in the 1044: Random and Sequential. These modes address individual converters rather than switching a signal to a common converter as with analog/digital conversion.



BLOCK DIAGRAM OF 1044

Figure 1

2. ADDRESSING THE 1044

The 1044 and the associated access path is designated by an I/O address. The I/O address is a sixteen-bit binary number and consists of two parts: a channel address in the eight high-order bit positions and a device address in the eight low-order bit positions. This section describes the addressing for the Model 44 High Speed Multiplexor Channel and is different for other machines. The channel-address field provides for identifying the number and type of channel as described in the System/360 Principles of Operation. The device address identifies the 1044 and the analog console address.

On the HSMPX channel, the device address identifies the subchannel as well as the 1044 and the analog console address. Addresses with a zero in the high-order bit position of the device-address field identify subchannels that are not shared.

The 1044 allows the subchannel to be used as a shared path; i.e., one additional control unit may be attached to the subchannel.

Input/output devices accessible through more than one channel have a distinct address for each path of communication. This address identifies the channel, the subchannel, and the control unit. In the 1044, this pertains to digital input/output operations as described in the following paragraphs.

Addressing the 1044 under the System/360 operating system is accomplished by the Request Control Blocks, Unit Control Blocks, and all other device routines, in the same manner as for other user added devices.

I/O Address

A write, read, control, or sense operation is initiated by first addressing the channel, subchannel, and device (1044). This is performed by the execution of the SIO instruction. The address field of the SIO instruction is described below.

Bit positions 16-31 of the sum formed by the addition of the content of register B, and the content of the D field identify the channel, subchannel, and I/O device to which the instruction applies (see System/360 Principles of Operation).

During the channel initial selection sequence, the unit address byte is placed on Bus-Out and the 1044 is signaled to decode the I/O device address.

The eight low-order bits of the address field specify the subchannel and the device address as follows:

Bit	0	1	2	3	4	5	6	7
	1	h	h	0	x	x	x	x

Since the 1044 allows the channel to be shared, bit zero should always be equal to "one" identifying a shared-path control unit. Bit 3 identifies the control unit and must be zero for the 1044 address. Bits 1 and 2 specify the subchannel and bits 4 through 7 the analog console address.

Subchannel Address

Two bits of the address byte (bits 1 and 2) identify the subchannel address. The 1044 uses three subchannels for transfer of information between the analog and digital computers. The operations of the 1044 are addressed by subchannels as follows:

Subchannel A (address 00) is used to perform:

1. Read analog/digital data
2. Write analog multiplexer address for Sequential mode A/D
3. Direct Read A/D

Subchannel B (address 01) is used to perform:

1. Write analog multiplexer addresses for Random mode A/D

2. Digital input (sense)
3. Digital output (control)

Subchannel C (address 10) is used to perform:

1. Write digital/analog data and address for Random or Sequential mode
2. Digital input (sense)
3. Digital output (control)

Note that the input/output of digital information may be performed on either subchannel B or C. This allows the operation to be performed on the subchannel with the lowest activity so the other subchannel may be performing another operation.

Analog Console Address

The low-order four bits of the unit address byte are used to identify the analog console. Up to sixteen analog consoles may be addressed by the unit address byte.

The console address remains unchanged in the 1044 until it is modified by a new I/O instruction. When a console is selected by the 1044, a console select line is raised which enables the transfer of information to the selected console. The control of the analog console is described in detail in Section 9.

3. SYSTEM/360 I/O INSTRUCTIONS

The 1044 is programmed by three System/360 I/O instructions: START I/O, HALT I/O, and TEST I/O. Operation of the 1044 is initiated by the execution of a START I/O instruction which references a Channel Address Word (CAW). The address in the CAW references a Channel Command Word (CCW) which contains the command issued to the 1044. The Command byte of the CCW designates the operation to be performed by the 1044. Additional information about the operation is specified by the byte count and address fields of

the CCW. Once the operation is initiated, the System/360 CPU is free to perform other tasks. The 1044 permits full use of the flag bits of the CCW as described in the System/360 Principles of Operation.

The HALT I/O instruction is used to allow the CPU to terminate an operation in the 1044 prior to completion of the I/O transfer. This would be used when an error condition or other abnormal condition is detected. The TEST I/O instruction is used to determine the condition of the 1044 and to clear pending interruption conditions.

The 1044 recognizes four System/360 I/O Commands which are designated in the Command byte of the CCW. The four commands are:

READ
WRITE
SENSE
CONTROL

The following sections describe the operations performed by the 1044 as a function of channel programming of the System/360 Model 44.

Start I/O Instruction (SIO)

The CPU must be in the supervisor state to execute a START I/O instruction. The SIO instruction can initiate a write, read, read backward, control, or sense operation. The read backward command is not recognized by the 1044 as a valid command. The CCW designated by the address in the CAW specifies the operation to be performed, the main-storage area to be used, and the action to be taken when the operation is completed. The I/O operation specified by the SIO is initiated if the addressed I/O device and subchannel are available, the channel is available or is in the interruption-pending state, and errors or exceptional conditions have not been detected.

Test I/O Instruction (TIO)

The state of the addressed channel, subchannel, and device (1044) is indicated by setting the condition code in the PSW and, under certain

conditions, by storing the CSW. Pending interruption conditions may be cleared. The instruction TEST I/O is executed only when the CPU is in the supervisor state. The TEST I/O instruction permits the program to clear interruption conditions selectively by I/O device.

Halt I/O Instruction (HIO)

Execution of the current I/O operation at the addressed I/O device, subchannel, or channel is terminated. The subsequent state of the subchannel depends on the type of channel. The HALT I/O instruction is executed only when the CPU is in the supervisor state.

When the channel is either available or in the interruption-pending state, with the subchannel either available or working, HIO causes the addressed device to be selected and to be signaled to terminate the current operation, if any. If the subchannel is available, its state is not affected. If, on the multiplexor channel, the subchannel is working, data transfer is immediately terminated, but the subchannel remains in the working state until the device provides the next status byte, whereupon the subchannel is placed in the interruption-pending state.

When HALT I/O is issued to a channel operating in the burst mode, data transfer for the burst operation is terminated, and the device performing the burst operation is immediately disconnected from the channel. The subchannel and I/O device address in the instruction, in this case, is ignored.

When an HIO instruction causes data transfer to be terminated, the 1044 remains unavailable until the data-handling portion of the operation in the control unit is terminated. Termination of the data-transfer portion of the operation is signaled by the generation of channel end and device end.

Execution of Input/Output Commands

The System/360 initiates an I/O operation by the execution of an SIO instruction which addresses the channel, subchannel, and device and references the Channel Address Word. The CAW specifies the address of the first CCW associated with the START I/O instruction. The pertinent information thereafter is stored in the subchannel, and the program is free to change the CAW.

The CCW specifies the command to be executed by the 1044, the storage area associated with the operation, and the action to be taken whenever transfer to or from the area is completed. The first CCW is fetched during the execution of the SIO. Each additional CCW in the sequence is obtained when the operation has progressed to the point where the additional CCW is needed.

The low-order byte of the CCW, bits 0-7 specify the operation to be performed by the 1044. The two low-order bits or, when these bits are 00, the four low-order bits of the command code identify the operation to the channel. The remaining bits are modifier bits which identify the operation to the 1044.

The command code assignment for the valid 1044 commands is listed in the following table. The symbol M identifies a modifier bit which is explained in Section 4.

CODE	COMMAND
MMMM0100	Sense
MMMMMM01	Write
MMMMMM10	Read
MMMMM111	Control

Whenever the channel detects an invalid command code, the program check condition is generated. When the 1044 detects an invalid command code, a Unit Check condition is generated, the command is rejected, and the

operation is terminated. The command reject bit of the unit sense byte is set. The command reject bit is reset by issuing a valid command other than TEST I/O, NOP or Sense.

The use of the flag bits in the CCW is unrestricted by the 1044 except for the Suppress Length Indication (SLI). This flag bit must always be on or an incorrect length error will occur. The following table lists the valid 1044 commands and indicates which flags are defined for each command. The channel ignores the flags on all commands for which they are not defined.

COMMAND	FLAG				
Write	CD	CC	SLI		PCI
Read	CD	CC	SLI	SKIP	PCI
Control	CD	CC	SLI		PCI
Sense	CD	CC	SLI	SKIP	PCI

CD - Chain data
CC - Chain command
SLI - Suppress length indication
SKIP - Skip
PCI - Program-controlled interruption

The WRITE and READ commands are used to perform analog/digital and digital/analog operations. The SENSE and CONTROL commands are used to perform input/output of digital information. These commands are explained in detail in the following section on Command Structure and Execution.

4. COMMAND STRUCTURE AND EXECUTION

The channel command word (CCW) specifies the command to be executed, and for commands initiating I/O operations, it designates the storage area associated with the operation and the action to be taken whenever transfer to or from the area is completed.

The READ and WRITE commands are used to perform analog/digital and digital/analog conversion or addressing of these units. The SENSE and CONTROL commands are used to perform the input/output of digital information. These operations are described in detail in the following paragraphs.

Subchannel Operation

The HSMPX channel can be operated in two modes: burst mode and multiplex mode.

In the multiplex mode, the channel can concurrently sustain one data-transferring I/O operation per subchannel, provided that the aggregate rate does not exceed the channel capacity.

In the burst mode, the HSMPX channel operates similar to a selector channel, which has only one subchannel. Only one device at a time communicates with the CPU via the particular channel, but communication is at a higher maximum data rate than in the multiplex mode.

The 1044 allows the programmer to determine the mode of operation of the HSMPX channel by use of the modifier bit in the CCW command byte. The 1044 may force burst mode under program control for a high speed data transfer operation.

Analog/Digital Conversion

The 1044 may be programmed to perform four modes of analog/digital conversion: Sequential, Random, Repetitive and Direct. Repetitive is a special case of a Sequential Read.

Sequential Read A/D

Command byte	Bit	0	1	2	3	4	5	6	7
		m	0	m	m	0	0	1	0

A Sequential Read A/D is performed on HSMPX subchannel A by the following sequence of operations.

The analog multiplexer (AMPX) addresses are set by a write operation which loads the Starting Address Register and Ending Address Register. These registers remain set until they are modified by another write address operation.

The Sequential Read can be initiated in two ways. Command chaining can be used to chain the Sequential Read command immediately after the Write AMPX address command. Two SIO instructions could also be used to issue the commands to the 1044.

Once the Starting Address Register and the Ending Address Register have been set, it is not necessary to reinitialize them on subsequent Sequential Read operations. The Sequential Read can then be initiated by one SIO instruction.

During the Sequential Read operation, the value of the Starting Address Register is incremented in a counter register after each conversion is performed and it is then compared with the value of the Ending Address Register. When the counter is equal to the value of the Ending Address Register, it is reset to the initial value of the Starting Address Register. The number of conversions performed is determined by the byte count in the CCW.

When a Sequential Read command is executed without reinitializing the Starting and Ending Address Registers, the operation will begin reading from the AMPX channel which was addressed last.

Each time a conversion is completed, two bytes of data are transmitted to memory and the CCW byte count is decremented by two. When the byte count equals zero, the operation is terminated.

Data Format

The format of the data from the analog/digital converter is binary with negative numbers represented in two's complement format. The data is sign extended to a sixteen-bit half word, allowing use of half-word arithmetic instructions on the System/360.

Modifier Bits for Sequential Read A/D

Three modifier bits in the Sequential Read command allow the programmer to specify additional options, as described below.

Bit 0 - mode of subchannel operation
Bit 2 - synchronization
Bit 3 - sample/hold amplifier control

Bit 0 - If bit zero equals "0", the HSMPX subchannel operates in multiplex mode. If bit zero equals "1", the HSMPX is forced to burst mode of operation by the 1044.

Bit 2 - If bit two equals "0", the analog/digital conversion rate is controlled by internal synchronization. If bit two equals "1", the conversion rate is controlled by external synchronization.

When internal synchronization is used, the conversion rate will be determined by the channel transfer rate or the conversion rate of the analog/digital converter, whichever is slower.

External synchronization allows the analog/digital conversion rate to be synchronized to an external timing signal, usually from the analog computer.

Bit 3 - If bit three equals "0", the digital computer has no control over the sample and hold amplifiers. If bit three equals "1", the digital computer can control the operation of the sample and hold amplifiers as described below.

During the Sequential Read Operation, each time the address counter is advanced, it is compared with the contents of the Starting Address Register and the Ending Address Register. When a compare equal is made with the contents of the Starting Address Register, the Sample and hold amplifiers are switched to "Hold" mode. They will remain in "Hold" until a compare equal is made with the contents of the Ending Address Register, when they are switched into "Track" mode.

The programmer must consider the tracking speed of the sample and hold amplifiers in determining the conversion rate on a continuous Sequential Read. Adequate time must be allowed for the amplifiers to track the signal before switching into "Hold" mode, or a tracking error will occur.

The following example shows the CCW command bytes to initiate a Sequential Read A/D on AMPX channels 0 through 15 with the channel operating in burst mode, internal synchronization, and control of the sample and hold amplifiers from the digital computer.

First CCW initializes the AMPX addresses (bit 5 equal to one specifies that the data transferred are AMPX addresses).

Command byte	Bit	0	1	2	3	4	5	6	7
		1	0	0	0	0	1	0	1

The byte count field of this CCW is equal to 2, causing two bytes of data to be transferred from memory to the 1044 and stored in the Starting Address Register and the Ending Address Register of the 1044.

Second CCW starts the Sequential Read A/D

Command byte	Bit	0	1	2	3	4	5	6	7
		1	0	0	1	0	0	1	0

The byte count field of this CCW is equal to 32 (16 analog multiplexer channels at two bytes per channel).

Random Read A/D

Command byte

Bit	0	1	2	3	4	5	6	7
	m	1	m	m	0	0	1	0

A Random Read A/D is performed on HSMPX subchannels A and B. Subchannel B is used to transfer a list of AMPX channel addresses from memory to the 1044; subchannel A is used to transfer the data from the analog/digital converter to memory. This operation should always be performed with the HSMPX channel operation in multiplex mode. Two Start I/O instructions are executed by the CPU to initiate this operation. Either subchannel may be issued the Start I/O first; however, the data conversion will not begin until the second Start I/O instruction is executed by the CPU.

The AMPX addresses are in the low-order eight bits of a half word. The byte count of both CCW's for this operation must be equal for the operation to terminate properly.

When the Start I/O instructions for both subchannel A and B have been executed and the correct CCW's issued to the 1044, the Random Read A/D is initiated. The first AMPX address is transferred over subchannel B to the AMPX address control. The AMPX will be set to the specified address and the analog/digital conversion initiated.

The 1044 will then request the channel to transmit two more bytes of data which is the next AMPX address. When the analog/digital conversion is complete, the 1044 transfers two bytes of data to memory via subchannel A. The address control then sets the AMPX to the next address and requests two more bytes of data. Each time a conversion is performed, two bytes of data are transferred to memory and two bytes of address are transferred to the 1044. When the byte count in both CCW's is decremented to zero, the operation is terminated.

Modifier Bits for Random Read A/D

The modifier bits in the CCW command byte allow additional control over the operation as described below.

Bit 0 - mode of subchannel operation
Bit 2 - synchronization
Bit 3 - sample/hold amplifier control

Bit 0 - If bit zero is equal to "0", the addressed HSMPX channel will operate in multiplex mode. If bit zero is equal to "1", the addressed channel is forced to burst mode of operation.

In order for this operation to function properly, the channel must be operated in multiplex mode, allowing concurrent transfer of addresses and data via the subchannels. If only a single analog multiplexer channel is being read, the operation can be performed in burst mode.

Bit 2 - If bit two is equal to "0", the analog/digital conversion rate is controlled by internal synchronization. If bit two is equal to "1", the conversion rate is controlled by external synchronization. This control is the same for Random Read as previously described for a Sequential Read A/D.

Bit 3 - If bit three is equal to "0", the digital has no control over the sample and hold amplifiers. If bit three equals "1", the digital computer can control the operation of the sample and hold amplifiers as described below.

This control is slightly different than described for a Sequential Read, since there is no Starting Address or Ending Address associated with the operation.

The half word containing the AMPX address determines the control of the sample and hold amplifiers. The first byte contains the AMPX address and the second byte contains a comparison address. If these addresses

are not equal, the sample and hold amplifiers are set to "HOLD" mode. If these addresses are equal, the sample and hold amplifiers are set to "TRACK" mode.

The following example shows the CCW command bytes to initiate a Random Read A/D on AMPX channels 1, 4, 9, and 13, with the channel operating in multiplex mode, internal synchronization and no control of the sample and hold amplifiers from the digital computer.

First CCW issues the Random Read command to the 1044 via subchannel A.

Command byte	Bit	0	1	2	3	4	5	6	7
		0	1	0	0	0	0	1	0

The byte count field of this CCW is equal to 8.

Second CCW issues the Write Random Address command to the 1044 via subchannel B (bit 5 equal to one specifies that the data transferred is AMPX addresses).

Command byte	Bit	0	1	2	3	4	5	6	7
		0	1	0	0	0	1	0	1

The byte count field of this CCW is equal to 8. The first byte of each half word contains the AMPX address specified: 1, 4, 9, and 13. The second byte of each half word has no effect in this example.

Repetitive Read A/D

Command byte	Bit	0	1	2	3	4	5	6	7
		m	0	m	0	0	0	1	0

A Repetitive Read A/D is performed on HSMPX subchannel A, the same as a Sequential Read A/D. The command byte is the same as for a Sequential

Read A/D. The Repetitive Read is performed by setting the Starting Address Register and Ending Address Register to the same address. The AMPX is then set to this address and does not change during the entire operation.

The use of the modifier bits is the same as previously described for a Sequential Read except for sample and hold amplifier control. Sample and hold control from the digital computer should not be used for this operation. If the sample and hold amplifier is connected to the multiplexer input, it should remain in the "TRACK" mode.

Direct Read A/D

Command byte	Bit	0	1	2	3	4	5	6	7
		m	0	0	0	0	1	1	0

A Direct Read A/D is performed on HSMPX subchannel A. It is used to perform the readout of analog components in the analog computer and is executed concurrent with a Digital Output operation to address the desired component.

The Direct Read A/D causes a switch to connect an analog signal from the addressed analog console directly to the input to the analog/digital converter. There is a high impedance buffer input amplifier on the input to the analog/digital converter to minimize loading error.

The output of the addressed analog component is then converted and transmitted to memory.

Modifier Bits for Direct Read A/D

Bit 0 - mode of subchannel operation
Bit 2 - not used
Bit 3 - not used

Bit 0 - This operation will normally be performed with the channel operating in multiplex mode. This will allow the analog component

addresses to be output on subchannel B or C and the converted data word to be input on subchannel A.

This operation will be performed under external synchronization. This is required to allow the address system in the analog computer time to address the specified component and connect the output to the readout bus, which is connected to the input to the analog/digital converter.

Digital/Analog Conversion

The 1044 is optionally available with digital/analog converters and digital/resistance multipliers. The programming of the 1044 is the same for either type of device.

Three modes of digital/analog conversion can be performed by the 1044: Sequential, Random, and Repetitive.

Data Format

The format of the data from the digital computer is binary with negative numbers represented in two's complement format. The data is sent via the HSMPX subchannel C as a sixteen-bit half word. When the data is transferred to the D/A input buffer registers, the two high-order bits are truncated.

Sequential Write D/A

Command byte	Bit	0	1	2	3	4	5	6	7
		m	0	m	m	1	0	0	1

A Sequential Write D/A is performed in HSMPX subchannel C by the following sequence of operations.

The D/A address control has an address counter (ACTR) which functions very similar to the analog multiplexer addressing for A/D conversion.

The D/A addresses are set by a write operation which loads the Starting Address Register and Ending Address Register. These registers remain set until they are modified by another write address operation.

The Sequential Write can be initiated in two ways. Command chaining can be used to chain the Sequential Write command immediately following the Write Address command. Alternately, two SIO instructions can be used to issue these commands to the 1044.

Once the Starting Address Register and the Ending Address Register have been set, it is not necessary to reinitialize them on subsequent Sequential Write operations. The Sequential Write can be initiated by one CCW in this case.

During the Sequential Write operation, the value of the address counter is incremented after each conversion is performed and it is then compared with the value of the Ending Address Register. When the address counter (ACTR) is equal to the value of the Ending Address Register, it is reset to the initial value of the Starting Address Register. The number of conversions performed is determined by the byte count in the CCW.

When a Sequential Write command is executed without reinitializing the Starting and Ending Address Registers, the operation will begin writing to the channel which was addressed last.

Each time a conversion is performed, two bytes of data are transferred from memory and the CCW byte count is decremented by two. When the byte count equals zero, the operation is terminated.

Modifier Bits for Sequential Write D/A

Three modifier bits in the Sequential Write command allow additional control options as described below.

Bit 0 - mode of subchannel operation
Bit 2 - synchronization
Bit 3 - double buffer control

Bit 0 - If bit zero equals "0", the HSMPX channel operates in multiplex mode. If bit zero is equal to "1", the HSMPX channel is forced to burst mode of operation by the 1044.

Bit 2 - If bit two is equal to "0", the digital/analog conversion rate is controlled by internal synchronization. If bit two is equal to "1", the conversion rate is controlled by external synchronization. This feature is the same as previously described for analog/digital conversion.

Bit 3 - This modifier bit is used for double buffer control of the digital/analog converter inputs. If bit three is equal to "0", no double buffering occurs. When a word is transferred from the digital computer, it is first stored in the input buffer register of the digital/analog converter. When no double buffer control is used, the data are immediately transferred from the input buffer register to the conversion control register and the conversion initiated on that converter.

If bit three is equal to "1", double buffering occurs as follows. During the Sequential Write operation, each time a conversion is performed, the address counter (ACTR) is incremented and compared with the contents of the Ending Address Register. When a compare equal is made with the contents of the Ending Address Register, a jam transfer from the input buffer register to the conversion control register occurs and the conversion is initiated simultaneously on all channels.

The following example shows the CCW command bytes to initiate a Sequential Write D/A on channels 0 through 20 with the HSMPX channel operating in burst mode, internal synchronization and double buffering.

First CCW initializes the ACTR addresses:

Bit	0	1	2	3	4	5	6	7
	1	0	0	0	1	1	0	1

The byte count field of this CCW is equal to 2, causing two bytes of data to be transferred from memory to the Starting Address Register and the Ending Address Register. Bit 5 equal one specifies that the data transferred are AMPX addresses.

Second CCW starts the Sequential Write D/A:

Bit	0	1	2	3	4	5	6	7
	1	0	0	1	1	0	0	1

The byte count field of this CCW is equal to 42, causing conversion of the specified 21 D/A channels.

Random Write D/A

Command byte

Bit	0	1	2	3	4	5	6	7
	m	1	m	m	1	0	0	1

A Random Write D/A is performed on HSMPX subchannel C. The address field of the CCW is the starting address of a table which contains the D/A addresses and data. The low order byte contains the D/A address and the high-order two bytes contain the data, as shown below.

Word Format for Random Write

0 — 7	8 — 15	16 — 31
Address	Comparison Address	D a t a

This format allows the information to be stored in half word boundaries to facilitate half word arithmetic operations. The byte count field is 4n, where n is the number of conversions to be performed. When the byte count is decremented to zero, the operation is terminated.

Modifier Bits for Random Write D/A

Three modifier bits in the Random Write command allow additional control options, as described below.

Bit 0 - mode of subchannel operation
Bit 2 - synchronization
Bit 3 - double buffer control

Bit 0 - If bit zero is equal to "0", the addressed HSMPX channel will operate in multiplex mode. If bit zero is equal to "1", the channel is forced to burst mode of operation.

Bit 2 - If bit two is equal to "0", the digital/analog conversion rate is controlled by internal synchronization. If bit two is equal to "1", the conversion rate is controlled by external synchronization. This is the same as previously described for analog/digital conversion.

Bit 3 - This modifier bit is used for double buffer control. If bit 3 is equal to "0", no double buffering occurs.

If bit three is equal to "1", double buffering occurs, as follows. During the Random Write operation, the half word containing the D/A address determines the double buffer control. The format of the word is shown below.

0 7	8 15	16 31
Address	Comparison Address	Data

Note that during this operation, the second byte is used for the comparison address for double buffer control in the same way the Ending Address Register is used for a Sequential Write. When the D/A address and the comparison address are equal, the jam transfer occurs and conversion is initiated.

The following example shows the CCW command byte to initiate a Random Write D/A on digital/analog converter channels 2, 5, 10, and 11, with the channel operating in burst mode, internal synchronization, and double buffer control.

The CCW issues the Random Write command to the 1044 via subchannel C.

Bit	0	1	2	3	4	5	6	7
	1	1	0	1	1	0	0	1

The byte count field of the CCW is equal to 16. The format of the data table is below.

2	0	Data
5	0	Data
10	0	Data
11	11	Data

Bit 0 7 8 15 16 31

Repetitive Write D/A

Command byte

Bit	0	1	2	3	4	5	6	7
	m	0	m	m	1	0	0	1

A Repetitive Write D/A is performed on HSMPX subchannel C, the same as a Sequential Write D/A. The command byte is the same as for a Sequential Write D/A. The Repetitive Write is performed by setting the Starting Address Register and the Ending Address Register to the same address. The ACTR is then set to this address and does not change during the entire operation.

The use of the modifier bits is the same as previously described for a Sequential Write except for the double buffer control. Since this operation is used to perform a high-speed update on a single D/A channel, the double buffer feature should not be used.

Write Addresses

The commands previously described for either Sequential Read A/D or Sequential Write D/A require that the appropriate address registers in the 1044 be loaded prior to initiating the operation. This can be accomplished by executing two SIO instructions or by command chaining to issue two CCW's by one SIO instruction.

The command byte format to load the Starting Address Register and Ending Address Register is shown below.

Load Starting Address Register and
Ending Address Register for Analog Multiplexer

Command byte	Bit	0	1	2	3	4	5	6	7
		m	0	0	0	0	1	0	1

Load Starting Address Register and
Ending Address Register for D/A Address Counter

Command byte	Bit	0	1	2	3	4	5	6	7
		m	0	0	0	1	1	0	1

Modifier bit zero will control the mode of operation of the HSMPX as previously explained. The remaining modifier bits are not used for this operation.

Data Format

The data format of the address to be sent to the address registers is binary. The Starting Address is right justified in the low-order byte, and the Ending Address in the high-order byte as shown below.

Starting Address	Ending Address
0 — 7	8 — 15

Digital Input

A Digital Input operation is initiated by a Sense command which sets up the subchannel to transfer data from the 1044 to main storage. This operation may be performed on either HSMPX subchannel B or C.

The data transferred during this operation may come from several different sources identified by the modifier bits in the CCW command byte. The devices which can be addressed by this command include:

	Command byte Bit	0	1	2	3	4	5	6	7
1044 Sense Byte		0	0	0	0	0	1	0	0
Sense Lines		0	0	0	1	0	1	0	0
Analog Computer Status		0	0	1	0	0	1	0	0

Command byte Bit	0	1	2	3	4	5	6	7
	m	m	m	m	0	1	0	0

The digital input operation is initiated by executing an SIO instruction which issues a CCW to either subchannel B or C. The command byte of the CCW is recognized by the HSMPX as a SENSE command. The four modifier bits identify the operation to be performed by the 1044. The meaning of the modifier bits is as follows. Bit zero determine the mode of operation of the subchannel as previously described. The remaining modifier bits identify a command to read the Sense lines or the analog computer status as described in the following paragraphs.

1044 Sense Byte

Command byte Bit	0	1	2	3	4	5	6	7
	0	0	0	0	0	1	0	0

Information provided in the Sense byte describes the condition of the 1044 and is more detailed than the information provided in the status byte described in Section 6. Any condition which sets a bit in the Sense byte will also set the Unit Check bit in the status byte.

Therefore, if the program detects a Unit Check condition at the completion of an operation on the 1044, the sense byte would usually be read to determine the cause of the Unit Check.

The meaning of the bits in the Sense byte are as follows:

Bit	Designation
0	Command Reject
1	Intervention Required
2	Bus-Out Check (Parity Error)
3	Not Used
4	Not Used
5	Not Used
6	Not Used
7	Not Used

Sense Conditions

Command Reject, Bit 0 - This bit indicates that the 1044 has received a command which it is not designed to execute.

Intervention Required, Bit 1 - This bit indicates that the last operation could not be executed because of a condition requiring some type of intervention at the analog computer. It is turned on when the analog computer is in the not-ready state.

Bus-Out Check (Parity Error), Bit 2 - This bit indicates that the 1044 has received a data byte or command code with incorrect parity. If the Bus-Out Check occurred during writing, incorrect data may have been sent to the 1044, but the operation is not terminated. If the parity error occurred on a command code, no operation is initiated.

Sense Lines

Command byte	Bit	0	1	2	3	4	5	6	7
		0	0	0	1	0	1	0	0

The 1044 can read data up to 128 Sense Lines in groups of 16 lines. The byte count of the CCW determines how many sense lines will be read. A byte count of two will cause two bytes of data to be transferred to memory with the condition of the first group of sixteen sense lines. Additional groups of sense lines are read by increasing the CCW byte count by two for each additional group of sense lines.

These sense lines are used to read digital information from the device to which they are connected. This may be the analog computer or a special hardware system. When they are connected to the analog computer, the sense lines are terminated on the logic patch panel (see Section 9).

Analog Computer Status

Command byte	Bit	0	1	2	3	4	5	6	7
		0	0	1	0	0	1	0	0

This command inputs information describing the status of the analog computer. The meaning of the analog computer status byte is described in Section 9.

Digital Output

A Digital Output operation is initiated by a Control command, which sets up the HSMPX subchannel to transfer data from main storage to the 1044. This operation may be performed on either HSMPX subchannel B or C.

The data transferred during this operation may be routed to several different destinations identified by the modifier bits in the CCW command byte. The devices which can be controlled are determined by

both the 1044 and the analog computer. The 1044 devices are described below and the analog computer devices are described in Section 9.

Analog Address System
Mode Control
Timer
Set Attenuators
Control Lines

Control Lines

Command byte	Bit	0	1	2	3	4	5	6	7
		0	0	0	1	1	1	1	1

The 1044 can write data for up to 128 Control Lines in groups of 16 lines. The byte count of the CCW determines how many control lines will be set. A byte count of two will cause two bytes of data to be transferred from memory to set the first group of sixteen control lines. Additional groups of control lines are set by increasing the CCW byte count by two for each additional group of control lines.

These control lines are used to send digital information to the device to which they are connected. This may be the analog computer or a special hardware system. When they are connected to the analog computer, the control lines are terminated on the logic patch panel (see Section 9).

Analog Computer Control Functions

Analog Address System

Command byte	Bit	0	1	2	3	4	5	6	7
		0	0	0	0	0	1	1	1

Mode Control

Command byte	Bit	0	1	2	3	4	5	6	7
		0	0	0	1	0	1	1	1

Timer

Command byte	Bit	0	1	2	3	4	5	6	7
		0	0	1	0	0	1	1	1

Set Attenuator

Command byte	Bit	0	1	2	3	4	5	6	7
		0	0	1	0	1	1	1	1

The above control functions are described in Section 9. The data format required by the analog computer and the response signals sent to the 1044 are described in detail.

5. INTERNAL PRIORITY OF THE 1044

The 1044 allows the operation of the units in the linkage to operate in the multiplex mode except when the program forces burst mode or when the operations require the use of the same subchannel.

Due to the high speed of the 1044 and the capability of full duplex data transfer, the 1044 operations are assigned a priority. Each subchannel has an equal priority within the 1044, allowing the subchannel serviced last to be serviced before the remaining subchannels. This insures that a high-speed data transfer on one subchannel will not "lock out" the other subchannels, when the HSMPX is operating in the multiplex mode.

The programmer need not be concerned with the internal priority of the operations of the 1044 as this is determined by the control logic of the 1044.

Programming for Channel Efficiency

The 1044 commands described in the preceding section provide the programmer with the ability to select the mode of data transfer on the HSMPX channel. The selection of the proper mode of data transfer will,

in many cases, speed the rate of operation significantly. The A/D conversion rate is a good example. If the operation is performed with the channel in burst mode, the conversion rate will be approximately 115 KHz per second. If the operation is performed with the channel in multiplex mode and no other I/O operations active, the data transfer rate will be approximately 90 KHz. If the operation is performed with the channel in multiplex mode and other I/O active (also in multiplex mode), the A/D conversion will be approximately 45 KHz per second. From this example, it can be seen how the mode of data transfer can affect the transfer rate. There are also functions performed by the 1044 in which the data transfer rate is not substantially affected by the mode of channel operation. These commands may be categorized as digital data transfer to and from the analog computer. The timing considerations for this type of operation limit the data transfer to fixed rates which are independent of the mode of channel data transfer. These rates are slower than could be achieved without the constraints placed on them by the analog. Since a Burst Mode command completely ties up the HSMPX channel, it is recommended that all Digital Input, Digital Output, and Direct Read Operations be performed only in Multiplex mode.

6. TERMINATING SEQUENCE

When the operation of the 1044 initiated by the START I/O instruction is terminated, the channel and the 1044 generate status conditions. These conditions can be brought to the attention of the program by an I/O interruption or by a TEST I/O. The status conditions, as well as an address and a count indicating the extent of the operation sequence, are presented to the program in the form of a channel status word.

The 1044 is accessible from more than one subchannel; therefore, the status conditions are indicated for the operation via the associated subchannel.

The channel status word (CSW) provides to the program the status of the 1044 or the conditions under which an I/O operation has been terminated. The CSW is placed in main storage at location 64 and is available to the program until the time the next I/O interruption occurs or until another I/O instruction causes its content to be replaced.

Bits 32-47 of the CSW identify the conditions in the 1044 and the channel that caused the storing of the CSW. The remaining fields in the CSW are described in the System/360 Principles of Operation.

1044 Unit Status Conditions

The status conditions represented by the 1044 status byte are as follows:

Bit	Designation
32	Not Used
33	Not Used
34	Not Used
35	Busy
36	Channel End
37	Device End
38	Unit Check
39	Unit Exception

Busy

Busy indicates that the 1044 cannot execute the command because it is executing a previously initiated operation or because it contains a pending interruption condition.

Device End

Device end is caused by the completion of an I/O operation at the 1044. The condition normally indicates that the 1044 has become available for another operation. Each I/O operation causes a device end condition,

and there is only one device end to an operation. The device end is not generated when any programming or equipment malfunction is detected during initiation of the operation. When command chaining takes place, only the device end of the last operation of the chain is made available to the program unless an unusual condition is detected during the initiation of a chained command, in which case the chain is terminated without the device end indication. The device end condition associated with an I/O operation to the 1044 is generated simultaneously with the channel end condition.

Channel End

Channel end is caused by the completion of the portion of an I/O operation involving transfer of data or control information between the I/O device and the channel. The condition indicates that the channel has become available for another operation.

Unit Check

Unit check indicates that the 1044 has detected an unusual condition that is described by the information available to a Sense command (1044 Sense byte). Unit check may indicate that a programming or an equipment error has been detected, that the not ready state of the 1044 has affected the execution of the command, or that an exceptional condition has occurred. The unit check bit provides a summary indication of the conditions specified by sense data. Termination of an operation with the unit check indication causes command chaining to be suppressed.

Unit Exception

Unit exception indicates that the 1044 has been sent a signal from the analog computer indicating that an unusual condition has occurred. It is used to indicate that a null failure has occurred during a pot set operation.

The 1044 treats the Unit Exception as a condition which precludes normal execution of the operation, and the Unit exception modifier is accompanied by Channel End and Device End.

The Unit Exception causes command chaining to be suppressed.

7. TEST PANEL

The 1044 Test Panel includes switches and indicators to monitor the operation of the unit when it is connected to the Model 44 channel and manually test the operation of the 1044 for preventive maintenance purposes. The switches are intended for use by maintenance personnel and should not be used by the programmer.

The following section describes the switches and indicators on the 1044 Test Panel.

Register Monitor

The Register Monitor switch is an eight-position rotary switch which has the following positions:

Digital Out	Digital In
Analog Out	Command 0
Analog In	Command 1
I/O Buffer	Command 2

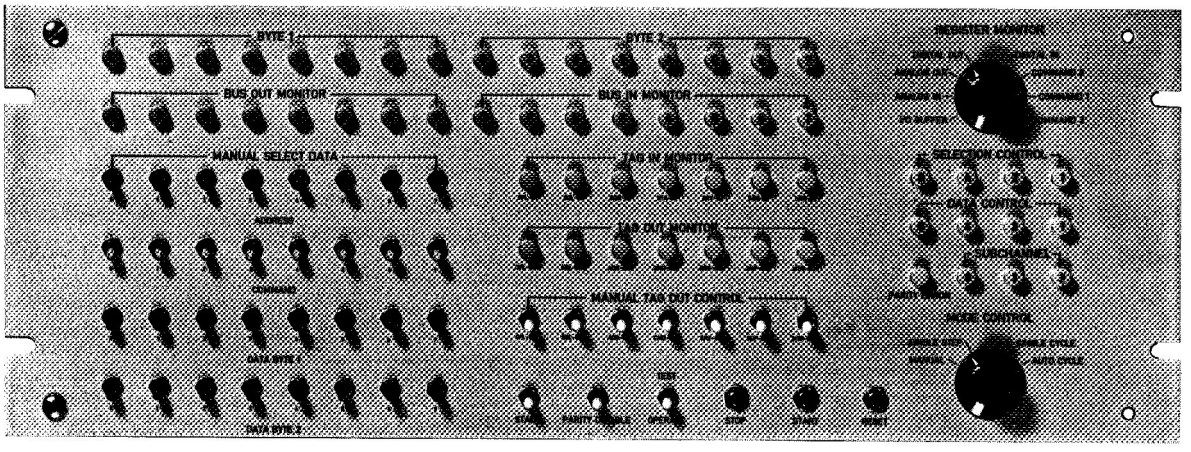
The switch positions of the Register Monitor are as follows:

Digital Out

When the Register Monitor switch is in the DIGITAL OUT position, the status of the 16-bit parallel bus to the analog computer interface is displayed in the BYTE 1 and BYTE 2 indicators on the test panel.

Digital In

When the Register Monitor switch is in the DIGITAL IN position, the status of the 16-bit parallel bus to the analog computer interface is displayed in the BYTE 1 and BYTE 2 indicators on the test panel.



TEST PANEL

Figure 2

Command 0

When the Register Monitor switch is in the COMMAND 0 position, the decoded command in the command register associated with subchannel 0 is displayed in the BYTE 1 and BYTE 2 indicators.

Command 1

When the Register Monitor switch is in the COMMAND 1 position, the decoded command in the command register associated with subchannel 1 is displayed in the BYTE 1 and BYTE 2 indicators.

Command 2

When the Register Monitor switch is in the COMMAND 2 position, the decoded command in the command register associated with subchannel 2 is displayed in the BYTE 1 and BYTE 2 indicators.

Analog Out

When the Register Monitor switch is in the ANALOG OUT position, the contents of the D/A Starting Address Register are displayed in BYTE 1 and the contents of the D/A Ending Address Register are displayed in BYTE 2.

Analog In

When the Register Monitor switch is in the ANALOG IN position, the contents of the A/D Starting Address Register are displayed in BYTE 1 and the contents of the A/D Ending Address Register are displayed in BYTE 2.

I/O Buffer

When the Register Monitor switch is in the I/O BUFFER position, the contents of the I/O buffer register are displayed in the BYTE 1 and BYTE 2 indicators. During an A/D operation, this will display the

value of the last conversion on the channel indicated by the Starting Address Register. During a D/A operation, this will display the value of the last conversion on the D/A converter indicated by the Starting Address Register.

Test/Operate

This is a two-position switch that enables the switches on the test panel. When this switch is in the TEST position, the 1044 is logically disconnected from the channel and switches on the test panel are enabled. When this switch is in the OPERATE position, the switches on the test panel are inhibited. The indicators remain active in either position.

Parity Disable

This switch disables the parity generator circuit in the 1044 and will turn on the PARITY ERROR indicator.

Stack

This switch will cause the selection control logic of the 1044 to generate a stack status condition for the Model 44 channel.

Reset

This is a momentary switch which will clear the logic of the 1044 to a reset condition. When the 1044 is connected to the Model 44, this is accomplished by the System Reset on the Model 44.

Start

This is a momentary switch which will cause the 1044 to operate for a period determined by the MODE CONTROL switch.

Stop

This is a momentary switch which will halt the operation of the 1044.

Mode Control

The Mode Control switch is a four-position rotary switch which has the following positions:

Single Step	Single Cycle
Manual	Auto Cycle

The switch positions of the MODE CONTROL switch are as follows:

Single Cycle

When the MODE CONTROL switch is in the SINGLE CYCLE position, pressing the START switch will cause the 1044 to execute one selection sequence to completion.

Single Step

When the MODE CONTROL switch is in the SINGLE STEP position, pressing the START switch will cause the 1044 to advance to the next logic state. This is used to step through the execution of a selection sequence.

Auto Cycle

When the MODE CONTROL switch is in the AUTO CYCLE position, pressing the START switch will cause the 1044 to continuously execute the command entered by the COMMAND switches on the test panel.

Manual

When the MODE CONTROL switch is in the MANUAL position, the MANUAL TAG OUT CONTROL switches on the test panel are enabled and can be used to manually simulate the channel initial selection sequence and operation.

Manual Tag Out Control

The Manual Tag Out Control includes seven switches to manually perform the channel tag line operations. The switches are as follows:

OPL-OUT	Operational Out
SEL-OUT	Select Out
ADR-OUT	Address Out
CMD-OUT	Command Out
SRV-OUT	Service Out
SUP-OUT	Suppress Out
MTR-OUT	Metering Out

The function of each of these tag lines is described in detail in the IBM System/360 I/O Interface Channel to Control Unit OEMI manual.

Tag Out Monitor

The Tag Out Monitor displays the status of the tag lines which are manually controlled by the corresponding tag out switches. The seven tag out indicators are the same as above for the Manual Tag Out Control.

Tag In Monitor

The Tag In Monitor displays the status of the tag lines which present information from the 1044 to the channel. The tag in indicators are as follows:

OPL-IN	Operational In
SEL-IN	Select In
ADR-IN	Address In
STA-IN	Status In
SRV-IN	Service In
REQ-IN	Request In
MTR-IN	Metering In

Bus Out Monitor

These eight indicators display the information being transferred to the 1044 from the channel. The parity line is not displayed.

Bus In Monitor

These eight indicators display the information being transferred to the channel from the 1044. The parity line is not displayed.

Byte 1 and Byte 2

The sixteen indicators display the information being transferred from/to the analog computer via the 16-bit bidirectional bus connected to the 1044. The information displayed in these indicators depends upon the setting of the REGISTER MONITOR switch as previously described.

Selection Control

These four indicators display the status of the selection control logic of the 1044.

Data Control

These four indicators display the status of the data control logic of the 1044.

Subchannel

These three indicators are turned on when the corresponding High Speed Multiplexor Subchannel is busy.

Parity Error

This display is turned on when the 1044 detects data with invalid parity.

Manual Select Data

The Manual Select Data switches are a group of 32 switches which are used to manually enter information which would be sent to the 1044 from the channel when the unit is in Operate mode. When the 1044 is in Test mode, these switches are enabled. The switches are grouped as follows:

Address

Command

Data Byte 1

Data Byte 2

Address

The Address switches are used to manually enter the address byte which, in Operate mode, is sent to the 1044 by the SIO instruction.

Command

The Command switches are used to manually enter the command to the 1044 which, in Operate mode, is sent from the command byte of the CCW.

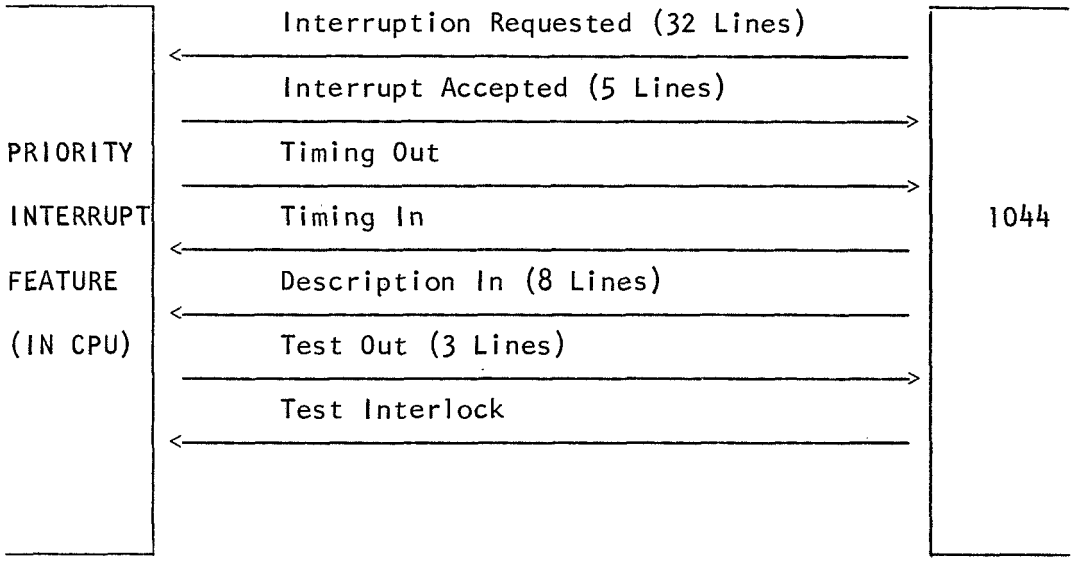
Data Byte 1 and Data Byte 2

These switches are used to manually send two bytes of data to the 1044.

8. PRIORITY INTERRUPTS

The 1044 includes 32 lines which are connected to the Priority Interrupt feature of the Model 44. There are 51 lines in the external interface of the priority interrupt feature as described in the Data Acquisition Special Features for the IBM System/360 Model 44.

A signal on any of the interruption-requested lines is taken as a request for a CPU intervention at the indicated level. Several requests may be pending at one time. The lines from the 1044 to the analog computer are usually terminated on the logic patch panel (see Section 9). None of the priority interrupt lines are required for use by the 1044 itself.

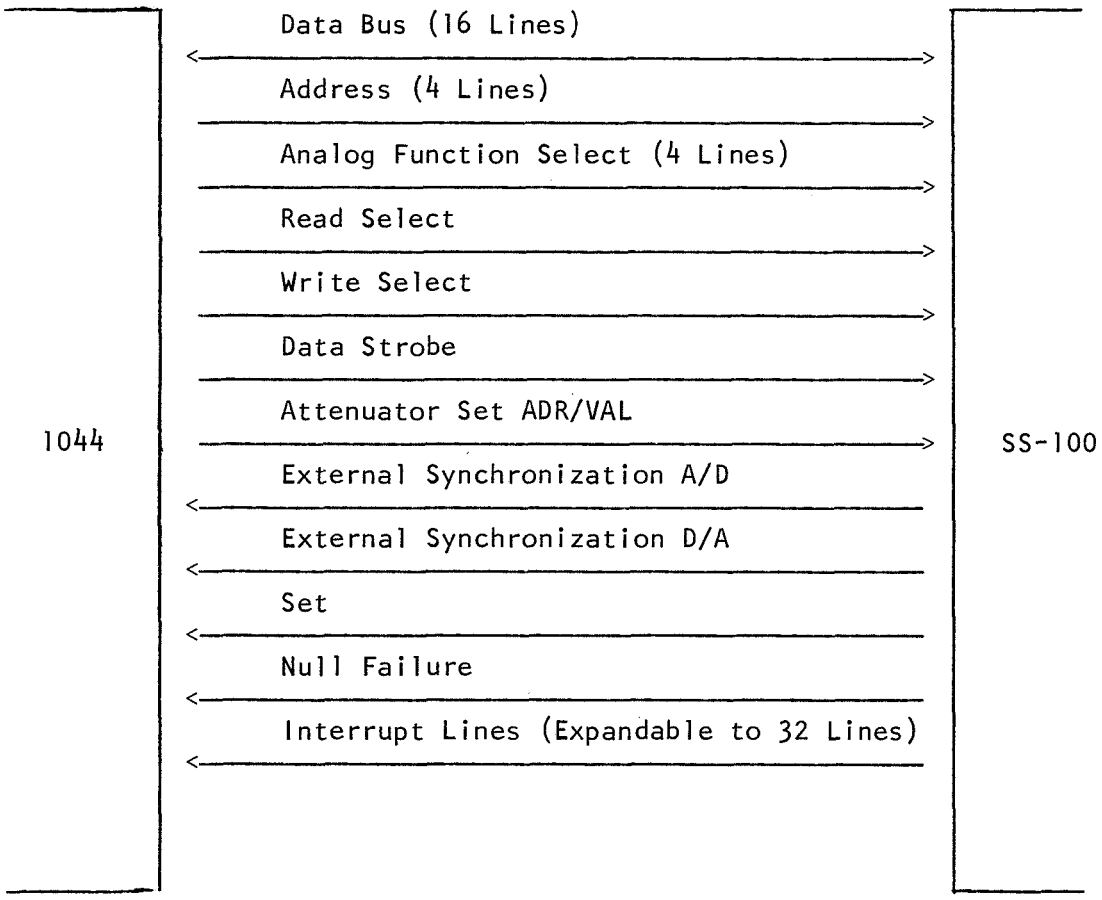


INTERRUPT LINES FOR PRIORITY INTERRUPT FEATURE

Figure 3

9. ANALOG COMPUTER INTERFACE

The 1044 is interfaced to the SS-100 by a control interface which includes all of the signal lines for sending control information to the analog computer. The interface includes a 16-bit bidirectional data bus, address and function lines. The lines in the control interface are shown below.



CONTROL INTERFACE FOR SS-100

Figure 4

The analog computer console is selected by the Address lines in the control interface. These lines address the console and enable the console to receive the data from the Data Bus.

The functions of the analog computer that are controlled from the 1044 are described below.

Analog Function Select

The four Analog Function Select lines and the Read Select and Write Select lines are used to identify the analog computer function which is being selected.

Analog Address System

Command byte	Bit	0	1	2	3	4	5	6	7
		0	0	0	0	0	1	1	1

The Analog Address System is selected by the above command. The 1044 will then send the desired address to the analog computer by the Data Bus. There is then a 10-millisecond delay for the address system to settle. The analog computer will then raise the External Synchronization A/D line. This signals the 1044 that the address system has settled and the desired component has been connected to the read-out bus. A Direct Read command to the 1044 is then executed to read the value of the addressed component.

The format of the address word sent to the Analog Address System is BCD. The address is four digits, with the first digit identifying the type of component, as follows:

Bit	0 1 2 3	Component
	0 0 0 0	A Amplifier
	0 0 0 1	P Attenuator
	0 0 1 0	M Quarter Square Multiplier
	0 0 1 1	T Trunk
	0 1 0 0	F Fixed Diode Function Generator
	0 1 0 1	X Multiple
	0 1 1 0	S/N Sample/Hold Amplifier
	0 1 1 1	S/C Sin/Cos Generator
	1 0 0 0	C Check (Initial Derivative)
	1 0 0 1	D/R Digital/Analog Multipliers

Address Format to address P115:

Bit 0			15
0001	0001	0001	0101
P	1	1	5

Mode Control

Command byte

Bit	0	1	2	3	4	5	6	7
	0	0	0	1	0	0	1	1

The Mode Control Command is used to set the mode of the analog computer from the 1044. In order to control the analog computer mode from the 1044, the mode switch on the analog computer console must be set to the REMOTE position.

The format of the mode commands sent to the Mode Control logic in the analog computer are as follows:

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Mode
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Static Test
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	Rate Test
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	Hold
	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	Operate
	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	Initial Condition
	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	Pot Set

Timer Control

Command byte

Bit	0	1	2	3	4	5	6	7
	0	0	1	0	0	1	1	1

The Master Timer and Auxiliary Timer are controlled from the 1044 by the Timer Control command.

After the Timer Control command is executed by the 1044, four half words are transmitted to either the master or auxiliary timer control. The format of the data sent to the timer is as follows:

Half Word	
1	Timer Control
2	IC Period
3	OP Period
4	HOLD Period

The Timer Control half word determines the following timer operating conditions:

Bit 0 - 2 Mode/3 Mode

0 - 2 Mode Operation

1 - 3 Mode Operation

Bit 1 - Master/Auxiliary

0 - Master Timer Control

1 - Auxiliary Timer Control

Bit 2 - Single Cycle/Continuous

0 - Single Cycle

1 - Continuous

Bit 3 and 4 - Time Scale Selection

00 - Factor 0.01

01 - Factor 0.1

10 - Factor 1.0

11 - Factor 10.0

The remaining three half words that are sent to the Timer determine the IC, OP, and HLD periods. The format of the half word is BCD.

Set Attenuator

Command byte	Bit	0	1	2	3	4	5	6	7
		0	0	1	0	1	1	1	1

The Set Attenuator command is used to set the Servo-Set Potentiometers or Digital Attenuators on the analog computer.

The format of the data word to set potentiometers is below.

Address	Value
0 — 15	16 — 31

The format of the address half word is as shown for the analog address system. The value is a BCD number between 0000 and 9999.

When the potentiometer control of the analog computer receives the address and value, it will cause the attenuator to set within 1.5 seconds. When the attenuator has been set, the analog computer will signal the 1044 by raising the SET line in the control interface. If the attenuator does not set within the allowed period of 1.5 seconds, the Null Failure line will be raised. This will also set the Unit Exception bit in the 1044 Unit Status byte.

Analog Computer Status

Command byte	Bit	0	1	2	3	4	5	6	7
		0	0	1	0	0	1	0	0

This command will cause the 1044 to read the SS-100 Status Information, indicating the condition of the analog computer. The SS-100 status conditions are as described below.

Bits 9-15 indicate the mode of the analog computer as follows:

Bit	9	Remote
	10	Pot Set
	11	Initial Condition
	12	Operate
	13	Hold
	14	Rate Test
	15	Static Test

Bit 8 - Indicates that the analog computer has received an Invalid Timer Control word.

Bit 7 - Indicates an overload condition in the analog computer.

Bits 4-6 - Indicate the mode of the digital logic in the analog computer as follows:

Bit	6	Reset
	5	Run
	4	Stop

Bits 0-3 are not used.

R E A D E R C O M M E N T S H E E T

Publication: Programming Reference Manual for the 1044 Hybrid Linkage Unit with the IBM System/360 Model 44, Publication 6030.2

FROM: Name _____

Business _____

Address _____

_____

Comments: (Describe errors, suggested additions or deletions)

Fold on dotted lines

FIRST CLASS
PERMIT NO.
11175
HOUSTON,
TEXAS

BUSINESS REPLY MAIL
No postage necessary if mailed in the United States

POSTAGE WILL BE PAID BY ADDRESSEE

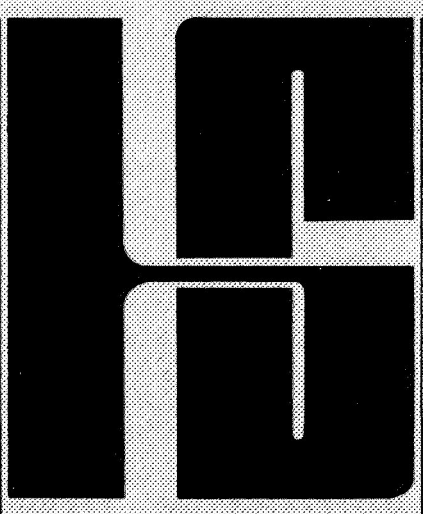
Hybrid Systems, Inc.

4301 DACOMA ROAD
HOUSTON, TEXAS 77018

Hybrid Systems, Inc.

4301 Dacoma Road
Houston, Texas 77018
(713) 681-3581

Hybrid Systems, Inc.



HYBRID EXECUTIVE

PROGRAMMING GUIDE

HYBRID EXECUTIVE

PROGRAMMING GUIDE

FOR

THE IBM SYSTEM/360 MODEL 44

THE HYBRID SYSTEMS 1044 HYBRID LINKAGE UNIT

THE HYBRID SYSTEMS SS-100 ANALOG/HYBRID COMPUTER

HYBRID SYSTEMS, INC.

A

SUBSIDIARY

OF

SCIENTIFIC RESOURCES CORPORATION

Publication 6033.2

10/68

Copyright, 1968, by Hybrid Systems, Inc., Houston, Texas.
Printed in the United States of America. This book, or parts thereof, may not be reproduced in any form without permission of Hybrid Systems, Inc.

PREFACE

This publication describes the Hybrid Executive written for use with a hybrid computer system which includes the Hybrid Systems 1044 Hybrid Linkage Unit, Hybrid Systems SS-100 Analog/Hybrid Computer, and the IBM System/360 Model 44 Computer. The Hybrid Executive is designed for use with the Data Acquisition Multiprogramming System (DAMPS) for the Model 44. Since the Hybrid Executive uses some of the facilities of DAMPS, this manual includes information on those parts of DAMPS which are necessary for the use of this program.

This publication is intended for use by the application programmer. For additional information, the user may refer to the following publications:

Hybrid Systems, Inc. 1044 Hybrid Linkage Unit - Reference Manual, Publication 6030

Hybrid Systems Hybrid Executive - Systems Guide, Publication 6034

IBM Data Acquisition Multiprogramming System:
System Description - Form H20-0467

CONTENTS

Section	Subject	Page
	Preface	i
1.	Introduction	1
2.	System Design Features of DAMPS	3
3.	DAMPS Conventions for a Real-Time Program	6
	IDENTRTJ - Identify Real Time Job	7
	FIDRTJ - Identify Fortran Real Time Job	7
	ATTACH - Assign Priority Interrupt	7
	CONVERT - Change Uncontrolled to Controlled	9
	ALLOW - - Allow Execution of CONVERT	10
	RETURN - Terminate a Real Time Task	10
	ENPILS - Enable Priority Interrupts	10
	DISPILS - Disable Priority Interrupts	11
	RTCANCEL - Cancel A Real Time Job	11
	EOTRTJ - Terminate a Real Time Job	12
	RTIO - Real Time Input/Output	12
	MASK - Mask I/O Channel Interrupts	12
	UNMASK - Unmask I/O Channel Interrupts	12
	FATACH - Assign Priority Interrupt, Fortran	12
	FRETRN - Terminate a Fortran Real Time Task	13
	FDSPIL - Disable Priority Interrupts, Fortran	14
	FENPIL - Enable Priority Interrupts, Fortran	14
	FQUEUE - Enter request for Foreground Task	15
	FDEQUE - Delete request for Foreground Task	16
	FEORTJ - Terminate a Fortran Real Time Job	17
	FCANCL - Cancel of Fortran Real Time Job	17
	FDONE - Terminate of Fortran Foreground Task	17
	FRCBSU - Build a Request Control Block, Fortran	17
	FRTIO - Fortran Real Time Input/Output	20
	FMSK - Mask I/O Channel Interrupts, Fortran	20
	FUNMSK - Unmask I/O Channel Interrupts, Fortran	21
	FCHECK - Check Real Time Input/Output	21

CONTENTS (Continued)

Section	Subject	Page
4.	Hybrid Executive Subroutines	23
	Programming Under DAMPS	23
	READAD - Analog/Digital Conversion, Sequential Addressing	26
	READAR - Analog Digital Conversion, Random Addressing	28
	RDMUX - Analog Multiplexer Address Initialization	30
	RDVAL - Analog/Digital Conversion	31
	WRITDA - Digital/Analog Conversion Sequential or Random Addressing	33
	WTMUX - Digital/Analog Address Initialization	37
	WTVAL - Digital/Analog Conversion	38
	SENSE - Read Sense Lines	39
	CONTRL - Write to Control Lines	40
	MODE - Analog Computer Mode Control	41
	ANALOG - Analog Computer Readout	42
	POTSS - Set Pontentiometers	44
	CCWCB - Build Channel Command Word	46
	PROBCK - Problem Checking	48

Appendix

A	Storage Requirements	A-1
B	Error Recovery Procedures and Messages	B-1

INTRODUCTION

The Hybrid Executive is a set of subroutines that generate channel programs for the transfer of information between the analog and digital computers in a hybrid system. The information is either control information or data used in the problem solution.

The subroutines allow the user to control the 1044 and the SS-100 by defining only a few basic parameters in a Fortran subroutine call statement. They are used to perform the following:

- Analog/Digital Conversion, Sequential Addressing
- Analog/Digital Conversion, Random Addressing
- Digital/Analog Conversion, Sequential Addressing
- Digital/Analog Conversion, Random Addressing
- Digital Input (Sense Lines)
- Digital Output (Control Lines)
- Analog Computer Mode Control
- Analog Computer Readout
- Potentiometer Set
- Problem Checking

In addition to the functions previously listed, the programmer can control additional hardware features of the Model 44 and the 1044 where applicable to an operation, including:

- Operation of the Model 44, High-Speed Multiplexor Channel in burst mode or multiplex mode
- Internal or external synchronization
- Internal control of the sample and hold amplifiers on the analog multiplexer inputs
- Digital/analog conversion with single or double buffering

The Hybrid Executive is designed for use on the IBM System/360 Model 44 with the minimum configuration required for the Data Acquisition Multiprogramming System (DAMPS). In addition, the system uses one High-Speed Multiplexor Channel with three subchannels for attachment of the 1044 Hybrid Linkage Unit. The Priority Interrupt Feature is used for notifying the CPU of an event in the analog computer; however, it is not required for the connection of the 1044.

DAMPS is an augmented version of the Model 44 Programming System which has been developed by IBM to extend the capabilities of the Model 44 for real-time scientific applications. The features of DAMPS are utilized by the Hybrid Executive. In particular, the DAMPS routines for responding to a priority interrupt are of major importance to the Hybrid Executive.

Due to the interrelationships of the Hybrid Executive and the Data Acquisition Multiprogramming System, a discussion on DAMPS is included in this publication. This information is included only for the convenience of the user, and no credit or responsibility for the DAMPS is intended.

SYSTEM DESIGN FEATURES OF DAMPS

The system provides the user with special subroutines to perform the following functions of prime importance in the hybrid system.

- Processing external priority interrupts
- Processing input/output on real-time channels with a channel scheduler
- Save and restore the status of the computer at interrupt time

The DAMPS subroutines and Hybrid Executive routines make additional capabilities available to the user of the Model 44 Programming System. These additional features are used by executing calls to the subroutines of DAMPS and the Hybrid Executive. All of the features of the Model 44 Programming System are available.

When running a program under DAMPS, a portion of core storage is allocated for use by the supervisor. The remaining core storage is available for use by the programmer for the Real Time Job. The Real Time Job may include one or more Real Time Tasks (RTT).

The basic elements of a DAMPS program are the Real Time Task (RTT), the Foreground Task (FGT), and the Real Time Job (RTJ). The RTT and FGT may be written either in Fortran or assembly language. The RTT is a user written routine, which receives control as a result of a priority interrupt. The RTT then relinquishes control by executing a supervisor call. While the RTT is in control, it can be interrupted only by a higher level priority. When the priority interrupt is accepted, its new Program Status Word (SW) takes effect and a latch is set, inhibiting all other priority interrupts except those with a priority level higher than that pending. The latch is reset by the supervisor call, after which the next lower level of processing resumes.

The use of the RTT is designed for processing which must be accomplished on a priority interrupt basis in response to external events in the analog computer. The analog computer has a number of holes on the logic patch panel which are assigned for use with priority interrupts. An RTT can be attached to the appropriate priority interrupt level to accomplish the

priority interrupt processing. The task associated with the highest level of priority will receive the CPU attention before others. The level of priority selected from an RTT is a function of the timing constraints on the function and must be determined by the programmer. Functions which do not require immediate response may use the sense lines on the analog computer to signal the CPU of an external event.

The FGT is a user written routine which is scheduled for execution by the supervisor, rather than receiving immediate attention, as the RTT. The FGT then relinquishes control by executing a supervisor call before the next FGT can be scheduled. The FGT is scheduled from a foreground queue. Entries in the queue may be generated by either an FGT or RTT. These requests are placed in the queue with a programmer assigned priority by use of a supervisor call and are processed in sequence when no priority interrupts are active.

The FGT is used for processing which need not be executed as a prompt response to an external event.

The Real Time Job is several Real Time Tasks and Foreground Tasks that are grouped together as an independent program.

When the RTJ has been loaded, the system functions on two levels; the priority interrupt level, and the foreground level. The operation of the system during each of these levels, and the transition between levels is described below.

When there are no active priority interrupts, the system is executing within the supervisor, executing an FGT, or waiting for a priority interrupt. The latter condition will arise only if there is no demand for processing at the foreground level. When a scheduled FGT is completed, control is returned to the supervisor and the FGT with the highest priority request in the queue is initiated.

When a priority interrupt has been accepted, the system is operating at the priority interrupt level. Control always operates on the interrupt at the highest level and returns to the next lower level. At the completion of the priority interrupt processing, control is returned to the routine which was interrupted by the original RTT.

The features of DAMPS that are utilized in the Hybrid Executive are the supervisor subroutines which control the execution of tasks. The Hybrid Executive subroutines generate input/output channel programs which are executed by the supervisor using the real time channel scheduler.

The supervisor subroutines are classified into groups according to the level at which they operate. The subroutines of interest to the hybrid programmer are summarized below.

Priority Interrupt Level Subroutines

- Save Interrupt Status. The status of the CPU is saved at the time of a priority interrupt which is associated with a Real Time Task (optional).
- Restore Interrupt Status. The status of the CPU is restored after processing of the priority interrupt has been completed (optional).
- Exit from RTT. Requests an exit from the RTT, allowing the supervisor to restore the system to its initial condition.
- Mask Channel. Masks the channel I/O interrupt.
- Unmask Channel. Unmasks the channel I/O interrupt.

Priority Interrupt or Foreground Level Subroutines

- Enable Priority Interrupt Level. Enables up to thrity-two priority interrupt levels. Pending interrupts on the indicated levels can optionally be cleared.
- Disable priority Interrupt Level. Disables up to thrity-two priority interrupt levels.
- Add to Queue. An entry point and its associated priority are added to the queue on the job's foreground list of sequential tasks.
- Delete from Queue. An entry point and its associated priority are deleted from the queue on the job's foreground list of sequential tasks.
- Attach a Real Time Task. The designated RTT becomes associated with the designated priority interrupt level.

Each RTT is assembled or compiled as a relocatable module. The modules are then link-edited into phases and placed in the phase library in absolute form as a job. At this point, all overlays have been converted into absolute code and are no longer relocatable.

The RTT is that portion of a Real Time Job which is activated in response to a priority interrupt. When a priority interrupt occurs, the old PSW is stored and a new PSW is used. A hardware latch is set at this time which inhibits a new priority interrupt at the same or lower level, but not at a higher level; therefore, allowing the RTT to be interrupted by a higher priority RTT. When the RTT is entered, all channels are masked in the new PSW, disabling all I/O interrupts. If the Save/Restore option is not used, the RTT must save the contents of any registers to be used, and any I/O is to be executed, the RTT must also save the status of the CPU, including the CAW, CSW, and IOPSW.

The channel scheduler can now be used by executing an input/output subroutine. When processing is completed and before the system is released by the LPSX, the RTT must restore the system status which it saved. The programmer may execute the RTT with the channels enabled or disabled. While enabled, channel interrupts from real time devices are processed at the priority level and others are stacked; otherwise all channel interrupts are processed during foreground time.

The RTT is used to respond to external events at a priority level. The RTT runs to completion as long as no higher priority interrupts occur. If a higher level priority interrupt occurs, control is returned to the interrupted RTT with the system status restored. As long as any priority interrupt is active, the system is operating in real time mode. Even though an RTT is operating at its priority level, it must allow for another interrupt unless it disables interrupts or is the highest priority interrupt level. The console typewriter must be assigned the address SYS000 before execution of the problem program.

3. DAMPS CONVENTIONS FOR A REAL TIME PROGRAM

This section includes a brief description of each of the DAMPS Supervisor Call routines and the FORTRAN or Assembly Language Subroutines that will

to be used with the Hybrid Executive subroutines. These routines are required either by DAMPS or the Hybrid Executive to perform real time input/output.

IDENTRTJ - Supervisor Call

The IDENTRTJ supervisor call is used to identify a real time job. Prior to the execution of this supervisor call, none of the DAMPS features may be used. An attempt to use a DAMPS supervisor call prior to the execution of IDENTRTJ will cause the job to be cancelled.

FIDRTJ - Fortran Subroutine

The FIDRTJ subroutine is used by the FORTRAN programmer to identify an RTJ. Prior to the execution of this subroutine, none of the DAMPS features can be used and attempt to use the DAMPS features will result in cancellation of the job. The subroutine is executed by a FORTRAN call statement as follows:

```
CALL  FIDRTJ
```

ATTACH - Supervisor Call

The ATTACH supervisor call is used to associate an RTT with a specific priority interrupt level. Subsequent priority interrupts at the given level will cause control to be transferred to the associated RTT. The ATTACH supervisor call does not enable the priority interrupt level; this is accomplished by the ENPILS supervisor call.

Three options of this supervisor call are available which provide for various levels of supervisor control at priority interrupt time. The user may have the supervisor perform all the necessary housekeeping at interrupt time in what is called a <u>controlled</u> RTT. Alternately, the user may have the RTT assume the responsibility for housekeeping in what is called an <u>uncontrolled</u> RTT. Or a user may convert an RTT from uncontrolled to controlled by use of the CONVERT supervisor call during the execution of the RTT.

<u>Controlled</u> Real Time Tasks receive control in the problem state with all necessary registers saved. <u>Uncontrolled</u> Real Time Tasks receive control in the supervisor state directly as a result of the priority interrupt with no registers saved. The uncontrolled RTT is totally responsible for

preserving the status of the system is not allowed to use any supervisor calls. However, the uncontrolled RTT can be converted to controlled during the priority interrupt by executing the CONVERT supervisor call.

If a CONVERT supervisor call is used, control is returned to the RTT in the problem state and all supervisor calls may be used. Subsequent priority interrupts will initiate the RTT in the uncontrolled mode.

When ATTACH is executed, general register 1 must contain the address of a parameter list. The list contains four words and is on a full-word boundary. Word 1 contains the absolute address of the entry point of the RTT. Word 2 of the list contains the absolute address of a 61-word save area which is on a full-word boundary. Word 3 contains two parameters: the first half of the word contains the hexadecimal representation of the priority interrupt level; the third byte contains an indication of the selected option, and the fourth byte is zero. Word 4 contains the absolute address of a four double-word save area for floating points registers. The last save area is optional. The format of the list is as follows:

Field	Contents
Word 1	Contains the absolute address of the entry point of the Real Time Task
Word 2	Contains the absolute address of a 61-word save area aligned on a full-word boundary
Word 3	
Bytes 1, 2	Contains the hexadecimal representation of a decimal number 0-31, representing the priority interrupt level to be attached.
Byte 3	Contains a hexadecimal indicator for the selection option, as follows:
	"00" - Indicates that the RTT is to be uncontrolled. The 61-word save area is not required if this option is selected.
	"0F" - Indicates that the RTT is to be initially uncontrolled but will be converted to a controlled task during execution.

DISPILS - Supervisor Call

The DISPILS supervisor call is used to disable priority interrupt levels. All the levels indicated in an individual call are disabled simultaneously.

When the DISPILS supervisor call is executed, general register 1 must contain the address of a parameter list. The parameter list will contain from 3 to 34 bytes depending on the number of priority interrupt levels to be disabled. Byte 1 of the list will contain either a hexadecimal "00" or "FF", and byte 2 will indicate the number of priority interrupt levels. The remaining bytes will each contain a number from 0 to 31 indicating the priority level to be disabled. The format of the parameter list is as follows:

Byte	Contents
1	Contains a hexadecimal "00" or "FF". Any other combination is invalid.
2	Contains a hexadecimal number indicating the number of priority interrupt levels to be disabled by the supervisor call. Any number greater than 32 or less than 1 is invalid.
3 - 34	One byte, containing the hexadecimal equivalent of a decimal number from 0 to 31 for each priority interrupt level to be disabled.

RTCANCEL - Supervisor Call

The RTCANCEL supervisor call is used for abnormal termination of an RTJ. EOTRTJ is used for normal termination. If an abort routine has been specified, it will be given control immediately after execution of RTCANCEL.

Following the abort routine, a system dump is provided if the abort routine returns control. Also, if the abort routine returned control, the supervisor will initiate loading the next job.

There is no parameter list associated with the RTCANCEL supervisor call. The RTCANCEL supervisor call is intended for use by any application program

which recognizes abnormal conditions during execution. If the abnormal condition resulted in a program check, the same procedure will be executed automatically.

EOTRTJ - Supervisor Call

The EOTRTJ supervisor call is used for normal termination of a Real Time Job. RTCANCEL is used for abnormal terminations. After EOTRTJ has been executed, control will be returned to the supervisor which will initiate loading the next job. Failure to end an RTJ in this manner may result in interference with later jobs. There is no parameter list associated with the EOTRTJ supervisor call.

RTIO - Supervisor Call

The RTIO supervisor call is used to execute input/output operations via the Real Time Channel Scheduler. When the RTIO supervisor call is executed, general register 1 must contain the address of the Request Control Block which defines the desired operation.

MASK/UNMASK - Supervisor Call

The MASK/UNMASK supervisor call is used to enable and disable I/O interrupts. When the MASK/UNMASK supervisor call is executed, general register 1 must contain the address of a byte which contains, in bit positions 0 - 2, the desired setting of the channel mask portion of the System Mask.

FATACH - Fortran Subroutine

The FATACH FORTRAN subroutine is used to associate an RTT with a specific priority interrupt level. Subsequent priority interrupts at that level will cause control to be transferred to the associated RTT. The RTT is terminated by use of the FRETRAN FORTRAN subroutine. The FATACH subroutine does not enable the priority interrupt level; this is accomplished by the FENPIL FORTRAN subroutine.

The FATACH subroutine is executed by a FORTRAN call statement with the following format:

```
CALL FATACH (RTT, RTSAVE, N, FPSAVE)
```

The parameter list is described below:

RTT - The name of the Real Time Task to be attached. The RTT must have been compiled as a separate FORTRAN subroutine. The name must appear in a FORTRAN EXTERN statement in the subroutine where the attach is executed.

RTSAVE - The name of an array containing a 61-word save area. A different save area must be established for each RTT, and it may not be used for any other purpose.

N - An integer from 0-31 indicating the priority interrupt level to be associated with the RTT.

FPSAVE - The name of an array containing a four double-word save area for floating point registers. A different area must be established for each RTT, and it may not be used for any other purpose.

The attached RTT is in asynchronous operation. Parameters may not be passed while the routine is in control and the save area may not be located in COMMON.

FRETRAN - Fortran subroutine

The FRETRAN FORTRAN subroutine is used to terminate a FORTRAN Real Time Task. Control is returned to the next lower RTT, when one is active. In the event that none are active, control is returned to foreground processing. Use of the FRETRN subroutine does not preclude the use of the END statement in a FORTRAN subroutine which is to be an RTT.

The FRETRN subroutine is executed by a FORTRAN call statement with the following format:

```
CALL FRETRN
```

The subroutine has no parameter list.

FDSPIL - Fortran Subroutine

The FDSPIL FORTRAN Subroutine is used to disable priority interrupt levels. All the levels indicated in an individual call are disabled simultaneously.

The FDSPIL subroutine is executed by a FORTRAN call statement with the following format:

```
CALL FDSPIL (1, J1, J2, ... Jn)
```

The parameter list is described below:

I	-	Is an integer zero.
Jn	-	Each Jn is an integer from 0 to 31 indicating a specific priority interrupt level to be disabled. There will be one Jn for each level to be disabled.

FENPIL - Fortran Subroutine

The FENPIL subroutine is used to enable priority interrupt levels. All the levels indicated in an individual call are enabled simultaneously. Pending interrupts on the indicated levels will be cleared if the reset option is specified.

The FENPIL subroutine is executed by a FORTRAN call statement with the following format:

```
CALL FENPIL (1, J1, J2, ... Jn)
```

The parameter list is described below:

I	-	Is an integer indicating the selection of the reset option. If I - 255, the selected priority interrupts will have pending interrupts reset at the time of enabling. If I = 0, the pending interrupts will not be reset.
Jn	-	Each Jn is an integer from 0 to 31 indicating a specific priority interrupt level to be enabled. There will be one Jn for each level to be enabled.

FQUEUE - FORTRAN Subroutine

The FQUEUE FORTRAN subroutine is used to enter a request for execution of a Foreground Task, with an associated priority, into the Foreground Queue. The requests in the Foreground Queue are honored in priority sequence.

The FQUEUE FORTRAN subroutine is executed by a FORTRAN call statement with the following format:

```
CALL FQUEUE (IP1, NAME, IRC, IP2)
```

The parameter list is described below:

IP1 - An integer from 0 - 127 representing the priority of the foreground task. All other values are invalid.

NAME - The name of the Foreground Task being queued. The name must also appear in an EXTERNAL statement, and the Foreground Task must have been compiled as a FORTRAN subroutine.

IRC - An integer variable which will contain the completion code returned after completion of the subroutine. The value is encoded as follows:

IRC Value	Meaning
IRC = 0	Indicates the queue request was successfully placed in the Foreground Queue.
IRC = 4	Indicates that the address of the parameter list was invalid.
IRC = 15	Indicates that an invalid priority was contained in IP1.
IRC = 240	Indicates that the name of the Foreground Task was invalid. The name may not have appeared in an EXTERNAL statement.
IRC = 255	Indicates that the Foreground Queue was full.
IP2	An integer variable which can be used to pass a variable to the queued subroutine.

Only if IRC is equal to 0 was the foreground task queued. Any invalid conditions not indicated above will cause the job to be cancelled.

FDEQUE - FORTRAN Subroutine

The FDEQUE FORTRAN subroutine is used to delete a previous request for execution of a Foreground Task. The unqueue request, which is actually placed in the Foregroung Queue, is honored just prior to the next foreground scheduling. A queue request is identified on the basis the name of the Foreground Task and the priority of the request. When the unqueue request is honored, all requests with the same name and priority will be deleted.

The FDEQUE FORTRAN subroutine is executed by a FORTRAN Call statement with the following format:

```
CALL FDEQUE (IP1, NAME, IRC)
```

The parameter list is described below:

IP1 - An integer from 0 - 127 representing the priority of the Foreground Task. All other values are invalid.

NAME - The name of the Foreground Task to be deleted from the queue. The name must also appear in an EXTERNAL statement.

IRC - An integer variable containing the completion code re-returned after completion of the subroutine. The return code is encoded as follows:

IRC Value	Meaning
IRC = 0	Indicates that the dequeue request was successfully placed in the queue.
IRC = 4	Indicates that the address of the parameter list was invalid.
IRC = 15	Indicates that an invalid priority was contained in IP1.
IRC = 240	Indicates that the name of the Foreground Task was invalid. The name may not have appeared in an EXTERNAL Statement.
IRC = 255	Indicates that the Foreground Queue was full.

Only if IRC is equal to 0 was the request queued. Any invalid condition other than those outlined above will cause the job to be cancelled.

FEORTJ - Fortran Subroutine

The FEORTJ subroutine is used for normal termination of a Real Time Job. After FEORTJ has been executed, control will be returned to the supervisor which will initiate loading the next job. Failure to end an RTJ in this manner may result in interference with subsequent jobs. The FEORTJ is executed by a FORTRAN call statement with the following format:

```
CALL FEORTJ
```

FCANCL - Fortran Subroutine

The FCANCL subroutine is used for abnormal termination of a Real Time Job. If the DUMP option was indicated in the job cards for the run, a storage dump will be provided. The subroutine is executed by a FORTRAN call statement with the following format:

```
CALL FCANCL
```

FDONE - Fortran Subroutine

The FDONE subroutine is used to terminate a FORTRAN Foreground Task. Control is returned to the supervisor and the FGT with the highest priority in the Foreground Queue is initiated. Where multiple requests of identical priorities are concerned, the oldest entry will be accepted first. Use of The FDONE subroutine does not preclude use of the END statement in a FORTRAN subroutine which is to be an FGT. The FDONE subroutine is executed by a FORTRAN call statement with the following format:

```
CALL FDONE
```

FRCBSU - Fortran Subroutine

The FRCBSU subroutine is used to build a Request Control Block. A Request Control Block is required for each request for real time I/O, and serves as the communication link between the program and the channel scheduler. The Request Control Block can be used by only one I/O request at a time. The

contents of the block must not be changed during execution; however, if the parameters for subsequent I/O operations are identical, the RCB can be reused.

Several options for I/O operations may be designated in the RCB. The associated transaction may be queued if the device is busy, or the user may wish to have it cancelled. The user may wish to treat the associated request as a burst operation. A request may also be initiated for the queueing of a certain foreground task upon successful completion of the I/O. The FRCBSU subroutine is executed by a FORTRAN call statement with the following format:

```
CALL FRCBSU (RCB, IUNIT1, CCWA1, IOPT1, IFGTA, NAME, IP1, IP2,
             IUNIT2, CCWA2, IOPT2, IUNIT3, CCWA3, IOPT3)
```

The parameter list is described below:

RCB		The name of the area where the RCB will be constructed and maintained. The area must be 8 full words for each RCB requested.
IUNIT1	-	An integer code for the symbolic unit.
CCWA1	-	The name of the area where the channel program will be located. This area will also be referenced in a call to the appropriate subroutine for construction of the channel program.
IOPT1	-	An integer indicating one or more of the following options; if more than one option is being described, the value is the sum of the applicable codes. If none are applicable, the parameter should be omitted.

IOPT1	=	16	(not used with the Hybrid Executive)
IPTO1	=	8	The I/O request is not to be queued by the channel if the facility is busy.
IOPT1	=	4	A second RCB is requested, for which additional parameters are required as described below. When this option is selected, eight additional full words of storage are used by the RCB.

IOPT1 = 2 The I/O request is to be a burst operation.

IOPT1 = 1 An FGR is to be queued when the I/O is completed. Additional parameters are required as described below.

The next four parameters are included only when IOPT1 is an odd integer indicating FGT queueing.

IFGTA - The name of a three full-word area where a parameter list will be constructed for queueing the FGT.

NAME - The name of the entry point of the FGT.

IP1 - An integer from 0 to 127 representing the priority of the FGT.

IP2 - An integer variable which can be used to pass a variable to the queued FGT.

The next two parameters are required if the multiple RCB option is indicated in IOPT1. The third parameter is optional unless a third RCB is requested.

IUNIT2 - An integer code for the symbolic unit.

CCWA2 - The name of the area where the channel program will be located. This area will also be referenced in a call to the appropriate subroutine for construction of the channel program.

IOPT2 - An integer code indicating the selected options. If none of the options are applicable, this parameter should be omitted.

IOPT2 = 4 Additional RCB's are requested for which additional parameters are required. When this option is selected, eight additional words of storage are used by the RCB.

IOPT2 = 2 The input/output request is to be treated as a burst operation.

IOPT2 = 6 Both of the above options.

IUNIT3 - An integer code for the symbolic unit.

CCWA3 - The name of the area where the channel program will be located. This area will also be referenced in a call to the appropriate subroutine for construction of the channel program.

IOPT3 - An integer code indicating an option for burst mode operation. If the option is not used, the parameter should be omitted.

IOPT3 = 2 The I/O request is a burst mode operation.

FRTIO - Fortran subroutine

The FRTIO subroutine is used to execute real time I/O. The FRTIO subroutine is executed by a FORTRAN call statement in the following format:

```
CALL FRTIO (RCB, IRET)
```

The parameter list is described below:

RCB - The address of the RCB which was established by a previous call FRCBSU.

IRET An integer variable where one of the following two codes will be placed by the FRTIO subroutine. This parameter is required only if parameter IOPT1 in the call FRCBSU statement indicated that the I/O request was not to be queued.

IRET = 0 The I/O operation was initiated.

IRET = 8 The I/O operation was not initiated because the unit or channel was busy.

FMSK - Fortran Subroutine

The FMSK subroutine is used to mask one or more channels during execution of a Real Time Task. The FMSK subroutine is executed by a FORTRAN call with the following format:

```
CALL FMSK  (RCB1, RCB2, RCB3)
```

The entries in the parameter list reference a previously established RCB. The channels to which the referenced devices are attached will be disabled and unreferenced channels will be enabled. From one to three RCB's may be specified.

FUNMSK - Fortran subroutine

The FUNMSK subroutine is used to unmask a channel during execution of a Real Time Task. The FUNMSK subroutine is executed by a FORTRAN call with the following format:

CALL FUNMSK (RCB1, RCB2, RCB3)

The entries in the parameter list reference previously established Request Control Blocks. The channels to which the indicated devices are attached will be enabled and unreferenced channels will be disabled. From one to three RCB's may be specified.

FCHECK - Fortran subroutine

The FCHECK subroutine is used to check the progress of a real time I/O operation. The FCHECK subroutine is executed by a FORTRAN call statement in the following format:

CALL FCHECK (RCB, IRET, IOPT)

The entries in the parameter list are defined as follows:

RCB	-	The name of the area containing the Request Control Block established with an FRCBSU subroutine call.
IRET	-	An integer variable where one of the following return codes will be placed by the FCHECK subroutine.
IRET	=	0 The input/output operation came to a normal completion.
IRET	=	1 A unit exception was indicated in the ending status of the input/output operation.
IRET	=	4 The input/output operation has not been completed.
IRET	=	8 The input/output operation was not started. A busy condition was encountered when the operation was requested and parameter IOPT1 in the FRCBSU call indicated that the request was not to be queued.

IOPT - An integer code equal to 1, designating that the FCHECK subroutine is to wait for completion of the operation before returning control to the calling program. If IOPT = 0, the subroutine will return to the main program upon completion.

4. HYBRID EXECUTIVE SUBROUTINES

The Hybrid Executive consists of subroutines that generate channel programs and then use the DAMPS real time channel scheduler to execute the input/output. The subroutines may be used either with a Real Time Task or Foreground Task.

The Hybrid Executive subroutines may be used according to two general DAMPS classifications, either Real Time Tasks or Foreground Tasks.

Programming Under DAMPS

Foreground tasks are executed in the problem state. The entry point to an FGT is defined when the FGT is queued and the FGT is completed when the DONE supervisor call is executed. In order for the FGT to receive control, a request must be placed in the foreground queue by execution of the QUEUE supervisor call which can be executed by an FGT or RTT. A task priority and a parameter value, consisting of one full word, may be specified when the task is queued.

If the foreground queue is empty when an RTT queues and FGT, the FGT will receive control immediately after the system returns to foreground level. If the queue is not empty, each entry is placed in the queue and all processing continues unchanged. When the FGT issues a DONE supervisor call, the entire foreground queue is examined. If an "unqueue" entry is found, all matching entries which were executed prior to the UNQUEUE are deleted. After the entire queue is processed, the request with the highest priority is given control. If two or more FGT's are found with equal priority, the oldest entry is given precedence. Once the FGT is given control, it will continue until it issues a DONE supervisor call. At that time, the entire queue will be examined to account for any additional FGT's which may have been queued and control is again given to the FGT with the highest priority.

Real Time Tasks are classified as either a controlled or uncontrolled Real Time Task. The primary difference between the two types of Real Time Tasks is in saving the system status and therefore, the response time to a priority interrupt.

The entry point of a controlled RTT is specified as a parameter to the ATTACH supervisor call routine when the RTT is attached to a specific Priority Interrupt Level (PIL). Completion of the RTT is accomplished by execution of the RETURN supervisor call which returns control to the point of interruption. Once the priority interrupt level has been enabled, the RTT will be given control every time the interrupt occurs. The RTT will run to completion unless it is interrupted by a higher priority level. In this case, the interrupting RTT completes and returns to the point of interruption as though the interrupt has not occurred. A controlled RTT executes instructions in the problem state and cannot use the Model 44 Programming System supervisor services.

When the controlled RTT is given control, all registers and DAMPS facilities are available because the system has stored all registers and system status.

Uncontrolled Real Time Tasks are further classified into two types as explained below.

1. The uncontrolled RTT which never uses the DAMPS supervisor state. Control is returned to the point of interruption by execution of the LPSX instruction.
2. The uncontrolled RTT which receives direct control in the supervisor state, and can convert to a controlled RTT in problem state in order to use the DAMPS supervisor. Control is returned to the point of interruption by execution of the RETURN supervisor call.

Uncontrolled RTT's receive control directly with all registers and status unchanged. Before the RTT can change the contents of any register, it must save the contents of that register. When the uncontrolled RTT is completed, it must restore all registers and system status to the condition at the time the interrupt occurred, and then execute the LPSX instruction for a unique location defined for each priority interrupt level.

Each analog computer in the system is considered a separate unit and has a unique unit address. The unit address is set when the system is generated and is identified to DAMPS as one of the parameters in the FRCBSU subroutine.

This section gives a detailed discussion of each subroutine which is included in the Hybrid Executive. The error messages, storage requirements, and execution times for each subroutine are given in the appendixes.

READAD

The READAD subroutine generates a channel program to perform analog/digital conversion with sequential addressing of the analog multiplexer input channels. The actual execution of the I/O operation is performed by the FRTIO subroutine.

The READAD subroutine is executed by a FORTRAN call statement with the following format:

CALL READAD (RSCCW, N, ICON, LOCA)

The parameter list is described below:

RSCCW - Address of the area where the CCW's will be constructed. It is two double words and must be on a double-word boundary.

N - The number of analog multiplexer channels what will be sequentially addressed and converted.

ICON - An integer that indicated the options that may apply to this operation. If more than one option is selected, the value of ICON should be the sum of the codes for each option.

ICON = 0 - The operation is to be performed with the High-Speed Multiplexor channel operating in multiplex mode, internal synchronization of the conversion rate and with no control of the sample and hold amplifiers by the 1044.

ICON = 8 - Causes the High-Speed Multiplexor Channel to operate in burst mode.

ICON = 2 - Causes the analog/digital conversion rate to be controlled by an external synchronization pulse. This signal will originate from a designated termination on the logic patch panel of the analog computer.

ICON = 1 - Enables the 1044 Hybrid Linkage Unit to control the mode of the sample and hold amplifiers. When the analog multiplexer address is equal to the value of the Starting Address Register

- in the 1044, the sample and hold amplifiers will be switched to "HOLD" mode. When the analog multiplexer address is equal to the value of the Ending Address Register in the 1044, the sample and hold amplifiers will be switched to "TRACK" mode.

LOCA - The name of the area where the data from the analog/digital converter will be stored, beginning with LOCA (2). The first two bytes in LOCA contain the value to be sent to the 1044 Starting Address Register and Ending Address Register. This specifies the beginning and ending analog multiplexer channels for the sequential analog/digital conversion. They must be integers between 0 and 255 and the value of the Starting Address Register must be less than the value of the Ending Address Register. The remaining area of LOCA is reserved for storing data from the analog/digital converter. One half-word is used for each converter value. LOCA is defined at INTEGER*2.

The addresses can be conveniently set by a FORTRAN statement, as follows:

```
LOCA  (1) = (SA*256 + EA)
```

If an incorrect value of ICON is recognized, an error message is printed on the console typewriter. A new value for ICON may then be entered from the console. If ICON has a negative value, this routine becomes NO-OPT.

READAR

The READAR subroutine generates a channel program to perform analog/digital conversion, with random addressing of the analog multiplexer input channels. This subroutine uses two high-speed multiplexer subchannel: one to output addresses to the analog multiplexer, and another to input data from the analog/digital converter. The actual execution of the I/O operation is performed by the FRTIO subroutine.

The READAR subroutine is executed by a FORTRAN call statement with the following format:

CALL READAR (RRCCW, N, ICON, LOCA, LOCB)

The parameter list is described below:

RRCCW - Address of the area where the CCW's will be constructed. It is two double words and must be on a double-word boundary.

N - The number of analog multiplexer channels that will be addressed in random mode and converted.

ICON - An integer that indicates the options that may apply to this operation. If more than one option is selected, the value of ICON should be the sum of the codes for each option.

ICON = 4 - In order for the operation to be executed properly by the 1044, the value of ICON must always be equal to 4 or greater. If ICON = 4, the random analog/digital conversion will be performed with the High-Speed Multiplexor Channel operating in multiplex mode, internal synchronization of the conversion rate and with no control of the sample and hold amplifiers by the 1044.

ICON = 2 - Causes the analog/digital conversion rate to be controlled by an external synchronization pulse. This signal will originate from a designated termination on the logic patch panel of the analog computer. NOTE: If this option is selected, two is added to the base value of ICON = 4.

ICON = 1 - Enables the 1044 Hybrid Linkage Unit to control the mode of the sample and hold amplifiers. When the analog multiplexer address is equal to the comparison address, the sample and hold amplifiers will be set to "TRACK" mode. If these addresses are not equal, the sample and hold amplifiers are set to "HOLD" mode.

LOCA - The name of the area where the analog multiplexer addresses are stored. It must be defined by a type statement as INTEGER*2. The first byte contains an integer which is the analog multiplexer channel address of the channel that will be be converted. The second byte contains an integer which is a comparison address for sample and hold amplifier control, as discussed above. If no control of the sample and hold amplifiers by the 1044 is desired, the comparison address is not used.

LOCB - The name of the area where the data from the analog/digital converter will be stored. It must be defined by a type statement as INTEGER*2.

If an incorrect value of ICON is recognized, an error message is printed on the console typewriter. A new value for ICON may then be entered from the console. If ICON has a negative value, this routine becomes NO-OPT.

Chained RCB's are required for proper execution.

RDMUX

The RDMUX subroutine generates a channel program to initialize the 1044 for a sequential mode analog/digital conversion. This subroutine causes two bytes of data to be transmitted to the 1044 which are loaded into the Starting Address Register and the Ending Address Register. The subroutine is intended for use with the RDVAL subroutine to perform the sequential analog/digital conversion. The actual execution of the I/O operation is performed by the FRTIO subroutine.

The RDMUX subroutine is executed by a FORTRAN call statement with the following format:

```
CALL RDMUX (RDCCW, INT, LAST)
```

The parameter list is described below:

RDCCW - Address of the area where the CCW will be constructed. It is a double word and must be on a double-word boundary.

INT - An integer that indicates the value to be transmitted to the 1044 and loaded into the Starting Address Register. It must be defined by a type statement as INTEGER*2.

LAST - An integer that indicates the value to be transmitted to the 1044 and loaded into the Ending Address Register. It must be defined by a type statement as INTEGER*2.

Both integers must be a value between 0 and 255 and the value of INT must be less than the value of LAST. This routine is not required for use with the READAD subroutine.

RDVAL

The RDVAL subroutine generates a channel program to perform a sequential analog/digital conversion on the analog multiplexer input channels, initialized previously by the RDMUX subroutine. The actual execution of the I/O operation is performed by the FRTIO subroutine.

The RDVAL subroutine is executed by a FORTRAN call statement with the following format:

```
CALL RDVAL (DSCCW, ICON, N, LOCA)
```

The parameter list is described below:

DSCCW - Address of the area where the CCW will be constructed. It is a double word and must be on a double-word boundary.

N - The number of analog multiplexer channels that will be addressed in a sequential mode and converted.

ICON - An integer that indicates the options that may apply to this operation. If more than one option is selected, the value of ICON should be the sum of the codes for each option.

ICON = 0 - The operation is to be performed with the high-speed multiplexor channel operating in multiplex mode, internal synchronization of the conversion rate and with no control of the sample and hold amplifiers by the 1044.

ICON = 8 - Causes the high-speed multiplexor channel to operate in burst mode.

ICON = 2 - Causes the analog/digital conversion rate to be controlled by an external synchronization pulse. This signal will originate from a designated termination on the logic patch panel of the analog computer.

ICON = 1 - Enables the 1044 Hybrid Linkage Unit to control the mode of the sample and hold amplifiers. When the analog multiplexer address is equal to the value of the Starting Address Register in the 1044, the sample and hold amplifiers will be switched to "TRACK" mode.

LOCA - The name of the area where the data from the analog/digital converter will be stored. It must be defined as INTEGER*2.

If an incorrect value of ICON is recognized, an error message is printed on the console typewriter.

WRITDA

The WRITDA subroutine generates a channel program to perform digital/analog conversion with either sequential or random addressing of the digital/analog converters. The actual execution of the I/O operation is performed by the FRTIO subroutine.

The WRITDA subroutine is executed by a FORTRAN call statement with the following format:

CALL WRITDA (WTCCW, ICON, NW, LOCA)

The parameter list is descirbed below:

WTCCW - The address of the area where the CCW's will be constructed. It is a two double words and must be on a double-word boundary.

NW - The number of digital/analog converters that will be sequentially or randomly addressed and converted.

ICON - An integer that indicates the options that may apply to this operation. If more than one option is selected, the value of ICON should be the sum of the codes for each option.

ICON = 0 - The operation is to be performed with the High-Speed Multiplexor Channel operating in multiplex mode, sequential addressing of the digital/analog converters, internal synchronization of the conversion rate and with no double buffering of the converter input data.

ICON = 8 - Causes the High-Speed Multiplexor Channel to operate in burst mode.

ICON = 4 - Indicates that the digital/analog converters will be addressed in random mode. Note that the data tables are organized differently for sequential and random addressing.

ICON = 2 - Causes the digital/analog conversion rate to be controlled by an external synchronization pulse. This signal will originate from a designated termination on the logic patch panel of the analog computer.

ICON = 1 - Causes the digital/analog conversion to be performed with double buffering of the input to the converters. Double buffering operates as follows when sequential addressing is used:

Each time a word is transferred from the 1044, it is loaded into the input buffer register of the addressed converter. When a comparison is made with the value of the Ending Address Register, a jam transfer occurs from the input buffer register to the conversion control register and the conversion is initiated simultaneously on all channels.

When random addressing is used, double buffering operates as follows:

The converter address is the first byte of the address word, which is compared with the comparison address in the second byte of the address word. If the addresses are equal, the jam transfer occurs from the input buffer register to the conversion control register and the conversion is initiated simultaneously on all channels. If the addresses are not equal, the data is transferred to the input buffer register and the conversion is not initiated.

LOCA - The name of the area where the data table is stored in memory. The format of the data table is different for sequential and random mode addressing as described below. It is defined by a type statement as INTEGER*2.

In the sequential addressing mode, the first two bytes of LOCA contain the value to be sent to the 1044 Starting Address Register and Ending Address Register for digital/analog conversion. These values specify the beginning and

ending digital/analog converters address for the sequential conversion. They must be an integer between 0 and 255 and the value of the Starting Address Register must be less than the value of the Ending Address Register. The remaining area of LOCA is reserved for the data which will be transferred to the digital/analog converters. The area for LOCA must be equal to NW + 1.

The addresses can be conveniently set by a FORTRAN statement, as follows:

```
LOCA (1) = (SA*256 + EA)  SA and EA are integers
```

Data Table for Sequential Mode Digital/Analog Conversion

Byte 0	Byte 1
SAR	EAR
DATA (1)	
DATA (2)	
:	
DATA (NW)	

In the random addressing mode, the data and converter addresses are stored in alternating half-words. The first half-word contains the address of the converter in the first byte and a comparison and address in the second byte. If double buffering is not used, the comparison address is not required. The next half-word contains the data word for the digital/analog converter. The area for LOCA must be equal to 2NW.

Data Table for Random Mode Digital/Analog Conversion

Byte 0	Byte 1
A1	CA
DATA (1)	
A2	CA
DATA (2)	
:	
ANW	CA
DATA (NW)	

If an incorrect value of ICON is recognized, an error message is printed on the conscle typewriter. A new value for ICON may then be entered from the console.

Chained RCB's are required for proper execution (random addressing).

WTMUX

The WTMUX subroutine generates a channel program to initialize the 1044 for a sequential mode digital/analog conversion. This subroutine causes two bytes of data to be transmitted to the 1044 which are loaded into the Starting Address Register and the Ending Address Register. The subroutine is intended for use with the WTVAL subroutine to perform the sequential digital/analog conversion. The actual execution of the I/O operation is performed by the FRTIO subroutine.

The WTMUX subroutine is executed by a FORTRAN call statement with the following format:

```
CALL WTMUX (WRCCW, INT, LAST)
```

The parameter list is described below:

WRCCW - Address of the area where the CCW will be constructed. It is a double word and must be on a double-word boundary.

INT - In integer that indicates the value to be transmitted to the 1044 and loaded into the Starting Address Register. It must be defined by a type statement as INTEGER*2.

LAST - An integer that indicates the value to be transmitted to the 1044 and loaded into the Ending Address Register. It must be defined by a type statement as INTEGER*2.

Both integers must be a value between 0 and 255 and the value of INT must be less than the value of LAST. This routine is not required for use with the WRITDA subroutine.

WTVAL

The WTVAL subroutine generates a channel program to perform a sequential digital/analog conversion of the digital/analog converters initialized perviously by the WRMUX subroutine. The actual execution of the I/O operation is performed by the FRTIO subroutine.

The WTVAL subroutine is executed by a FORTRAN call statement with the following format:

```
CALL WTVAL (WSCCW, ICON, N, LOCA)
```

The parameter list is described below:

WSCCW - The address of the area where the CCW will be constructed. It is a double word and must be on a double-word boundary.

N - The number of digital/analog converters that will be sequentially addressed and converted.

ICON - An integer that indicates the options that may apply to this operation. If more than one option is selected, the value of ICON should be the sum of codes for each option.

ICON = 0 - The operation is to be performed with the high-speed multiplexor channel operating in multiplex mode, sequential addressing, internal synchronization and with no double buffering of the converter input data.

ICON = 8 - Causes the high-speed multiplexor channel to operate in burst mode.

ICON = 4 - Invalid.

ICON = 2 - Causes the digital/analog conversion rate to be controlled by an external synchronization pulse.

ICON = 1 - Causes the digital/analog conversion to be performed with double buffering of the input to the converters. Refer to the WRITDA subroutine description for additional information.

SENSE

The SENSE subroutine generates a channel program to read the sense lines in the 1044 which are terminated on the logic patch panel of the analog computer. The actual execution of the I/O operation is performed by the FRTIO subroutine.

The SENSE subroutine is executed by a FORTRAN call statement with the following format:

```
CALL SENSE (SCECCW, LOC, NB)
```

The parameter list is described below:

SCECCW - Address of the area where the CCW will be constructed. It is a double word and must be on a double-word boundary.

LOC - The name of the area where the data from the sense lines will be stored. The first two bytes contain the condition of the first group of sixteen sense lines. It is defined by a type statement as INTEGER*2.

NB - The number of bytes of sense information read by this operation. The number of bytes must always be equal to an even integer, as the sense lines are read in groups of sixteen lines.

CONTRL

The CONTRL subroutine generates a channel program to transfer data to the control lines in the 1044 which are terminated on the logic patch panel of the analog computer. The actual execution of the I/O operation is performed by the FRTIO subroutine.

The CONTRL subroutine is executed by a FORTRAN call statement with the following format:

```
CALL CONTRL (CONCCW, LOC, NB)
```

The parameter list is described below:

CONCCW - Address of the area where the CCW will be constructed. It is a double word and must be on a double-word boundary.

LOC - The name of the area where the data to be transferred to the control lines will be stored. The first two bytes contain the condition that will set the first group of sixteen control lines. It is defined by a type statement as INTEGER*2.

NB - The number of bytes of control information to be transferred by this operation. The number of bytes must always be equal to an even integer as the control lines are set in groups of sixteen lines.

MODE

The MODE subroutine generates a channel program that will issue a mode control command to the analog computer. The actual execution of the I/O operation is performed by the FRTIO subroutine.

The MODE subroutine is executed by a FORTRAN call statement with the following format:

```
CALL MODE (MODCCW, M)
```

The parameter list is described below:

MODCCW	-	Address of the area where the CCW will be constructed. It is a double word and must be on a double-word boundary.
M	-	An integer that identifies the mode of the analog computer. It is defined by a type statement as INTEGER*2.
M = 17	-	Static Test (ST)
M = 10	-	Rate Test (RT)
M = 4	-	Hold (HD)
M = 8	-	Operate (OP)
M = 16	-	Initial Condition (IC)
M = 32	-	Potentiometer Set (PS)

If M is not equal to any of the above values, this subroutine sets the mode of Initial Condition and types an error message on the console typewriter.

ANALOG

The ANALOG subroutine generates a channel program to perform readout of components in the addressed analog computer. The operation is performed by the use of two high-speed multiplexor subchannels. One subchannel transfers the address of the component to the address system of the analog computer and the other subchannel issues a direct read command to read out the value of the addressed component. The actual execution of the I/O operation is performed by the FRTIO subroutine.

The ANALOG subroutine is executed by a FORTRAN call statement with the following format:

```
CALL ANALOG   (ANACCW, N, NAME, LOCA)
```

The parameter list is described below:

ANACCW - Address of the area where the CCW's will be constructed. It is two double words and must be on a double-word boundary.

N - An integer that specifies the number of analog components that will be addressed and read out.

NAME - The name of an array that contains the analog component addresses. The names of the analog components are stored in A4 format. The address is four characters, with the first digit identifying the type of components, followed by a three-digit number. If the component number is less than three digits, the leading zeros must be included (blanks are not valid). The component identifications are as follows:

A Amplifier
P Attenuator
M Quarter Square Multiplier
T Trunk
F Fixed Diode Function Generator
X Multiple

If one of the potentiometers does not null properly, a signal is sent to the 1044. This causes an error message to be typed on the console typewriter giving the pot address. The subroutine will then continue with execution and attempt to set the remainder of the pots in the list. When the subroutine has completed execution, it will cause a message to be typed on the console typewriter indicating the pots that the operator should attempt to set manually. When this task is complete, the operator presses EOB and the job continues.

PSRCB - Address of the RCB setup by FRCBSU that POTSS will use. This RCB should reference subchannel B of the High-Speed Multiplexor Channel.

CCWCB

The CCWCB subroutine is an assembler language subroutine used by each of the subroutines in the Hybrid Executive to generate the channel programs. The parameters in the calling sequence are the inputs to the subroutine that are decoded into the required fields to construct the Channel Command Words (CCW).

The CCWCB subroutine is executed by a FORTRAN call statement with the following format:

```
CALL CCWCB (CCWADR, COMBTE, OPCODE, ARRAY, COUNT)
```

The parameter list is described below:

CCWADR - The address of the area where the CCW will be constructed. It is a double word and must be on a double-word boundary.

COMBTE - The command code which specifies the command to be executed. Bits 0 - 7 of the CCW. It is defined by a type statement as INTEGER*2.

OPCODE - The flag code which specifies the use of the flag bits in the CCW. It is defined by a type statement as INTEGER*2.

If more than one option is selected, the value of OPCODE should be the sum of the codes for each option.

Bits 32 - 36 of the CCW are defined as follows:

OPCODE = 16 (Bit 32) Chain-Data Flag

OPCODE = 8 (Bit 33) Chain-Command Flag

OPCODE = 4 (Bit 34) Supress-Length-Indication

OPCODE = 2 (Bit 35) Skip Flag

OPCODE = 1 (Bit 36) Programmed-Controlled-Interruption

ARRAY - The data address field of the CCW which specifies the address of the first location of the area designated by the CCW.

COUNT - The count field of the CCW which specifies the number of byte locations in the storage area designated by the CCW.

This subroutine is a service routine used by the other Hybrid Executive subroutines. It may be used for generating a channel program to perform a special function not included in the system. No checking for valid parameter definition is done within this routine.

PROBCK

The PROBCK subroutine performs a check of the wiring of a problem on the analog patch panel and checks the static accuracy of the analog computer. The program performs calculations on a series of parameter groups representing the patching configuration on the analog patch panel and then compares these results to a computed value. The input signals, the computed value, the actual value read from the analog computer, and the difference between the calculated and actual value are then printed on the line printer. Patching errors or a component failure are readily detected by this technique.

The PROBCK subroutine may be stored in relocatable form and called by another FORTRAN subroutine. It requires the use of the card reader and printer, in addition to the 1044 and is designed to operate under DAMPS. The program which calls PROBCK must perform the DAMPS initialization by calling FIDRTJ.

The input to PROBCK is via the card reader. Each card may contain up to four parameter groups where one parameter group identifies an operation to PROBCK. A number of parameter groups identifies the patching configuration being checked. The parameter group includes the following:

1. Analog component address, COMVAL
2. Digital constant multiplier, FAC
3. Integer number indicating the operation to be performed, ID

The PROBCK program performs the operation specified by the identifier, ID, on the value read from the addressed analog component, COMVAL, and then multiplies by the value specified by the multiplier, FAC. The value calculated for each parameter group is summed and compared to the value of the last component in the group (which represents the left side of the equation in FORTRAN notation).

Each time a calculation is performed, an error constant for the specified type of device is added to the value of the maximum allowable error. When all of the calculations for a set of parameter groups have been performed,

the maximum allowable error is compared with the difference between the calculated value and the actual value. The results are printed on the line printer.

The component names in COMVAL are the same as used with subroutine ANALOG, summarized below:

A Amplifier
P Attenuator
M Quarter Square Multiplier
T Trunk
F Fixed Diode Function Generator
X Multiple
N Sin/Cos Generator
S Sample/Hold Amplifier
C Check (Initial Derivative)
D Digital/Analog Multiplier

The integer number, ID, which indicates the operation to be performed, causes the execution of one of the following routines:

ID = 0 Calculate the produce of COMVAL*FAC
ID = 1 Calculate the $\log_{10}$ of COMVAL
ID = 2 Calculate the Sin of COMVAL
ID = 3 Calculate the Arcsin of COMVAL
ID = 4 Calculate the square root of COMVAL
ID = 5 Calculate the Cos of COMVAL
ID = 6 Calculate the Arccos of COMVAL
ID = 7 Calculate the exponetial of COMVAL
ID = 8 Calculate the product of two sequential values of COMVAL, X + Y
ID = 9 Calculate the quotient of two sequential values of COMVAL, X/Y.

The functions performed by each of the routines specified by ID are as follows:

ID = 1 COMVAL (M) = ALOG10 (ABS(COMVAL (M)))
Maximum error factor = 40 millivolts

ID = 2 COMVAL (M) = SIN (COMVAL(M))
Maximum error factor = 40 millivolts

ID = 3 COMVAL (M) = ARSIN (COMVAL(M))
Maximum error factor = 80 millivolts

ID = 4 COMVAL (M) = SQRT (ABS(COMVAL(M)))/10
Maximum error factor = 50 millivolts

ID = 5 COMVAL (M) = COS (COMVAL(M))
Maximum error factor = 40 millivolts

ID = 6 COMVAL (M) = ARCOS (COMVAL(M))
Maximum error factor = 80 millivolts

ID = 7 COMVAL (M) = EXP (COMVAL(M) x 2303)/50
Maximum error factor = 100 millivolts

ID = 8 COMVAL (M+1) = (COMVAL (M) x COMVAL (M+1)/100
Maximum error factor = 50 millivolts

ID = 9 COMVAL (M+1) = COMVAL (M)/COMVAL (M+1))/100
Maximum error factor = 100 millivolts

The card format is as shown below:

Column		
1 - 4	Component name, COMVAL	Parameter group 1
5 - 11	Digital Modifier, FAC	
12 - 13	Digital routine, ID	
14 - 18	Not used	
19 = 22	Component name	Parameter group 2
23 - 29	Digital Modifier	
30 - 31	Digital routine	
32 - 36	Not used	
37 - 40	Component name	Parameter group 3
41 - 47	Digital Modifier	
48 - 49	Digital routine	
50 - 54	Not used	
55 - 58	Component name	Parameter group 4
59 - 65	Digital Modifier	
66 - 67	Digital routine	
68 - 73	Not used	
74	Card Type, 1 designates more data cards follow, 0 designates last card	

COMVAL - Four characters to identify the analog component address (A4 format)

FAC - Seven digits including sign and decimal point for digital modifier (F7.2 format)

ID - Two-digit integer between 0 and 9 designating the operation to be performed (I2 format).

APPENDIX A - STORAGE REQUIREMENTS

Appendix A contains the storage requirements for the Hybrid Executive subroutines. The figures given do not include any additional library subprograms or DAMPS subroutines that the subroutine uses during execution.

ROUTINE	NUMBER OF BYTES HEXADECIMAL	NUMBER OF BYTES DECIMAL
MODE	2C8	712
READAD	3D4	980
READAR	3B0	844
CONTRL	1F0	396
SENSE	1F0	396
ANALOG	620	1570
WRITDA	478	1144
RDVAL	354	852
WTVAL	354	852
RDMUX	218	540
WTMUX	218	540
POTSS	E10	3590
CCWCB	D8	216
PROBCK	104	260

APPENDIX B - ERROR RECOVERY PROCEDURES AND MESSAGES

Appendix B contains a listing of all of the error messages given by the Hybrid Executive subroutines with a brief discussion of each message. No DAMPS error messages are included.

ROUTINE	MESSAGE/RECOVERY
MODE	WARNING. . .INVALID MODE CODE. MODE DEFAULTS TO IC. No operator action.
READAD	RANDOM MODE NOT VALID IN READAD. ENTER ICON FROM CONSOLE . . . EOB. Enter new ICON as four digits. EOB.
READAR	SEQUENTIAL ORDER OR BURST MODE NOT VALID IN READAR ENTER ICON FROM CONSOLE . . . EOB. Enter new ICON as four digits. EOB.
CONTRL	No message.
SENSE	No message.
ANALOG	No message.
WRITDA	ENTER ICON FROM CONSOLE . . . EOB. Enter new ICON as four digits. EOB.
RDVAL	ICON = ID DENOTES RANDOM. INVALID IN RDVAL. Enter new ICON as four digits. EOB.
WTVAL	ICON = ID DENOTES RANDOM, INVALID IN WTVAL. Enter new ICON as four digits. EOB.
RDMUX	No message.
WTMUX	No message.
CCWCB	No message.

PROBCK ENTER CONSOLE ID. 0 OR 1.
Enter 0 or 1 from keyboard. EOB.

ENTER MODE CODE AS IN HYBRID PROGRAM LIBRARY.
Enter two digits from keyboard. EOB.

16 = Initial Condition Mode (IC)

32 = Pot Set Mode (PS)

The mode entered will be the mode the components are read in.

POTSS VALUE FOR POT XXXX INVALID
REENTER VALUE . . . EOB. Enter value as four digits.

POT ID XXXX INVALID REENTER POT IO.
Enter pot ID as four characters.

SET MODE TO POT SET IS NOT BEING DONE
PAUSE 0. Analog console is not responding to the digital command.

SET ABOVE INDICATED POTS BY HAND, THEN . . . EOB.
Set POTS then EOB.

RESIDUAL COUNT = 0
The last pot has failed to null.

Note: Analog channel addresses are <u>not</u> checked for validity in the routines READAD, READAR & WRITDA.

R E A D E R C O M M E N T S H E E T

Publication: Hybrid Executive, Programming Guide, Publication 6033.2

FROM: Name _____

Company _____

Address _____

Comments: (Describe errors, suggested additions or deletions)

FIRST CLASS
PERMIT NO.
11175
HOUSTON,
TEXAS

BUSINESS REPLY MAIL
No postage necessary if mailed in the United States

POSTAGE WILL BE PAID BY ADDRESSEE

Hybrid Systems, Inc.

4301 DACOMA ROAD
HOUSTON, TEXAS 77018

Hybrid Systems, Inc.

A SUBSIDIARY OF SCIENTIFIC RESOURCES CORPORATION

4301 Dacoma Road
Houston, Texas 77018
(713) 681-3581

ANASET

ANASET is a program set which may be used to statically setup and checkout an analog computer. The program set is basically FORTRAN and operates under the S/360 Model 44 DAMPS monitor system.

INTRODUCTION

ANASET is a FORTRAN program which operates under the Model 44 DAMPS system. ANASET may be used to set up and statically check a problem wired for the SS-100 analog console.

ANASET's input is a set of statements describing the analog elements and their connectives. Potentiometer settings, and other variables such as scaling factors, etc. may also be input to accomplish the following.

1. Compute POT settings as functions of scale factors, problem input variables, etc.
2. Calculate the theoretically correct static check values for the analog problem.
3. Set the POTS on the SS-100 to the proper value while monitoring for any difficulties.
4. Set the SS-100's multiplying DACS to the desired values for static check.
5. Perform the static check, reading values from the SS-100 into the program's arrays.
6. Use these analog values to determine if the theoretically correct static values have been equaled.
7. List the elements' theoretical values, their values calculated using the analog data, and their actual analog values.

INPUT DATA

The user must describe the SS-100 analog wiring to ANASET in terms of statements equating analog components outputs to functions of their respective inputs. For ANASET's use the analog elements are input in the alphameric form

Xn (mm)

where the X represents an amplifier, potentiometer, DFG, etc., n denotes the SS-100 sector where the component is located, and mm denotes the particular element of the sector.

ANASET recognizes the following SS-100 elements

P0 (nn)	$01 \leq nn \leq 26$	POTENTIOMETERS
P1 (nn)	$01 \leq nn \leq 26$	
P2 (nn)	$01 \leq nn \leq 26$	
P3 (nn)	$01 \leq nn \leq 24$	
P4 (nn)	$01 \leq nn \leq 15$	
P5 (nn)	$01 \leq nn \leq 15$	

M0 (nn)	$01 \leq nn \leq 6$	MULTIPLIERS
M1 (nn)	$01 \leq nn \leq 6$	
M2 (nn)	$01 \leq nn \leq 6$	
M3 (nn)	$01 \leq nn \leq 6$	
M4 (nn)	$01 \leq nn \leq 3$	
M5 (nn)	$01 \leq nn \leq 3$	

A0 (nn)	$01 \leq nn \leq 29$	AMPLIFIERS (INTEGRATOR/SUMMERS, SUMMERS, INVERTERS)
A1 (nn)	$01 \leq nn \leq 29$	
A2 (nn)	$01 \leq nn \leq 29$	
A3 (nn)	$01 \leq nn \leq 31$	
A4 (nn)	$01 \leq nn \leq 32$	
A5 (nn)	$01 \leq nn \leq 32$	
DA0 (nn)	$01 \leq nn \leq 15$	INTEGRATOR/SUMMERS (DERIVATIVE INPUTS)
DA1 (nn)	$01 \leq nn \leq 15$	
DA2 (nn)	$01 \leq nn \leq 15$	
DA3 (nn)	$01 \leq nn \leq 15$	
DA4 (nn)	$01 \leq nn \leq 09$	
DA5 (nn)	$01 \leq nn \leq 09$	
DX (nn)	$01 \leq nn \leq 16$	DIGITAL/ANALOG CONVERTERS (DIGITAL INPUTS)
VG0 (nn)	$01 \leq nn \leq 03$	VARIABLE DIODE FUNCTION GENERATORS
VG1 (nn)	$01 \leq nn \leq 03$	
VG2 (nn)	$01 \leq nn \leq 03$	
VG3 (nn)	$01 \leq nn \leq 03$	
VG4 (nn)	$01 \leq nn \leq 10$	
VG5 (nn)	$01 \leq nn \leq 10$	

SC4 (nn)	$01 \leq nn \leq 42$	SIN/COS GENERATORS
SC5 (nn)	$01 \leq nn \leq 42$	
SH (nn)	$01 \leq nn \leq 16$	SAMPLE/HOLD AMPLIFIERS
DR (nn)	$01 \leq nn \leq 16$	DIGITAL/ANALOG CONVERTERS
RM4 (nn)	$01 \leq nn \leq 19$	RESOLVER MULTIPLIERS
RM5 (nn)	$01 \leq nn \leq 19$	

ANASET recognizes every addressable element of the SS-100 with the exception of trunks, multiples, and reference power supplies.

ANASET provides routines which facilitate the description of the analog problem in terms of non-linearities. In addition any routine of the FORTRAN Library may be used.

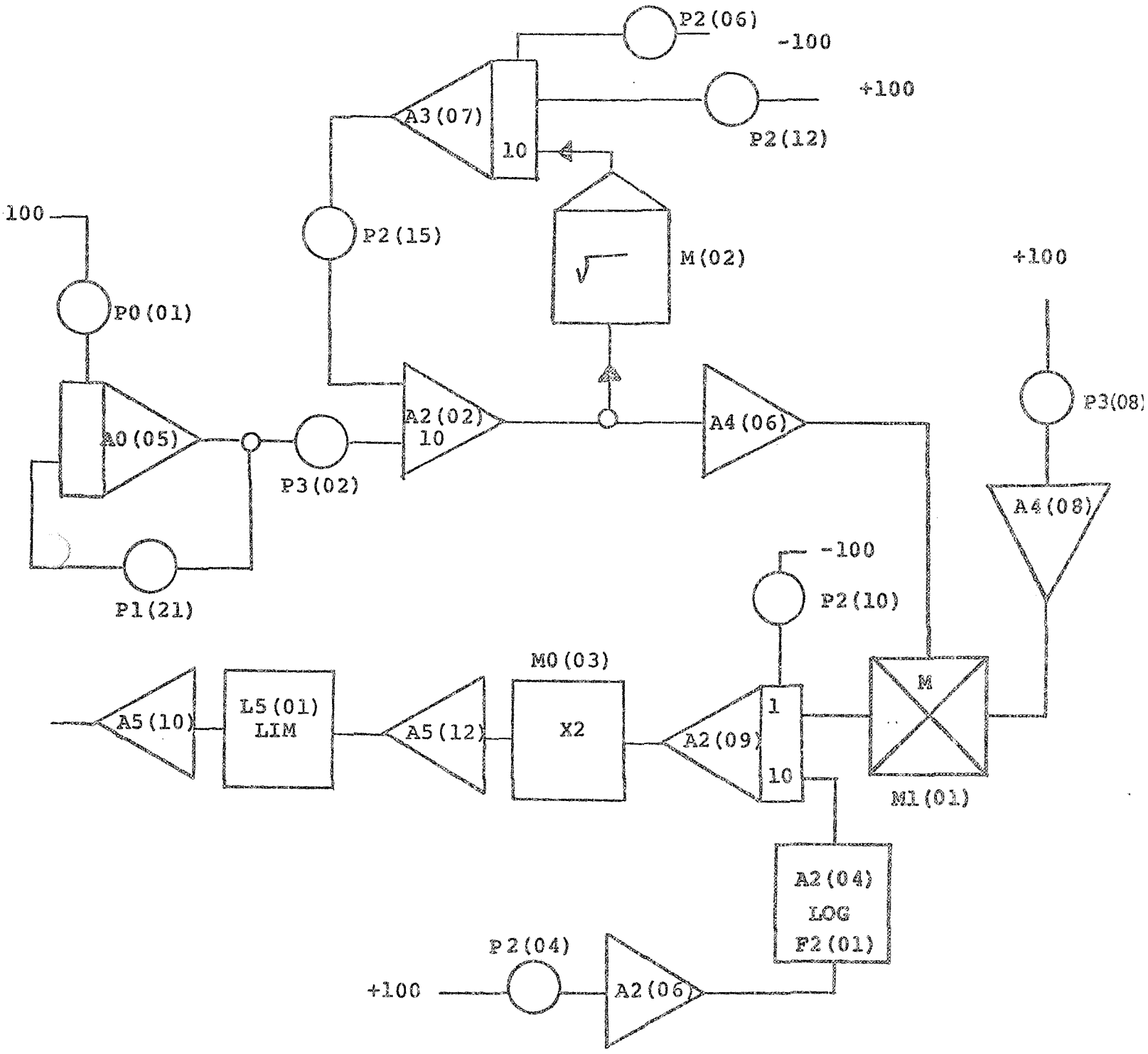
For Example: - SIN, COS, LOG, SQUARE, LIMITER (ELIMT), VARIABLE FUNCTION GENERATOR (VFGEN), RESOLVER (RESLVR).

ANASET will accept statements describing the interconnection of the analog elements within limitations imposed by FORTRAN arithmetic statement construction plus the following:

ALL ELEMENTS APPEARING ON THE LEFT HAND SIDE OF THE EQUAL SIGNS MUST BE PREFIXED WITH AN S EXCEPT DAn (mm) AND Pn (mm). DAn (mm) MAY APPEAR ONLY ON THE LEFT HAND SIDE OF THE EQUAL SIGN. Pn (mm) MAY APPEAR ON EITHER SIDE.

EXAMPLE

For purposes of illustrating the input statements the analog diagram shown below will be prepared for ANASET input.



The ANASET statements for the diagram would be:

```
SFAC      =  1.663
P0(01)    =  .5326
P1(21)    =  SFAC* .1500
P2(04)    =  .5000
P3(02)    =  .1250
P2(15)    =  .8000*SFAC/2.0
P2(12)    =  .5000
P2(06)    =  .3333
P3(08)    =  SQRT (SFAC*P1(21) + .1250)
P2(10)    =  .1000
SA0(05)   =  -(100. * P0(01))
DA0(05)   =  -(A0(05) *P1(21))/10.0
SA3(07)   =  -(-100.0 * P2(06))
SA2(02)   =  -(A0(05) *P3(02) *10.0 + A3(07) *P2(15))
SM0(02)   =  10.*SQRT (ABS(A2(02)))
DA3(07)   =  -(-10.0 * M(02) + 100. *P2(12))/10.
SA4(06)   =  -(A2(02))
SA4(08)   =  -(100.0 * P3(08))
SM1(01)   =  -(A4(08) * A4(06))/100.0
SA2(06)   =  -50.
DA2(09)   =  -(10.0 * A2(04) + M1(01))/10.0
SA2(09)   =  -(-100. *P2(10))
SA5(12)   =  (A2(09) **2)/100.
SA5(10)   =  -(ELIMT (A5(12), -5.0, +50.0))
SA2(04)   =  50. * ALOG 10(-A2(06))
```

NOTES:

1. The minus signs used on the right hand side denote the inversion accomplished when the input signals pass through the analog amplifier.

2. The right hand side for DAn (mm), shows inversion plus division by 10. This is necessary because the SS-100 measures the derivative inputs using the output of a special amplifier with a gain of 1/10.

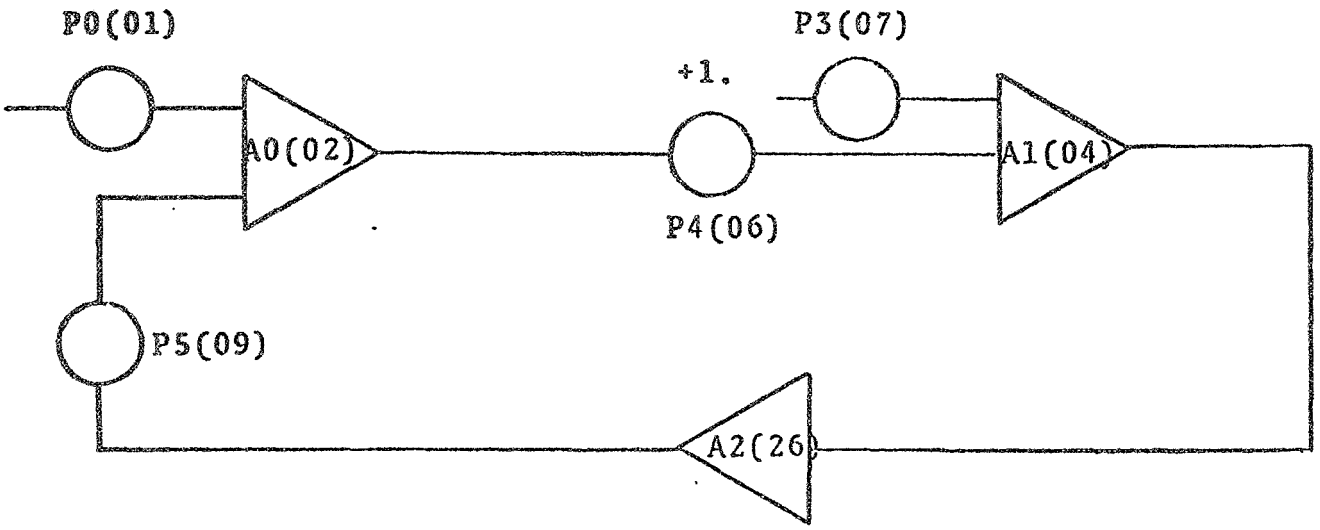
3. ELIMT is a function subroutine of ANASET used to compute outputs based on input signal using a set of given breakpoints.

4. Because ANASET statements are procedural in the FORTRAN sense, right hand side terms must in someway be defined before the left hand side can be calculated. The integrator's presence in most analog loops make statement preparation straight forward since their static value is equivalent to their initial condition value, e.g.

```
SA4(08) = -(100.0 *P3(08))
```

In some cases algebraic loops will be present which require an iterative process to seek a static value for all loop elements. Note 5 discusses the treatment of these loops.

5. Certain analog diagrams will contain algebraic loops such as



*Required

Here the statements for ANASET would be

1 SA0(02) = -(P0(01) + A2(26) * P5(09))

2 SA1(04) =-(P4(06) * AO(02) + P3(07))

3 SA2(26) = -(A1(04))

This situation requires ANASET to compute the output of AO(02) based on A2(26) which is not calculated itself until statement 3. To compute a theoretical value for these elements modify the statements by inserting these two statements ahead.

SA2(26) = ±n.m where n.m, an educated guess

DO 3 I = 1, abc abc, a number of times

must number (identify) the three statements are

last statement in loop to be re-computed.

and place that number here

The statements now read, for example

```
     SA2(26) = +10.0

     DO 33 I = 1, 50

     SAO(02) - -(PO(01) + A2(26) * P5(09))

     SA1(04) = -(P4(06) * AO(02) + P3(07))

33   SA2(26) = -(A1(04))
```

This method will provide A2(26) with an initial value for use in the SAO(02) computation, etc. Finally SA2(26) will get a new value = -(A1(04)) which will be used to compute a new SAO(02). If the loop is convergent within 50 iterations, then A2(26), AO(02), and A1(04) have theoretical values. If the loop is divergent the values calculated will probably cause 100 volts to be exceeded and ANASET will identify these components readily. If the loop is oscillatory this fact may escape detection particularly if the amplitude is within ±100 volts.

Should this method not be used to calculate theoretical component values simply write the statements as planned. Static check values and amplifier values using the analog data will still be computed properly. Errors of the type shown in the OUTPUTS section, paragraph 13, amplifier nn2, will occur, but may be readily attributed to lack of proper theoretical value.

6. In order to write an equation specifying the static output of a RESOLVER the statement takes the form:

```
CALL RESLVR (M, INPUT 1, INPUT 2, INPUT 3, OUTPUT 1,

....OUTPUT 4)
```

where M represents the MODE of the resolver: 1R, 2R, ROTATION, or INVERSE. The inputs are usually An (xx) representing angle, radius vector, x coordinate, y coordinate, etc. The outputs are stated as SRMn (xx) representing the various resolver output multipliers.

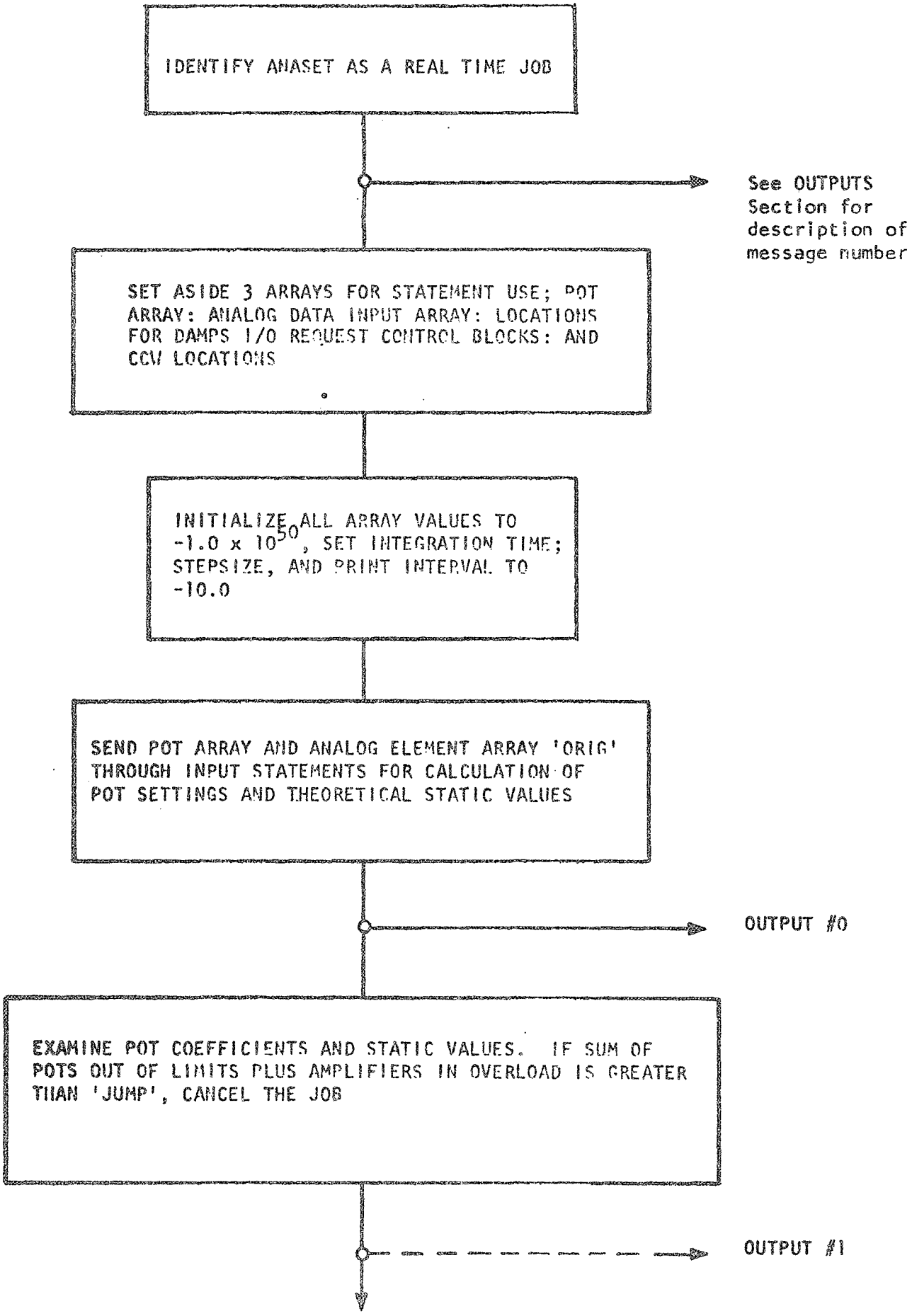
7. If the problem contains multiplying DACS a statement of the sype SDI (nn) = XX.XXXX must be specified to set the proper digital value into the DAC (nn). For static check purposes the DAC being used would be listed as

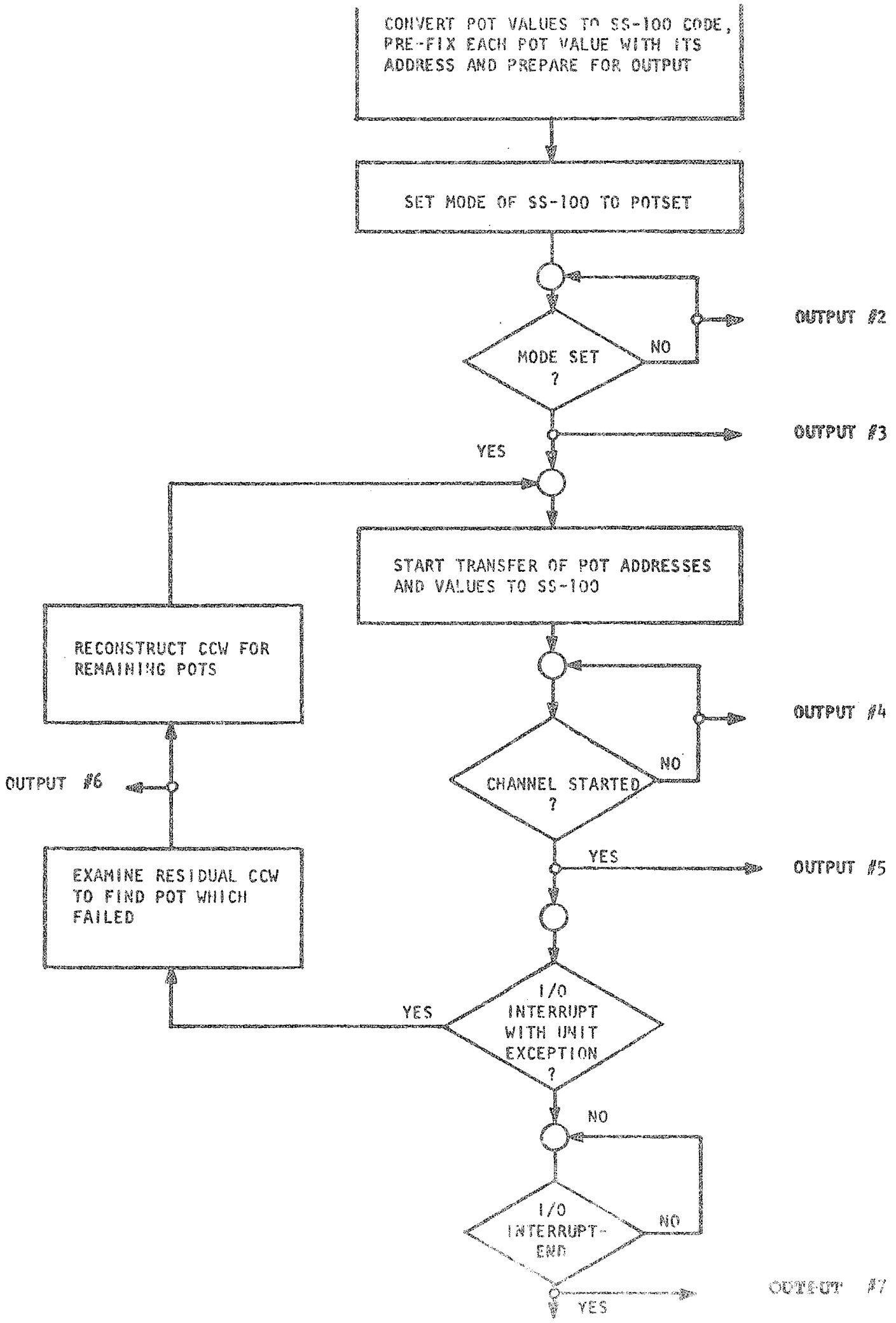
```
SDI (nn) = XX.XXXX

SDR (nn) = -(DI(nn) *An(mm))/100.0
```

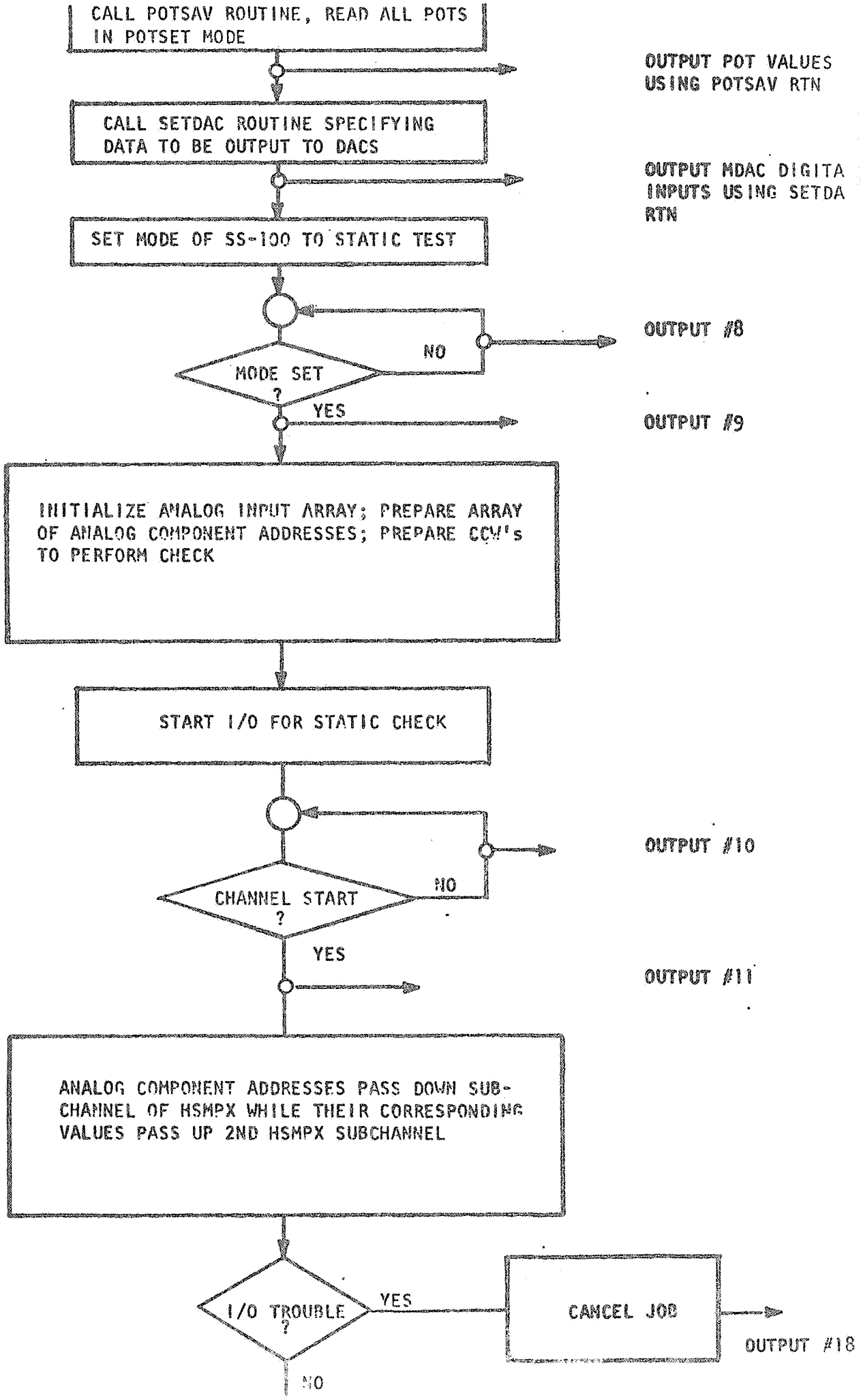
where An (mm) provides the analog input to the multiplying DAC. An (mm) may be replaced by a constant such as -100.0 where a reference voltage is used.

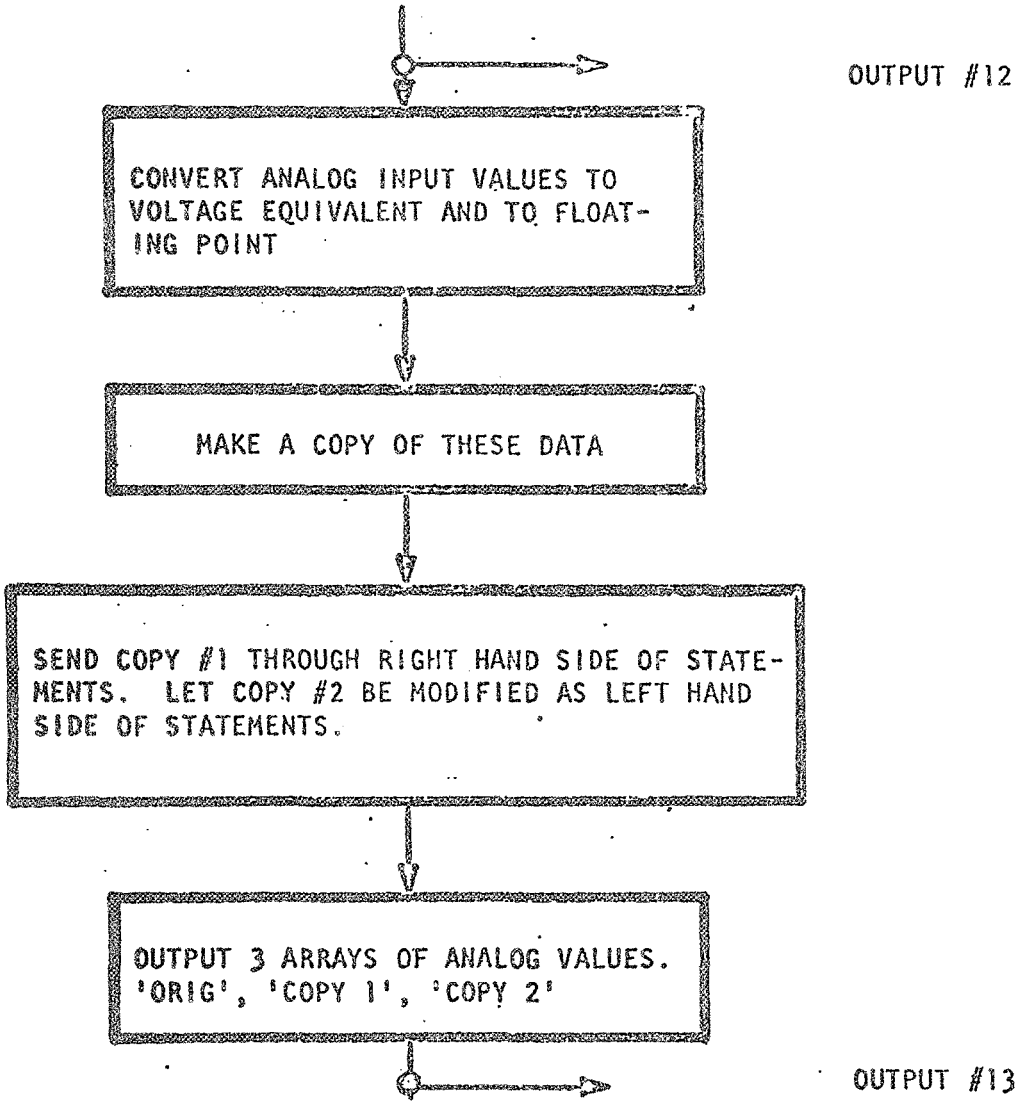
The output of the MDAC, when used in further statements, is written DR (nn) as in the statement

```
A1(06) = -(A1(08) *P1(09) + DR(12) * P4(07))
```


IDENTIFY ANASET AS A REAL TIME JOB
See OUTPUTS Section for description of message number
SET ASIDE 3 ARRAYS FOR STATEMENT USE; POT ARRAY: ANALOG DATA INPUT ARRAY: LOCATIONS FOR DAMPS I/O REQUEST CONTROL BLOCKS: AND CCW LOCATIONS
INITIALIZE ALL ARRAY VALUES TO -1.0 x 10^50, SET INTEGRATION TIME; STEPSIZE, AND PRINT INTERVAL TO -10.0
SEND POT ARRAY AND ANALOG ELEMENT ARRAY 'ORIG' THROUGH INPUT STATEMENTS FOR CALCULATION OF POT SETTINGS AND THEORETICAL STATIC VALUES
OUTPUT #0
EXAMINE POT COEFFICIENTS AND STATIC VALUES. IF SUM OF POTS OUT OF LIMITS PLUS AMPLIFIERS IN OVERLOAD IS GREATER THAN 'JUMP', CANCEL THE JOB
OUTPUT #1


CONVERT POT VALUES TO SS-100 CODE, PRE-FIX EACH POT VALUE WITH ITS ADDRESS AND PREPARE FOR OUTPUT
SET MODE OF SS-100 TO POTSET
MODE SET ?
NO
OUTPUT #2
YES
OUTPUT #3
START TRANSFER OF POT ADDRESSES AND VALUES TO SS-100
CHANNEL STARTED ?
NO
OUTPUT #4
YES
OUTPUT #5
I/O INTERRUPT WITH UNIT EXCEPTION ?
YES
EXAMINE RESIDUAL CCW TO FIND POT WHICH FAILED
OUTPUT #6
RECONSTRUCT CCW FOR REMAINING POTS
NO
I/O INTERRUPT-END
NO
YES
OUTPUT #7


CALL POTSAV ROUTINE, READ ALL POTS IN POTSET MODE
OUTPUT POT VALUES USING POTSAV RTN
CALL SETDAC ROUTINE SPECIFYING DATA TO BE OUTPUT TO DACS
OUTPUT MDAC DIGITA INPUTS USING SETDA RTN
SET MODE OF SS-100 TO STATIC TEST
OUTPUT #8
NO
MODE SET ?
YES
OUTPUT #9
INITIALIZE ANALOG INPUT ARRAY; PREPARE ARRAY OF ANALOG COMPONENT ADDRESSES; PREPARE CCW's TO PERFORM CHECK
START I/O FOR STATIC CHECK
OUTPUT #10
NO
CHANNEL START ?
YES
OUTPUT #11
ANALOG COMPONENT ADDRESSES PASS DOWN SUB-CHANNEL OF HSMPX WHILE THEIR CORRESPONDING VALUES PASS UP 2ND HSMPX SUBCHANNEL
I/O TROUBLE ?
YES
CANCEL JOB
OUTPUT #18
NO


OUTPUT #12
CONVERT ANALOG INPUT VALUES TO VOLTAGE EQUIVALENT AND TO FLOATING POINT
MAKE A COPY OF THESE DATA
SEND COPY #1 THROUGH RIGHT HAND SIDE OF STATEMENTS. LET COPY #2 BE MODIFIED AS LEFT HAND SIDE OF STATEMENTS.
OUTPUT 3 ARRAYS OF ANALOG VALUES. 'ORIG', 'COPY 1', 'COPY 2'
OUTPUT #13

OUTPUTS

0. This output is a listing of each analog element used in the statement input. Pot coefficients, static values, and integrator derivative values are printed out. Out-of-limit components are flagged as are pots which must be set manually ANASET refers to these values as the THEORETICAL.

1. If the nubmer of out-of-limit components above exceeds a value 'JUMP', a check value of ANASET, then this output appears.

   'EXCESSIVE OUT OF LIMIT COMPONENTS FORCES CANCELLATION OF THIS JOB'.

2. 'ATTEMPTING TO SET ANALOG TO POTSET MODE, UNITS ARE BUSY AND REQUEST IS QUEUED'. This message indicates busy status on Model 44 HSMPX; perhaps non-availability of analog console. The message may not appear because of DAMPS error messages should analog components be in the off state. If there is no continuance of job after cursory examination of equipment, cancel the job and notify maintenance personnel.

3. 'MODE SET FOR POTSET'. Message indicates normal program flow.

4. 'ATTEMPTING TO START POTSET, UNITS ARE BUSY AND REQUEST IS QUEUED'. This message is similar to #2. If not preempted by cancellation data from DAMPS, the 'busy' time should be accountable, otherwise system is causing looping. Cancel job and call maintenance.

5. 'POTSET NOW IN PROGRESS'. Data is flowing to the analog console and potentiometers are being set properly. The SS-100 is responding correctly to the 1044 linkage system.

6. 'POTENTIOMETER NUMBER XXX WAS NOT SET PROPERLY, POT SET CONTINUES'. This message is self-explanatory; however, it indicates a sluggish, if not malfunctioning POT. ANASET keeps only the coefficient values for future use, therefore this POT must be manually set or a substitute made on the analog computer for proper static check.

7. 'POT SET COMPLETE'. This message indicates normal program flow. Pause 66 also appears at this time allowing the operator to set manual POTS.

8. 'ATTEMPTING TO SET MODE TO STATIC CHECK, LIMITS ARE BUSY AND REQUEST IS QUEUED'. Similar to #2 and #4. Should be temporary condition, if not, call maintenance.

9. 'MODE SET TO STATIC CHECK'. Message indicates normal flow.

10. 'ATTEMPTING TO START ANALOG SCAN, UNITS ARE BUSY AND REQUEST IS QUEUED'. See messages 2, 4, and 8 for proper action.

11. 'DIGITAL COMPUTER NOW SCANNING ANALOG'. Analog component addresses are passing through HSMPX subchannel 1 while their converted outputs are passing through HSMPX subchannel 0. If I/O trouble occurs, DAMPS will cause cancellation of job.

12. 'ANALOG SCAN OF CONSOLE n IS NOW COMPLETE'. Message indicates suffessful completion of reading analog values. Normal program flow.

13. This output is a listing of amplifier data in two forms. First the theoretical value of each amplifier is listed, then the value of the amplifier as measured on the analog computer.

OPERATIONS

ANASET is a set of subroutines in the system relocatable library which are combined with the user's ANASET structure statements prior to execution by a catalogued procedure. The executable ANASET program is first compiled and link-edited into the system absolute library then the actual execution is a separate job step (RTJOB).

The job control language for an ANASET run is as follows:

```
//Anyname JOB accounting information
// ACCESS SDSABS
// DELETE SDSABS(myjob)
// CONDENSE SDSABS
//SYSRDR ACCESS PROCLIB(ANASET),SAME=SYSAB1
/*
      (User's ANASET structure statements)
/*
       PHASE myjob,R
       INCLUDE XL201,L
       INCLUDE XL201001,L
/*
/&
```

After execution of the above, the program is in the system absolute library and may be executed using the following:

```
//Anyname RTJOB accounting information
//SYSRDR ACCESS RTPROC(HYBEXEC),192='DAMPS2'
/*
// EXEC myjob
/*
/&
```

Sample ANASET Output

The following computer listing is a sample run with the example discussed earlier in this write up. The SS-100 was actually patched with the sample problem.

********** THEORETICAL VALUES OF POTENTIOMETERS ARE AS FOLLOWS *****

POT NUMBER 1 COEFFICIENT = 0.5326

POT NUMBER 121 COEFFICIENT = 0.2494

POT NUMBER 204 COEFFICIENT = 0.5000

POT NUMBER 206 COEFFICIENT = 0.3333

POT NUMBER 210 COEFFICIENT = 0.1000

POT NUMBER 212 COEFFICIENT = 0.5000

POT NUMBER 215 COEFFICIENT = 0.6652

POT NUMBER 302 COEFFICIENT = 0.1250

POT NUMBER 308 COEFFICIENT = 0.7347

********** MODE SET FOR POTSET **********

********** POTSET NOW IN PROGRESS **********

***** ***** POT NUMBER 215 VALUE = 6652 WAS NOT PROPERLY SET *****

- ********** POT SET COMPLETE **********

PAUSE SET MANUAL POTS AND REPLY EOB

********** DIGITAL COMPUTER NOW READING CONSOLE **********

********** POT SCAN IS NOW COMPLETE **********

********** ACTUAL VALUES OF POTENTIOMETERS ARE AS FOLLOWS **********

```
749 003 0.0034 004 0.0048 005 0.0397 006 0.9995 007 0.3989 008 0.9998 009 0.1007 010 0.0040
201 013 0.0314 014 0.9995 015 0.9995 016 0.1001 017 0.1002 018 0.9995 019 0.9995 020 0.0596
998 023 0.7915 024 -.1034 025 0.0049 026 0.0104

195 103 0.9995 104 0.9996 105 0.1002 106 0.9995 107 0.9995 108 0.9995 109 0.9998 110 0.3510
995 113 0.0629 114 0.9998 115 0.9995 116 0.0482 117 0.0997 118 0.9995 119 0.9998 120 0.0629
995 123 0.4075 124 0.2631 125 0.1701 126 0.1127

151 203 0.5361 204 0.4999 205 0.5512 206 0.3332 207 0.2392 208 0.0048 209 0.0046 210 0.0995
99 213 0.1403 214 0.0096 215 0.6649 216 0.1002 217 0.0861 218 0.3659 219 0.0497 220 0.1106
07 223 0.1110 224 0.0824 225 0.0436 226 0.0366

46 303 0.0098 304 0.0231 305 0.9998 306 0.3996 307 0.2582 308 0.7343 309 0.3018 310 0.2431
04 325 0.0687 326 0.0488
```

********** POTENTIOMETERS OUT OF LIMITS - MORE THAN 10 MILLIVOLTS **********

METER	THEORETICAL VALUE	ACTUAL ANALOG VALUE	DIFFERENCE
	0.5326	0.5313	0.0013

********** END OF OUTPUT FOR POTSAV JOB **********

FOR POTS WITH SUSPECT VALUES

********** MODE SET TO STATIC CHECK **********

********** DIGITAL COMPUTER NOW SCANNING ANALOG **********

********** ANALOG SCAN IS NOW COMPLETE **********

********** AMPLIFIER READINGS AND CALCULATIONS **********

	THEORETICAL VALUES	ACTUAL ANALOG VALUES	DIFFERENCE
MULTIPLIER 2 VALUES ARE	66.63625	66.52423	
MULTIPLIER 101 VALUES ARE	-32.62506	-32.52350	
AMPLIFIER 5 VALUES ARE	-53.25999	-53.10706	0.1529
AMPLIFIER 202 VALUES ARE	44.40390	44.20705	0.1969
AMPLIFIER 204 VALUES ARE	84.94849	84.95909	
AMPLIFIER 206 VALUES ARE	-50.00000	-50.06714	
AMPLIFIER 209 VALUES ARE	10.00000	9.97436	
AMPLIFIER 307 VALUES ARE	33.32999	33.26822	
AMPLIFIER 406 VALUES ARE	-44.40390	-43.86520	0.5387
AMPLIFIER 408 VALUES ARE	-73.47343	-73.50749	
AMPLIFIER 510 VALUES ARE	-1.00000	-1.11098	
AMPLIFIER 512 VALUES ARE	1.00000	1.01331	
SCALED FOR CHK AMP INTEGRATOR 5 INPUT EQLS	1.32857	1.30631	
SCALED FOR CHK AMP INTEGRATOR 209 INPUT EQLS	-81.68596	-81.33316	0.3528
SCALED FOR CHK AMP INTEGRATOR 307 INPUT EQLS	61.63623	61.51374	

********** STATIC TEST COMPLETE **********

STOP 0
IAD6I END RTJOB

AutoTrak

AutoTrak is a package which provides the programmer with an execution-time interactive capability. It is designed to run on the IBM 360 Model 44 under the Data Acquisition Multiprogramming System (DAMPS).

The AutoTrak feature allows the FORTRAN programmer to modify a variable dynamically during execution by entering the variable name from the console typewriter. He can also display the variable by name without modifying it. The display can be obtained upon request or a single request can specify continuous monitoring in which case the variable is displayed periodically until monitoring is cancelled. All variables -- including one and two dimensional arrays -- in all programs compiled under AutoTrak may be accessed by name.

If a variable is to be modified, the programmer has a choice of entering the new value in the following formats depending on the type of variable

(1) Integer

(2) Real (Single or Double Precision)

(3) Hexadecimal

(4) Logical

(5) Complex

After the new value is entered, the address of the variable and its old and new values are typed out. The program then continues with the new value.

If a variable is to be displayed, the programmer enters the program and variable names. The address and current contents of the variable are displayed after which the program continues.

During execution the programmer requests access to a variable by pressing the interrupt key on the console. AutoTrak then asks for

the program and variable names and the mode of access (i.e. modify, display, or monitor). If modify is specified, a value is requested.

The variable is changed and execution continues. If the name of a dimensioned variable is entered, AutoTrak requests the row and column index and then accesses the variable. If monitor is specified, periodic displaying continues until the interrupt key is pressed again.

Additional features of AutoTrak are:

1) Relative Addressing -

The programmer may leave the variable name blank when he enters names from the console. In this case AutoTrak will request an address. The address is added to the starting address of the program which was entered and the resulting location is accessed. This allows any location within a program to be accessed regardless of whether the location has an associated variable name. In addition, variables or instructions in assembly language programs or FORTRAN programs not compiled under AutoTrak can be accessed using this method.

2) Absolute Access -

An absolute core location can be accessed under AutoTrak. This is useful for monitoring locations outside the problem program area during execution.

3) Error Control -

Normally, if invalid characters are entered as data, the job is cancelled. Before accepting input data from the typewriter, AutoTrak

temporarily inhibits this cancel, and, instead, requests new data if input is invalid. Thus, the programmer cannot inadvertently cancel his own job by a keyboard mistake.

4) Input Control -

The programmer may use up to four data sets to store AutoTrak information. During execution, he can dynamically control the number of data sets used and their order of use. Thus, variables can be accessed by name from routines compiled in previous independent jobs.

AutoTrak is composed of 7 programs which construct, store, and access a table of program and variable names, and addresses. The table is built while the FORTRAN Compiler and Linkage Editor are in execution. It is then written on disk and read in as required during problem program execution. A description of individual programs follows.

1. Scanner - Resides in core during compile and link edit job steps. It extracts program and variable information. A table of addresses is built in core.

2. Lload - Loads and relocates program "Scanner".

3. Unload - Writes the table built by Scanner onto the data set assigned to SYS004, and restores pointers and indicators set by "Lload".

4. Finder - Searches the table for program and variable names. The corresponding addresses are returned to the program which calls Finder. Entry point "plunk" accesses an absolute location.

5. Prezz - Scans the name entry and formats it for input to Finder.

6. Talke - Communicates with the programmer via the console typewriter.

7. Dsterm - Inhibits job cancellation when invalid data is entered.

Programming

The programmer invokes AutoTrak by attaching a routine called "Talke" to the console interrupt key. This is done via a call to subroutine Extint specifying Talke as one of the parameters. This "initialization" call to Extint associates a program code with subroutine Talke. Subsequent "Request" calls to Extint specifying the same program code will result in execution of Talke if the console interrupt key has been pressed. The user should be familiar with Subroutine Extint before AutoTrak is used.

Subroutine Talke communicates with the programmer via the console typewriter. It accepts as input subprogram names, variable names and values, and indicators for mode of access. It then accesses the variable and performs the indicated operations.

An example of coding in a FORTRAN Program using AutoTrak is as follows:

```
      EXTERNAL TALKE
         .
         .
         .
         .
      CALL EXTINT (1, TALKE)            (Initialization)
         .
         .
         .
         .
10    CALL EXTINT (0, 1, IRET)          (Request)
         .
         .
         .
         .
         .
         .
         .
15    CALL NOEXT                        (Release Control)
      STOP
      END
```

At statement 10 Subroutine Talke will be executed, if the console interrupt key has been pressed and input will be requested from the typewriter. The programmer can then access a variable and return control to his program. Statement 15 releases control of the interrupt key. (See Subroutine Extint)

Operating Procedure

1) Name Entry

When the Interrupt key is pressed, the message "Enter Names" will be written on the console typewriter. The programmer enters a program name, variable name, and access code in the format

xxxxx,yyyyy,H

where xxxxx is a program name or key name (1-8 characters)

yyyyy is a variable name (0-8 characters)

and H is a format code.

Upper or lower case letters may be entered. Blanks are ignored.

The valid entries for H are as follows:

I - Integer Modification
F - Real Modification (F Format)
E - Real Modification (E Format)
D - Double Precision Modification
Z - Hexadecimal Modification
L - Logical Modification
C - Complex Modification
Q - Relative Modification
R - Display Variable
M - Monitor Variable
I2 - Halfword Integer Modification
R2 - Display Halfword Integer
M2 - Monitor Halfword Integer

Codes I, F, E, D, Z, L, and C request variable modification and the letter indicates the FORTRAN Format in which the new value should be entered. Code I2 is the same as I except the variable will be accessed using halfword instructions. Code Q, is used to modify a location relative to the beinning of a program. No variable name is specified

for this code. An address is typed in and it is added to the beginning address of the specified program. The resulting location is then accessed. Codes R and R2 display the accessed location using fullword and halfword instructions respectively. Codes M and M2 monitor a location using fullword and halfword instructions respectively. The variable name may be omitted for Codes R, R2, M, M2 also.

Examples of name input are as follows:

```
MAIN44, J, I.
   MAIN   ,   DATA   ,   F
M, IRET, I2 ·
       SUBX , , Q
   READER, INPUT, R
READER , , R
   MAIN , , M2
```

A special key name "12ZZ" should be entered when an absolute location is to be accessed. The entry is

12ZZ , , H

where H is Q, R, M, R2, or M2

The absolute address is then entered in hexadecimal.

A key name entry -& (Ampersand)- is used to signal AutoTrac that the last entry has been made and the user wants control returned to his program. The entry is:

Enter Names

&

2) Address Entry

After the name entry has been processed the message "ADDR" will be typed if an address entry is required. An address is necessary when code Q is used, when codes R, R2, M, M2 are used without specifying a variable name, or when an absolute location is to be accessed. The programmer enters the address in 8 hexadecimal characters. The address is added to the beginning address of the specified program and the resulting location is accessed.

Examples:

a) Enter Names

DRIVER , , Q

ADDR

000008E4

b) Enter Names

MAIN , , R

ADDR

000000EC

c) Enter Names

12ZZ , , R

ADDR

000012B0

3) Index Entry

If AutoTrak finds that the variable name entered is a dimensioned variable, the message "Index" will be typed. The programmer replies

with an entry of the form,

RCTL

where R is a 5-digit row index

C is a 5-digit column index

T is the total number of rows (5-digits)

L is the length of each variable in the array in bytes (5-digits)

Entries C and T are required only if the array is two-dimensional. Entry L is required only if the length of each variable is other than 4. If L is not entered, the length defaults to 4.

Examples:

Assume the following FORTRAN statements:

```
DIMENSION JARRAY (50), RARRAY (10, 10)
INTEGER*2 KAP (10), MAP (5, 5)
```

a) Enter Names

MAIN, JARRAY, I

INDEX

00025

b) Enter Names

MAIN, RARRAY, F

INDEX

0000099996000010

c) Enter Names

MAIN, KAP, I2

INDEX

00005000000000000002

e) Value Entry

If code I, I2, F, E, D, Z, L, C, or Q was entered in the name entry (indicating variable modification), the message "value" will be typed. The programmer enters a new value for the variable. The format of the entry depends on the type of variable being accessed.

The following table gives the FORTRAN format in which new values are entered:

Access Code	Format
I	I8
I2	I8
F	F20.10
E	E20.10
D	D20.10
Z	Z8
L	L4
C	F20.10
Q	Z8

Examples:

a) Enter Names

MAIN, J, I

VALUE

00000682

b) Enter Names

PROG1, RET, F

VALUE

22.6

c) Enter Names

PROG2, , Q

ADDR

00000E4

VALUE

000000E6

d) Enter Names

PROG1, LOGIC, L

VALUE

T

5) <u>Value Output</u>

After AutoTrak has determined the absolute location to be accessed, the address, old value, and new value of the location are typed out. The format is

AAAAAAAA VVVVVVVV NNNNNNNN

where AAAAAAAA is the Address

VVVVVVVV is the old value

NNNNNNNN is the new value

Each is an 8-character hexadecimal number.

In the case of displaying or monitoring a variable, only the address and old (current) value are significant.

Examples:

a) Enter Names

MAIN44, JRET, I

VALUE

00000030

000098D4 000000A4 0000001E

b) Enter Names

SUBX, FTIME, F

VALUE

22.5

0000A120 00000000 42168000

c) Enter Names

DRIVE, , Q

ADDR

000000E8

VALUE

00000168

000088E8 00000001 00000168

6) <u>Error Messages</u>

The following messages are typed if input errors are found. For each message, the action taken by AutoTrak is specified.

MESSAGE: INVALID FORMAT

CAUSE: The mode of access was invalid. The code must be one of those listed in the

ACTION: Request new name entry

MESSAGE: INVALID PROGRAM NAME

CAUSE: The program specified in the names entry could not be found

ACTION: Request new name entry

MESSAGE: INVALID VARIABLE NAME

CAUSE: The variable specified in the names entry could not be found, or no variable name was specified for codes I, I2, F, E, D, Z, L, C.

ACTION: Request new name entry

MESSAGE: INVALID INPUT DATA

CAUSE: An invalid character has been entered under FORTRAN formats I, E, F, D, or Z.

ACTION: Request new name entry.

7) Cancel Codes

If an unrecoverable error occurs during AutoTrak processing, a cancel code is placed in location 76 (Hex 4C). The cancel code occupies one byte and specifies the error and the program in which the error occurred. The system is placed in a loop and processing halts. Register 15 contains additional information concerning the cause of the error. The following table gives the code, program, and cause of error.

01(FINDER) - An error has occurred in the point (SVC 8) operation on the data set assigned to program and variable information.

02(FINDER) - An error has occurred in the read (SVC 4) operation on the data set assigned to program and variable information.

03(UNLLOAD) - An error has occurred in the write (SVC 5) operation on the data set assigned to SYS004.

04(UNLLOAD) - An error has occurred in the point (SVC 8) operation on the data set assigned to SYS004.

05(UNLLOAD) - An error has occurred in opening (SVC 2) the data set assigned to SYS004.

06(UNLLOAD) - An error has occurred in closing (SVC 3) the data set assigned to SYS004.

07(LLOAD) - An error has occurred in loading (SVC 13) program Scanner from SDSABS.

For each error the contents of Register 15 are described in the IBM System/360 Model 44 Programming System Guide to System Use Manual under the appropriate SVC.

SUBROUTINE EXTINT

This subroutine gives the user control of the external interrupt key on the console. The programmer may "attach" two routines to the interrupt key. Subsequent calls to EXTINT request execution of one of the routines if the interrupt key has been pressed.

The attached routines are written as subroutines and may be FORTRAN or Assembly language. All facilities are available to these routines subject to BAMPS and PS/48 restrictions.

Subroutine EXTINT does no input/output. It is serially reusable but not reentrant. If EXTINT is executed on Priority Interrupt Levels, the user should be certain that it will never be interrupted and reentered from a higher level.

Calling Sequence

Two types of calls are used: initialization and request.
The initialization call is

CALL EXTINT (A,B)

where A is A program code - 1 or 2

and B is a program name which will be assicated with code A.

Example: CALL EXTINT (1, SUBX)

This call associates a routine named SUBX with program code 1. Subsequent "request" calls to EXTINT specifying code 1 will cause execution of SUBX if the interrupt key has been pressed.

The request call is

CALL EXTINT (0, D, R)

where D is a program code - 1 or 2

and R is a return code which will be set to 1 if the interrupt key has been pressed and 0 if it has not.

Example: CALL EXTINT (0, 1, IRET)

This call says execute the routine associated with code 1 if the interrupt key has been pressed. Upon return from EXTINT, the variable IRET will equal 1 if the interrupt key had been pressed (i.e. the routine associated with code 1 has been executed) or 0 if the key had not been pressed (i.e. the associated routine has not been executed).

Note that variable IRET can be used to alter execution of the calling program in accordance with the interrupt key. A special request call to EXTINT,

CALL EXTINT (0, 0, IRET)

Specifying a program code of 0 can be used to determine whether the interrupt key has been pressed without executing an associated program.

Every request call to EXTINT resets the interrupt key indicator. Therefore, subsequent request calls will have a return code of 0 until the key is pressed again.

Each routine specified in an EXTINT call must appear in a FORTRAN EXTERNAL statement. A new routine may be associated with a program code at any time, and it replaces the old routine.

If an error is encountered in the calling sequence to EXTINT, it will exit with no action taken. The following calls are invalid

1) CALL EXTINT (A, B) , A ≠ 0, 1, 2
2) CALL EXTINT (0, D, IRET) , D ≠ 0, 1, 2
3) Request call specifying program code for which no intialization call has been made.

Subroutine EXTINT has an additional entry point called NOEXT. This part of the routine releases control of the external interrupt key and restores the system to normal operation. The calling sequence is

CALL NOEXT

A "CALL NOEXT" must always be issued prior to completion of the job in which subroutine EXTINT is used. It should be issued after the last call to EXTINT.

Subroutine EXTINT uses locations 1400 and 1404 (decimal) to save registers. These locations are boot strap words for a DAMPS uncontrolled real time task attached to Priority Interrupt Level 15. Therefore, if EXTINT is used in a job, an uncontrolled RTT should not be attached to level 15.

The following illustrates use of EXTINT in a FORTRAN program.

```
EXTERNAL SUBX, SUBY, SUBZ
CALL EXTINT (1, SUBX)                    (Initialization)
CALL EXTINT (2, SUBY)
    .
    .
    .
CALL EXTINT (0, 2, IRET)                 (Request)
    .
    .
    .
```

```
CALL EXTINT (0, 1, JRET)                    (Request)
    .
    .
CALL EXTINT (1, SUBZ)                       (Re-initialize code 1)
    .
    .
    .
CALL EXTINT (0, 1, KRET)                    (Request)
    .
    .
    .
    .
CALL NOEXT                                  (Released Control)
STOP
END
```

DAMPS ROUTINES FOR HYBRID PROGRAMMING

INTRODUCTION

The Data Acquisition Multiprogramming System (DAMPS) is designed to support the S/360 Model 44 in online real-time applications. The Hybrid user will find this system provides an easy to use monitor facility which allows the Hybrid programmer to

1) Write real-time applications programs in FORTRAN.
2) Secure I/O overlap with CPU activity during execution.
3) Associate various tasks of the Hybrid program with either Priority Interrupt Levels or queued priority facilities.
4) Assume little responsibility or as much responsibility as desired for the preservation of system status during interrupt servicing. (The response time of the system is inversely proportional to the degree of responsibility assumed.)
5) Have re-entrant routines for a variety of tasks including real-time I/O and mathematical functions with attendant main storage savings.
6) Tailor the system to support new or different I/O devices as needs dictate.

There are a number of "convenience" routines which augment the DAMPS monitor for Hybrid use and these are the subject of this paper. Only those routines which are analog device independent are described since specific device routines are usually peculiar to a few installations.

The reader should be familiar with the DAMPS monitor discussed in several IBM publications.

DETACH

When programming a Hybrid System whose digital facility is demand responsive to asynchronous conditions, it is advantageous to be able to dynamically re-order the Hybrid program structure. This capability is required when the initial judgment of the programmer in attaching various tasks to the Priority Interrupt System is found to be in error. The ability to DETACH a Real-Time-Task from its associated PIL and subsequently RE-ATTACH to a different PIL is required for this dynamic re-allocation. The RE-ATTACH is provided by the DAMPS routine FATACH. The DETACH is provided by executing the FORTRAN statement:

```
CALL DETACH (I)
```

where I is an integer PIL number (0-31).

PRTQUE

The DAMPS Foreground Task Queue may have entries placed in it by Real-Time-Tasks or other executing Foreground Tasks. It is convenient, particularly when debugging a Hybrid program, to know the current order of the FTQ. Subroutine PRTQUE provides this capability through a listing on SYSOPT of

1. Queue slot used or not used.
2. Entry point of queued FGT.
3. Priority of FGT.
4. Type of entry (enter or remove).
5. Next inactive slot.
6. Next active slot.
7. Parameter passed to FGT.

PRTQUE is executed by the FORTRAN statement:

```
CALL PRTQUE
```

TRIGR

A means to attach various routines to Priority Interrupt Levels for demand responsive execution is highly important to the Hybrid programmer. It is often desirable to cause the execution of these routines by triggering a PIL internally, even from self-suspending lower level tasks. Also, as a checkout aid, the facility to trigger PILs internally is a time saver by divorcing the system from hardwired interrupt. The subroutine TRIGR provides an internal trigger for the PILs desired through the FORTRAN statement

CALL TRIGR (QUAN,PIL#, PIL#,...)

where QUAN is an integer specifying the number of PILs to be triggered simultaneously ($1 \leq QUAN \leq 32$); PIL# is the specific Priority Interrupt Level to be triggered ($0 \leq PIL\# \leq 31$).

Example: To cause interrupts at levels 5, 17, 20, and 31, the calling sequence could be

CALL TRIGR (4, 17, 31, 20, 5)

HATACH

The DAMPS subroutine FATACH is used to attach a "controlled" Real-Time-Task (RTT) to a specific Priority Interrupt Level. A "controlled" RTT is one where system status is preserved by DAMPS prior to entry at interrupt time. When the Hybrid programmer requires faster response time to the external interrupt, 5 to 10 microseconds, as opposed to approximately 200 microseconds, an "uncontrolled or convertible" RTT should be attached. The subroutine HATACH provides for the attachment of an "uncontrolled or convertible" RTT to a PIL through the FORTRAN statement

CALL HATACH (RTT,SAVEAREA,PIL#,FPSAVE,IOPT)

where each parameter of the calling sequence is as described under DAMPS FATACH with the exception of ICPT. IOPT = 0, means the RTT is uncontrolled; IOPT = 15, means the RTT is convertible.

The user is cautioned that uncontrolled RTTs written in FORTRAN are not recommended unless considerable detail is known regarding the machine code generated by the compiler.

IOISER

The methods by which the DAMPS Real-Time Channel Scheduler handles I/O interrupts are concerned with posting I/O completion in Request Control Blocks, monitoring errors, and numerous other clean-up tasks. On occasion a very fast I/O interrupt servicing capability is required and there is absolutely no time for posting and checking. Rather the Hybrid programmer is interested in taking certain liberties through faith in the Real-Time devices in order to accomplish a high speed solution. Subroutine IOISER (I/O Interrupt Servicer) allows the user to gain control whenever a Real-Time I/O interrupt occurs. After the user's interrupt routine is executed (may be as little as Test I/O instruction) control returns to point of interruption. IOISER is executed by a FORTRAN statement

CALL IOISER

The user's instructions for servicing the RT interrupts must be written in Assembler language and inserted in the IOISER routine. IOISER can then be reassembled and inserted in the user's job.

Subroutine IOISER has an additional entry point, DEIOSR, which cancels the effects of IOISER, i.e., I/O interrupts will cause a branch to the system and interrupt servicing will proceed under DAMPS control. The call to DEIOSR is simply

CALL DEIOSR

NOTE: The user cannot use PIL 0 or PIL 1 with IOISER.

The response time preceding the user's in-line code is normally 8 microseconds.

PERF

The routine PERF provides the Hybrid programmer with I/O and CPU overlap in the important area of non real-time devices (printer, card reader, typewriter, etc.). Overlapped I/O with CPU is normally not available from the FORTRAN programming level and as DAMPS provides this feature for real-time I/O, PERF augments DAMPS for non real-time I/O. Print, punch, card read activity can thus be accomplished by FORTRAN routines attached to PILs without the necessity of queueing at the Foreground Level.

PERF accomplishes its tasks by using certain DAMPS real-time facilities, by setting up the proper channel programs and control blocks, and by using the format conversion sections of FORTRAN IBCOM. The user may then execute the I/O using DAMPS as if the non real-time device were real-time.

The coding in a FORTRAN program to do a "real-time", and thus overlapped, write to the line printer would be:

```
      DIMENSION SAV(90)
400   FORMAT(' DATA',15F10.3,E20.10)
                  .
                  .
                  .
      CALL PERF(SAV,&20,&30)
                  .
                  .
                  .
      CALL FRTIO(SAV)
                  .
                  .
                  .
                  .
20    WRITE(6,400)I,X,Y
30    CONTINUE
```

The actual I/O execution is started at the call 'CALL FRTIO'. Output

is formatted in accordance with statement 400. The Hybrid user may monitor the results of the operation by using the DAMPS routine FCHECK.

TIMSER

The TIMSER routine allows the FORTRAN programmer to get the current value of the Model 44 internal timer placed in a reserved integer location.

The routine executes via

```
CALL TIMSER (ITEM)
```

Upon return the core location ITEM will contain, in INTEGER*4 format, the current timer value to the nearest 1/300 second.

ISENSE

ISENSE is a function subroutine which provides the Model 44 with 32 sense switches. The sense switches are set by stopping sequential instruction processing, setting the selected sense switches to "one" or "zero", and resuming sequential instruction processing. The switches use hex location 4C in main storage for memory.

To use the sense switches to change program flow the programmer may write

```
IF (ISENSE(M).EQ.J)  GO TO 83
```

where M is an integer between 1 and 32; J = 1 or 0; and the "GO TO 83" clause may be any executable FORTRAN statement.

Combinations such as

```
IF (ISENSE(2).EQ.0.AND.ISENSE(13).EQ.1)  ....
```

are allowed.

TIMOUT

A convenience for the Hybrid programmer is manifest in a variable delay which may be interposed in an application for timing purposes (relay setting time on external devices for example) and as a debugging aid. The routine TIMOUT is provided for this purpose. The routine DOES NOT use the Model 44 interval timer. The delay is invoked through the FORTRAN statement

CALL TIMOUT (n)

where n is an integer greater than or equal to 1. The time of the delay is n times 1 millisecond.

RUNTIM

Prior to the execution of a Hybrid application program a run time estimate may be derived by using the RUNTIM routine. RUNTIM has a second entry point FINTIM.

To determine the run time of a program the FORTRAN statement

```
CALL RUNTIM
```

should be placed as the first executable statement of the code section participating. The last executable statement should be

```
CALL FINTIM
```

The execution time is listed on SYSOPT in seconds and milliseconds.

Example:

```
CALL RUNTIM
X=Y**Z
G=COS(X)
   .
   .
   .
CALL FINTIM
END
```

DAMPS AND FORTRAN

INTRODUCTION

Because Engineering personnel are concerned with accomplishing simulations on Hybrid Systems in a timely fashion, it is important that they be provided with an easy to use programming structure. This structure should include a high level programming language and a monitor facility readily directed from the high level language. The Data Acquisition Multiprogramming System (DAMPS) supporting the IBM System/360 Model 44, in combination with FORTRAN, offers the Hybrid System user a software structure which permits FORTRAN real-time programming.

This paper briefly describes the features of the DAMPS system concerned with the Real-Time job and how the Hybrid programmer may use these supervisory features from the FORTRAN level.

DAMPS CONCEPTS

DAMPS is an extension of the Model 44 Programming System. The extensions include numerous supervisory functions required to meet real-time demands without obviating the features of 44 PS. The features of DAMPS that make it particularly useful for real-time jobs are:

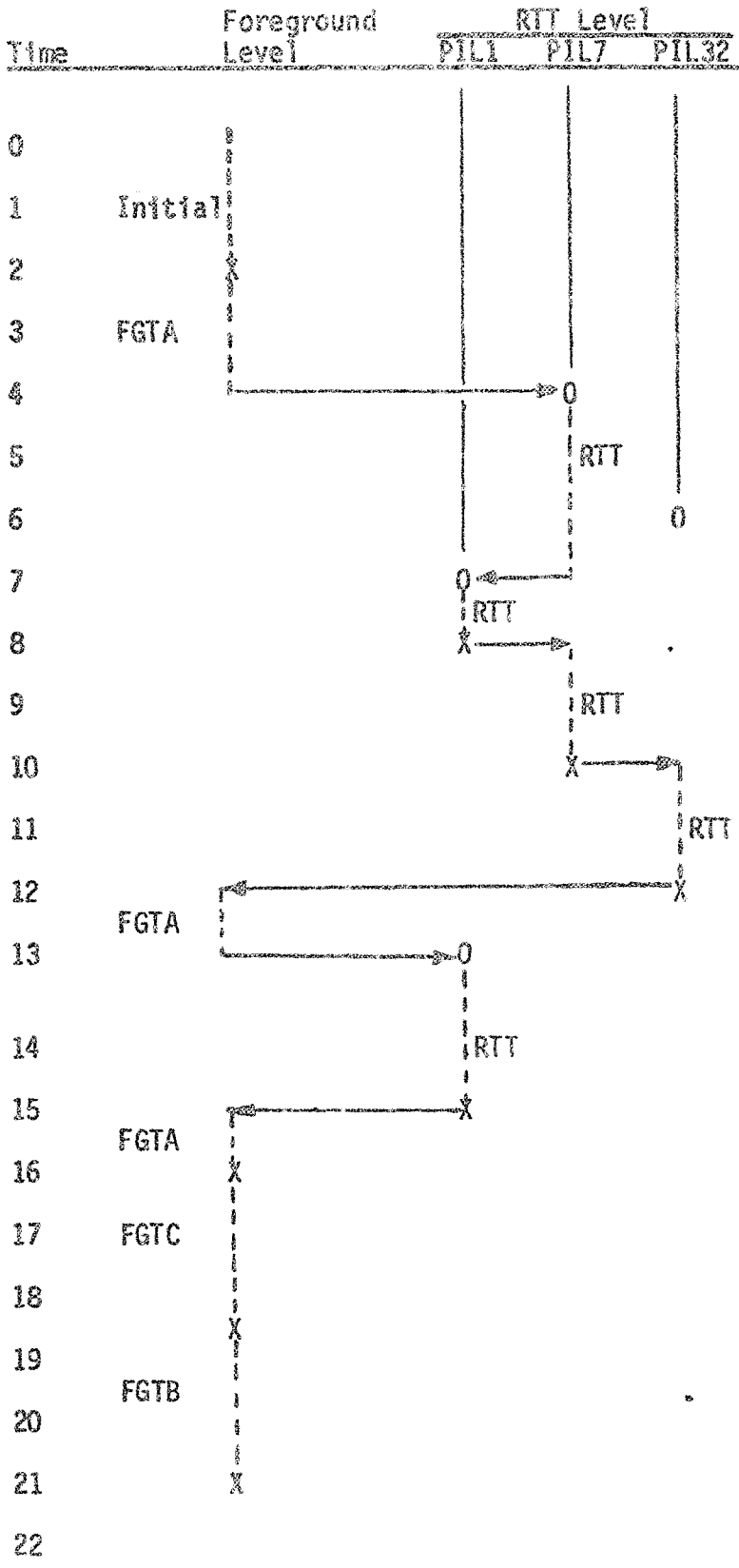
- Response to external demands through the Priority Interrupt System
- Ease of scheduling foreground processing through queueing facilities
- Special channel scheduling for real-time I/O
- Capability of dynamic overlay inherent in 44 PS available to real-time job
- Real-time I/O overlap with CPU activity
- Re-entrancy capability including real-time I/O and mathematical functions

For the Hybrid user the fundamental DAMPS concept is the Real-Time-Job (RTJ). An RTJ consists of Real-Time-Tasks (RTTs) and Foreground Tasks (FGTs). As an RTJ begins execution it assumes the role of an FGT. During this early execution RTTs may be attached to various levels of the Priority Interrupt System and other FGTs may be placed in an FGT queue. Normally this RTJ set-up FGT will terminate by enabling the Priority Interrupts to which RTTs are attached. The CPU will then begin processing the highest priority FGT in the queue while processing RTTs on a demand basis. RTTs and FGTs and their relationships are further explained in the following paragraphs.

The RTT is a routine attached to a particular level of the Priority Interrupt System. The routine begins execution as an external pulse triggers the applicable Priority Interrupt Level (PIL). The CPU suspends its current processing and immediately transfers to the RTT. At this point the user may allow the DAMPS supervisor to preserve machine status or may assume this responsibility in the RTT thereby gaining time saving in interrupt response. After the RTT completes execution the CPU returns to the point of suspension in the task interrupted. RTTs may be suspended themselves when interrupts with higher priority occur but may not be suspended by FGT processing. RTTs may attach other RTTs to PILs, may place FGTs in the Foreground queue, and may enable/disable levels of the Priority Interrupt System. In addition, RTTs may cause the execution of Real-Time I/O by invoking the facilities of the Real-Time Channel Scheduler.

The FGT is a routine which resides in a Foreground priority queue until it is executed. FGTs are placed in the queue by RTTs or other FGTs. The hierarchy of processing is determined when the queue is searched after the current FGT completes. FGTs found resident with equal priority are processed "first-in-first-out". The FGT is thus used for processing which need not be carried out on the demand responsive basis of the RTT.

The illustration below depicts the two basic operation levels of DAMPS RTJs, the RTT and FGT levels, while demonstrating the interaction of their operations.



Comments

Start RTJ and attach RTTs to PILS 32, 7 and 1; place FGTA in queue and enable PILs. Exit from current FGT and examine queue; start FGTA processing. PIL7 is triggered, start executing 7's RTT which places FGT B in queue. PIL32 triggers.... PIL1 triggered suspending processing at Level 7. PIL1's RTT will complete and system returns to complete RTT at 7. As RTT completes, RTT at 32 begins. Where complete system returns to processing suspended FGT. PIL1 triggered and RTT now queues FGT C with higher priority than FGT B. System returns to complete FGTA; then processes FGTC followed by FGTB. FGTB ends the RTJ.

The essential control needed by the FORTRAN programmer to organize the flow of an RTJ can be summed as the ability to

ATTACH RTTs to PILs

QUEUE and UNQUEUE FGTs

ENABLE and DISABLE PILs

BEGIN and END or CANCEL an RTJ

END an FGT

END an RTT

These capabilities are provided the DAMPS user and the routines are described in the following section.

DAMPS RTJ ORGANIZATIONAL ROUTINES

FIDRTJ - FORTRAN Subroutine

The FIDRTJ subroutine is used to identify an RTJ to the DAMPS supervisor. Prior to the execution of this call none of the DAMPS features can be used and any attempt to use these features would result in job cancellation. The subroutine is executed via :

```
CALL FIDRTJ
```

FEORTJ - FORTRAN Subroutine

The FEORTJ subroutine is used for normal termination of a Real Time Job. After FEORTJ has been executed, control will be returned to the supervisor which will initiate loading the next job. Failure to end an RTJ in this manner may result in interference with subsequent jobs. The FEORTJ is executed by a FORTRAN call statement with the following format:

```
CALL FEORTJ
```

FCANCL - FORTRAN Subroutine

The FCANCL subroutine is used for abnormal termination of a Real Time Job. If the DUMP option was indicated in the job cards for the run, a storage dump will be provided. The subroutine is executed by a FORTRAN call statement with the following format:

```
CALL FCANCL
```

FDONE - FORTRAN Subroutine

The FDONE subroutine is used to terminate a FORTRAN Foreground Task. Control is returned to the supervisor and the FGT with the highest priority in the Foreground Queue is initiated. Where multiple requests of identical priorities are concerned, the oldest entry will be accepted first. Use of the FDONE subroutine does not preclude use of the END statement in a FORTRAN subroutine which is to be an FGT. The FDONE subroutine is executed by a FORTRAN call statement with the following format:

CALL FDONE

FRETRN - FORTRAN Subroutine

The FRETRN FORTRAN subroutine is used to terminate a FORTRAN Real Time Task. Control is returned to the next lower RTT, when one is active. In the event that none are active, control is returned to foreground processing. Use of the FRETRN subroutine does not preclude the use of the END statement in a FORTRAN subroutine which is to be an RTT. The FRETRN subroutine is executed via:

CALL FRETRN

FQUEUE - FORTRAN Subroutine

The FQUEUE FORTRAN subroutine is used to enter a request for the execution of a Foreground Task, with an associated priority, into the Foreground Queue. The requests in the Foreground Queue are honored in priority sequence.

The FQUEUE FORTRAN subroutine is executed by a FORTRAN call statement with the following format:

CALL FQUEUE (IP1, NAME, IRC, IP2)

The entries in the parameter list have the following meanings:

IP1 - An integer from 0 - 127 representing the priority of the foreground task. All other values are invalid.

NAME - The name of the Foreground Task being queued. The name must also appear in an EXTERNAL statement.

IRC - An integer variable which will contain the completion code returned after completion of the subroutine. The value is encoded as follows:

IRC Value	Meaning
IRC = 0	Indicates the queue request was successfully placed in the Foreground Queue.
IRC = 4	Indicates that the address of the parameter list was invalid.
IRC = 15	Indicates that an invalid priority was contained in IP1.
IRC = 240	Indicates that the name of the Foreground Task was invalid. The name may not have appeared in an EXTERNAL statement.

IRC = 255 indicates that the Foreground Queue was full.

IP2 - an integer variable which can be used to pass a variable to the queued subroutine.

Only if IRC is equal to 0 was the Foreground Task queued. Any invalid conditions not indicated above will cause the job to be cancelled.

FDEQUE - FORTRAN Subroutine

The FDEQUE FORTRAN subroutine is used to delete a previous request for execution of a Foreground Task. The unqueue request, which is actually placed in the Foreground Queue, is honored just prior to the next Foreground scheduling. A queue request is identified on the basis of the name of the Foreground Task and the priority of the request. When the unqueue request is honored, all requests with the same name and priority will be deleted.

The FDEQUE FORTRAN subroutine is executed by a FORTRAN CALL with the following format:

```
CALL FDEQUE (IP1, NAME, IRC)
```

Each of the names in the parameter list have the following meaning:

IP1 - An integer from 0 - 127 representing the priority of the Foreground Task. All other values are invalid.

NAME - The name of the Foreground Task to be deleted from the queue. The name must also appear in an EXTERNAL statement.

IRC - An integer variable containing the completion code returned after completion of the subroutine. The return code is encoded as follows:

IRC = 0	Indicates that the dequeue request was successfully placed in the queue.
IRC = 4	Indicates that the address of the paramter list was invalid.
IRC = 15	Indicates that an invalid priority was contained in IP1.
IRC = 240	Indicates that the name of the Foreground Task was invalid. The name may not have appeared in an EXTERNAL statement.
IRC = 255	Indicates that the Foreground Queue was full.

Only if IRC is equal to 0 was the request queued. Any invalid condition other than those outlined above will cause the job to be cancelled.

FATACH - FORTRAN Subroutine

The FATACH FORTRAN subroutine is used to associate an RTT with a specific priority interrupt level. Subsequent priority interrupts at that level will cause control to be transferred to the associated RTT. The RTT is terminated by use of the FRETRN FORTRAN subroutine. The FATACH subroutine does not enable the priority interrupt level; this is accomplished by the FENPIL FORTRAN subroutine.

The FATACH subroutine is executed by a FORTRAN call statement with the following format:

```
CALL FATACH (RTT, RTSAVE, N, FPSAVE)
```

The parameter list is described below:

RTT - The name of the Real Time Task to be attached. The RTT must have been compiled as a separate FORTRAN subroutine. The name must appear in a FORTRAN EXTERN statement in the subroutine where the attach is executed.

RTSAVE - The name of an array containing a 61-word save area. A different save area must be established for each RTT, and it may not be used for any other purpose.

N - An integer from 0-31 indicating the priority interrupt level to be associated with the RTT.

FPSAVE - The name of an array containing a four double-word save area for floating point registers. A different area must be established for each RTT, and it may not be used for any other purpose.

FENPIL - FORTRAN Subroutine

The FENPIL subroutine is used to enable priority interrupt levels. All the levels indicated in an individual call are enabled simultaneously. Pending interrupts on the indicated levels will be cleared if the reset option is specified.

The FENPIL subroutine is executed by a FORTRAN call statement with the following format:

CALL FENPIL (I, J1, J2, ... Jn)

The parameter list is described below:

I - Is an integer indicating the selection of the reset option. If I = 255, the selected priority interrupts will have pending interrupts reset at the time of enabling. If I = 0, the pending interrupts will not be reset.

Jn - Each Jn is an integer from 0 to 31 indicating a specific priority interrupt level to be enabled. There will be one Jn for each level to be enabled.

FDSPIL - FORTRAN Subroutine

The FDSPIL FORTRAN subroutine is used to disable priority interrupt levels. All the levels indicated in an individual call are disabled simultaneously.

The FDSPIL subroutine is executed by a FORTRAN call statement with the following format:

CALL FDSPIL (I, J1, J2, ... Jn)

The parameter list is described below:

I - Is an integer zero.

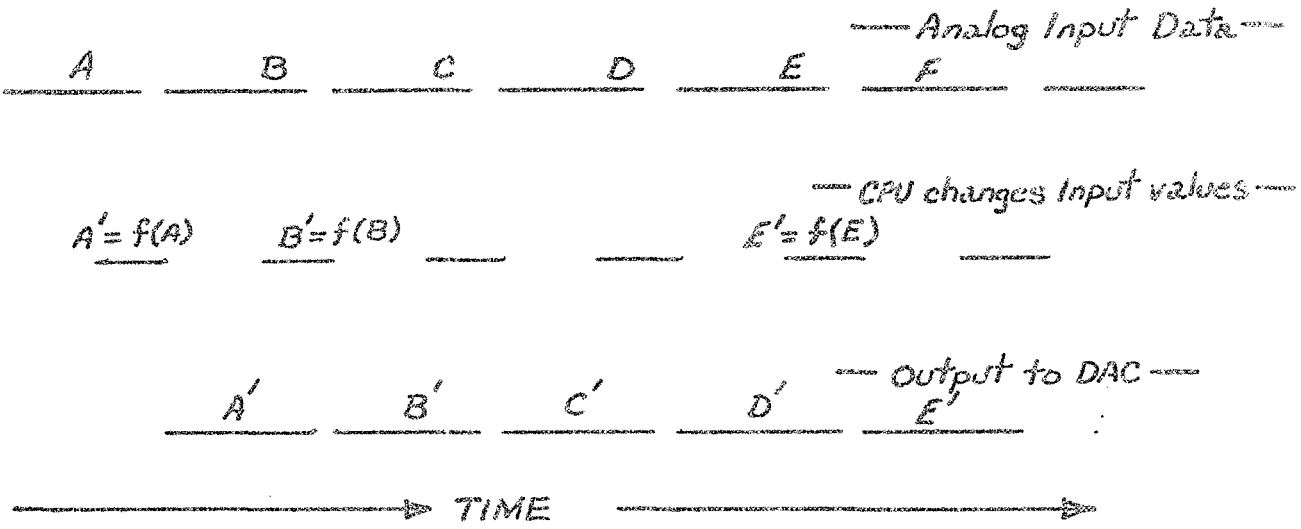
Jn - Each Jn is an integer from 0 to 31 indicating a

specific priority interrupt level to be disabled.

There will be one Jn for each level to be disabled.

DAMPS REAL TIME I/O

The nature of a Hybrid job makes essential the ability to overlap Real-Time I/O (RTIO) with CPU activity. On numerous occasions the digital computer may be receiving converted analog data, simultaneously transmitting data for analog conversion and computing new data for output based on data received. The figure below depicts such a case.



Ordinarily in FORTRAN when the statement WRITE (n,m) A,B, etc. is encountered the CPU suspends processing until this function has completed. DAMPS provides an additional channel scheduler (RTCS) dedicated to real-time devices which permits I/O activity to be started by the CPU but the CPU continues processing.

The FORTRAN programmer can use the DAMPS channel scheduler in a variety of ways. The programmer may

- Mask or unmask system channels
- Cause the execution of specific channel programs
- Determine the status of RTIO channels
- Queue channel programs if facilities are busy
- Suspend CPU activity until RTIO completes
- Cause overlap of CPU and RTIO

When the RTCS is used it must be directed to the appropriate RT Channel Program. This is accomplished through a Request Control Block which contains pertinent information regarding the location of the Channel Program, the Real Time device participating, etc. The Channel Programs themselves are constructed by subroutines which generate S/360 Channel Command Words (CCWs) in accordance with the design of the Real Time device. For example, in a particular Hybrid linkage system the subroutine READAD is used to prepare a CCW which causes the read of the analog to digital converter with numerous options. READAD can be used in two ways; as a set up program for a variety of ADC input schemes, or as an immediate builder of a specific CCW whose nature is determined during run-time. The set up method is more prevalent in a Hybrid system because I/O needs are generally known a priori and because of the time saving gained by not using CCW building routines once the RTTs and FGTs become active.

RTTs are often used to start the execution of RTIO simply by executing a CALL to the RTCS. The example problem of a subsequent section shows how this is accomplished.

FORTRAN ROUTINES FOR REAL TIME I/O

FRCBSU - FORTRAN Subroutine

The FRCBSU subroutine is used to build a Request Control Block. A Request Control Block is required for each different use of RTCS, and serves as the communication link between the program and the channel scheduler. The Request Control Block can be used by only one I/O request at a time. The contents of the block must not be changed during execution; however, if the parameters for subsequent I/O operations are identical, the RCB can be reused.

Several options for I/O operations may be designated in the RCB. The associated transaction may be queued if the device is busy, or the user may wish to have it ignored. The user may wish to treat the associated request as a burst operation. A request may also be initiated for the queueing of a certain Foreground Task upon successful completion of the I/O. The FRCBSU subroutine is executed by a FORTRAN call statement with the following format:

```
CALL FRCBSU (RCB, IUNIT1, CCWA1, IOPT1, IFGTA, NAME, IP1,
             IP2, IUNIT2, CCWA2, IOPT2, IUNIT3, CCWA3, IOPT3)
```

The parameter list is described below:

RCB - The name of the area where the RCB will be constructed and maintained. The area must be 8 full words for each RCB requested.

IUNIT1 - An integer code for the symbolic unit, (real-time device designation).

CCWA1 - The name of the area where the channel program will be located. This area will also be referenced in a call to the appropriate subroutine for construction of the channel program.

IOPT1 - An integer indicating one or more of the following options; if more than one option is being described, the value is the sum of the applicable codes. If none are applicable, the parameter should be omitted.

IOPT1 = 8 The I/O request is not to be queued by the channel scheduler if the facility is busy.

IOPT1 = 4 A second RCB is requested, for which additional parameters are required as described below. When this option is selected, eight additional full words of storage are used by the RCB.

IOPT1 = 2 The I/O request is to be a burst operation.

IOPT1 = 1 An FGT is to be queued when the I/O is completed. Additional parameters are required as described below.

The next four parameters are included only when IOPT1 is an odd integer indicating FGT queueing.

IFGTA - The name of a three full-word area where a parameter list will be constructed for queueing the FGT.

NAME - The name of the entry point of the FGT.

IP1 - An integer from 0 to 127 representing the priority of the FGT.

IP2 - An integer variable which can be used to pass a variable to the queued FGT.

The next two parameters are required if the multiple RCB option is indicated in IOPT1. The third parameter is optional unless a third RCB is requested.

IUNIT2 - An integer code for the symbolic unit.

CCWA2 - The name of the area where the channel program will be located. This area will also be referenced in a call to the appropriate subroutine for construction of the channel program.

IOPT2 - An integer code indicating the selected options. If none of the options are applicable, this parameter should be omitted.

IOPT2 = 4 Additional RCB's are requested for which additional parameters are required. When this option is selected, eight additional full words of storage are used by the RCB.

IOPT2 = 2 The input/output request is to be treated as a burst operation.

IOPT2 = 6 Both of the above options.

IUNIT3 - An integer code for the symbolic unit.

CCWA3 - The name of the area where the channel program will be located. This area will also be referenced in a call to the appropriate subroutine for construction of the channel program.

IOPT3 - An integer code indicating the an option for burst mode operation. If the option is not used, the parameter should be omitted.

IOPT3 = 2 The I/O request is a burst mode operation.

FMSK - FORTRAN Subroutine

The FMSK subroutine is used to mask one or more channels during execution of a Real Time Task. The FMSK subroutine is executed by a FORTRAN call with the following format:

CALL FMSK (RCB1, RCB2, RCB3)

The entries in the parameter list reference a previously established RCB. The channels to which the referenced devices are attached will be disabled. From one to three RCB's may be specified.

FUNMSK - FORTRAN Subroutine

The FUNMSK subroutine is used to unmask a channel during execution of a Real Time Task. The FUNMSK subroutine is executed by a FORTRAN call with the following format:

CALL FUNMSK (RCB1, RCB2, RCB3)

The entires in the parameter list reference previously established Request Control Blocks. The channels to which the indicated devices are attached will be enabled. From one to three RCB's may be specified.

FRTIO - FORTRAN Subroutine

The FRTIO subroutine is used to execute real time I/O. The FRTIO subroutine is executed by a FORTRAN call statement in the following format:

CALL FRTIO (RCB, IRET)

The parameter list is described below:

RCB - The address of the RCB which was established by a previous call FRCBSU.

IRET - An integer variable where one of the following two codes will be placed by the FRTIO subroutine. This parameter is required only if parameter IOPT1 in the call FRCBSU statement indicated that the I/O request was not to be queued.

IRET = 0 The I/O operation was initiated.

IRET = 8 The I/O operation was not initiated because the unit or channel was busy.

FCHECK - FORTRAN Subroutine

The FCHECK subroutine may be used to check the progress of a real time I/O operation. The FCHECK subroutine is executed by a FORTRAN call statement in the following format:

```
CALL FCHECK (RCB, IRET, IOPT)
```

The entries in the parameter list are defined as follows:

RCB - The name of the area containing the Request Control Block established with an FRCBSU subroutine call.

IRET - An integer variable where one of the following return codes will be placed by the FCHECK subroutine.

IRET = 0 The input/output operation came to a normal completion.

IRET = 1 A unit exception was indicated in the ending status of the input/output operation.

IRET = 4 The input/output operation has not been completed.

IRET = 8 The input/output operation was not started. A busy condition was encountered when the operation was requested and parameter IOPT1 in the FRCBSU call indicated that the request was not to be queued.

IOPT - An integer code equal to 1, designating that the FCHECK subroutine is to wait for completion of the operation before returning control to the calling program. If IOPT = 0, the subroutine will return to the main program upon completion.

EXAMPLE PROBLEM

This section shows the FORTRAN coding for an RTJ which has a program flow like the RTT-FGT configuration in the DAMPS CONCEPTS section.

The initial code, down to the CALL FDONE, sets aside arrays (save areas, RCB areas, etc.) for the DAMPS routines comprising the RTTs and FGTs; attaches RTTs to PILs; and queues FGTA.

RTT7, attached to PIL7, initiates an Analog to Digital conversion using the RTCS and building the channel program through routine READAD. RTT7 also queues FGTB.

RTT1 checks for the A to D read being complete. If A to D is complete, FGTC is queued.

RTT32 executes a channel program constructed by routine CONTRL in FGTA.

The parameters of READAD and CONTRL are not pertinent.

The reader is encouraged to review each calling sequence and to determine the use of ITEM. System units 19 and 20 are RTIO; System unit 6 is a line printer.

```
      COMMON ITEM
      COMMON/RCBS/RCBAD(8), RCBDC(8)
      COMMON/CCWS/CCWAD, CCWDC
      REAL*8 CCWAD, CCWDC, FPS1(4), FPS7(4), FPS32(4)
      DIMENSION SAV1(61), SAV7(61), SAV32(61)
      EXTERNAL RTT1, RTT7, RTT32, FGTA
      CALL FQUEUE (1, FGTA, IRC)
      IF (IRC.NE.0) CALL FCANCL
      CALL FATACH (RTT1, SAV1, 1, FPS1)
      CALL FATACH (RTT7, SAV7, 7, FPS7)
      CALL FATACH (RTT32, SAV32, 32, FPS32)
      CALL FENPIL (255, 7, 32, 1)
      CALL FRCBSU (RCBAD, 19, CCWAD)
      ITEM = 111
      CALL FDONE
      END


      SUBROUTINE FGTA
      COMMON/RCBS/RCBAD(8), RCBDC(8)
      COMMON/CCWS/CCWAD, CCWDC
      COMMON ITEM
      CALL CONTRL (CCWDC,.....)
      CALL FRCBSU (RCBDC, 20, CCWDC)
10    IF (ITEM. EQ. 111) GO TO 10
      CALL FDSPIL (0, 32)
      CALL FDONE
      END
```

```
      SUBROUTINE RTT1
      EXTERNAL FGTC
      COMMON/RCBS/RCBAD(8), RCBDC(8)
      CALL FCHECK (RCBAD, IRET)
      IF (IRET.GE.8) GO TO 10
      CALL FQUEUE (2, FGTC, IRC)
      IF (IRC.NE.0) CALL FCANCL
10    CALL FRETRN
      END


      SUBROUTINE RTT7
      EXTERNAL FGTB
      COMMON/CCWS/CCWAD, CCWDC
      COMMON/RCBS/RCBAD(8), RCBDC(8)
      REAL*8 CCWAD, CCWDC
      CALL READAD (CCWAD,.....)
10    CALL FRTIO (RCBAD, IST)
      IF (IST.EQ.8) GO TO 10
      CALL FQUEUE (5, FGTB, IRC)
      IF (IRC.NE.0) CALL FCANCL
      CALL FRETRN
      END
```

```
      SUBROUTINE RTT32
      COMMON ITEM
      COMMON/RCBS/RCBAD(8), RCBDC(8)
      CALL FRTIO (RCBDC)
      ITEM = 0
      CALL FRETRN
      END


      SUBROUTINE FGTB
      CALL FDSPIL (0, 1, 7)
      CALL FEORTJ
      END


      SUBROUTINE FGTC
      WRITE (6, 100)
100   FORMAT ('THIS INDICATES A TO D INPUT COMPLETE')
      CALL FDONE
      END
```

p

UNIVERSITY OF HOUSTON

Engineering Systems Simulation Laboratory

TO: Hybrid Users

FROM: R. L. Motard

DATE: June 18, 1970

A. Job control language for hybrid computation under the DAMPS operating system

	Card No.
// *Name* b JOB b NODUMP, *Project No.* b *User Ident.*	1
// b ACCESS b SDSABS	2
// b DELETE b SDSABS(*Myjob*)	3
// b CONDENSE b SDSABS	4
/*	5
// *Myjob* b EXEC b FORTRAN,$	6
↑	
FORTRAN Source Deck	
↓	
/*	7
// b EXEC b LNKEDT(KEEP,MAP),$	8
/*	9
/&	10
// *Name* RTJOB b NODUMP, *Project No.* b *User Ident.*	11
// SYSRDR b ACCESS b RTPROC(HYBEXEC),192='DAMPS2'	12
/*	13
// b EXEC b *Myjob*	14
↑	
Data, if any	
↓	
/*	15
/&	16

To rerun a program which has already been compiled and debugged use only the job control deck from card no. 11 on. Programs which have been compiled successfully are kept in the so-called phase library SDSABS. They will remain there until deleted by the running of the JCL cards 1 - 5 followed by a /& card.

Name ≡ any user identification or blank

Myjob ≡ name by which program is known to the system in the phase library †

b ≡ one or more blanks

Project No. ≡ assigned by ESSL upon application

User Ident. ≡ optional

B. Standard FORTRAN I/O operations

Input-output is handled in the standard FORTRAN manner, i.e., READ (5,...) and WRITE (6,...) will read cards and write on the printer. WRITE (7,...) will access the card punch unit and WRITE (15,...) will access the console typewriter which can be READ (15,...) as well. Tape I/O operations are also accessible; please check with lab personnel.

In programming hybrid I/O operations, the SYSUNI numbers have been standardized. These numbers are used in calling the FRCBSU subroutine in DAMPS to set up I/O request control blocks: control information requires the use of SYSUNI number 28, reading the A/D converter requires number 29, and writing to the D/A converters requires number 30.

Since hybrid executive subroutines will be available in three forms, the user will only be concerned with SYSUNI numbers in the unusual case where the standard unit assignments will not serve. In two of the hybrid executive modes the call to FRCBSU is embedded in the executive routine and is not required of the user; these two modes should serve most of the time.

C. Large Hybrid Programs

The job control language described in A will compile and link-edit programs that load in the background partition and are

---

† Not to be over 8 characters long and should be prefixed with user's initials.

executed as RTJOB. The maximum size program allowed is 48,890 bytes. Should the user's program exceed this size, the link-edit step needs to be changed:

1. Remove ',$' from card 8 and card 6
2. Add, between cards 8 and 9

```
b PHASE b Myjob,R                                  8a
b INCLUDE b Myjob,L                                8b
b INCLUDE card for every subroutine                8c
```

This procedure should allow the program to load in the real-time partition which provides 85,760 bytes when executed as RTJOB, i.e., cards 11-16.

'X' (Execute) SERIES OF HYBRID EXECUTIVE SUBROUTINES

Engineering Systems Simulation Lab

1. CALL XSENSE(LOCA,NB)
   LOCA = INTEGER*2 locations containing bit pattern from analog sense lines (0-15) or (0-31).
   NB = Number of bytes to be read, two or four.

2. CALL XCNTRL(LOCA,NB)

   Arguments same as above for control lines (0-15).

3. CALL XIC - set analog mode to INITIAL CONDITION.

4. CALL XPS - set analog mode to POT SET.

5. CALL XST - set analog mode to STATIC TEST.

6. CALL XHD - set analog mode to HOLD.

7. CALL XRT - set analog mode to RATE TEST.

8. CALL XOP - set analog mode to OPERATE.

9. CALL XPOTSS(NAME,IVAL) - used to set value of one pot.
   NAME = four character name of pot - P123
   IVAL = value in range of 0-9999 to which pot is to be set.

10. CALL XRDAD(N,ICON,SA,EA,LOCA) - used to do a sequential analog to digital read.
    N = number of multiplexers to be read.
    ICON = mode of A/D conversion.
    SA = Address of first MUX, INTEGER mode.
    EA = Address of last MUX to be read, INTEGER mode.
    LOCA = INTEGER*2 vector to receive read values. the first cell, LOCA(1) is reserved for the system.

11. CALL XWRTDA(N,ICON,SA,EA,LOCA) - used to write sequential digital values to the analog DAC's (digital to analog converters). Arguments same as XRDAD.

12. CALL XRDAN(NAME,N,LOCA) - used to read the value of any analog component. Up to 100 may be read in one call.
    NAME = name or vector of component names in four character format to be read. - P110, A005.
    N = number of components to be read.
    LOCA = INTEGER*2 area dimensioned the size of N to hold the read analog values.

13. XSTCTL(IHOLD,LINE) - set one control line at the logic patch panel (lines 0-15).
    IHOLD = Two byte integer which saves the current status of the control lines and is updated by each entry to XSTCTL.
    LINE = control line number to be set.

14. XRSCTL(IHOLD,LINE) - same as XSTCTL except that control line is reset.

15. XRDSNS(LINE) - logical function which returns
    .TRUE. if sense line, LINE, is set or
    .FALSE. if the sense line is reset for sense lines 0-31.

ted in U.S.A.

70-BHF-17

an ASME publication

The Society shall not be responsible for statements or opinions advanced in papers or in discussion at meetings of the Society or of its Divisions or Sections, or printed in its publications.

Discussion is printed only if the paper is published in an ASME journal or Proceedings.

Released for general publication upon presentation

$2.00 PER COPY
$1.00 TO ASME MEMBERS

A Dynamic Model of the Circle of Willis

J. C. CHAO
Graduate Assistant
N. H. C. HWANG
Associate Professor
Cullen College of Engineering,
University of Houston,
Houston, Texas

A basic pulsatile-flow model of the circle of Willis at the base of the brain was built with integrated lumped circuits. The nonsteady Navier-Stokes equation and one other equation which governs the radial movement of the distensible walls were applied to each of the 22 consecutive short segments into which the circle of Willis was subdivided. The segments were so arranged that the regions where circulation would be influenced by local increase of resistance or altered field of blood supply were located at junctions of the segments. Direct measurement of intravascular blood pressure from a dog's carotid artery was taken to test the model performance. The computer model compared favorably with the data measured from the animal prototype.

Contributed by the Biomechanical & Human Factors Division of The American Society of Mechanical Engineers for presentation at the ASME Biomechanical & Human Factors Conference, Washington, D. C., May 31 - June 3, 1970. Manuscript received at ASME Headquarters, March 16, 1970.

Copies will be available until March 1, 1971.

A Dynamic Model of the Circle of Willis

J. C. CHAO **N. H. C. HWANG**

INTRODUCTION

Aneurysm on cerebral arteries is a matter of acute concern to neurologists and neurosurgeons alike. Under most circumstances, the aneurysm(s) is (are) found on or near the vicinity of the circle of Willis. Treatment of the cerebral aneurysm has been mostly surgical, either by direct approach to the aneurysm through craniotomy or by elective occlusion of one or more of the afferent cervical arteries.

In the hands of most surgeons, the surgical procedure of cervical artery occlusion carries less risk than the direct intracranial approach to the aneurysm. However, because of the uncertainty of post-operative flow pattern and the pressure distribution in the brain arterial system, it has frequently led to recurrent hemorrhages from the aneurysm(s) for which occlusion was performed and other post-operative bleeding episode. The comparative values of the cervical arterial occlusion treatments has remained a controversial subject.

The basic purpose of cervical artery occlusion is to reduce the local pressure in the weakened portion of the arterial wall in which the aneurysm resides. Therefore, it is extremely important to be able to predict preoperatively the post-occlusive pressure distribution in every part of the brain artery system.

Past attempts to model the circle of Willis have dealt entirely with the study of the much simplified steady flow in rigid tubes using a fluid model (1-4),[1] an electric analog model (5-7) and a computer model (7). Most of these models were comparatively studied in the form of their computer representation recently by Clark, et al. (8). This paper will present a basic model of the circle of Willis with pulsatile flow in distensible tubes.

CONSTRUCTION OF THE MODEL

The measurement of major cerebral arteries of a group of dogs as quoted in reference (4) have been taken as the prototype for our studies. The prototype circle was subdivided into 22 segments of approximately equal volumes. Some irregularities of the sectioning are introduced from the need to have the regions where circulation would be influenced by local increase of resistance or

[1] Underlined numbers in parentheses designate References at end of paper.

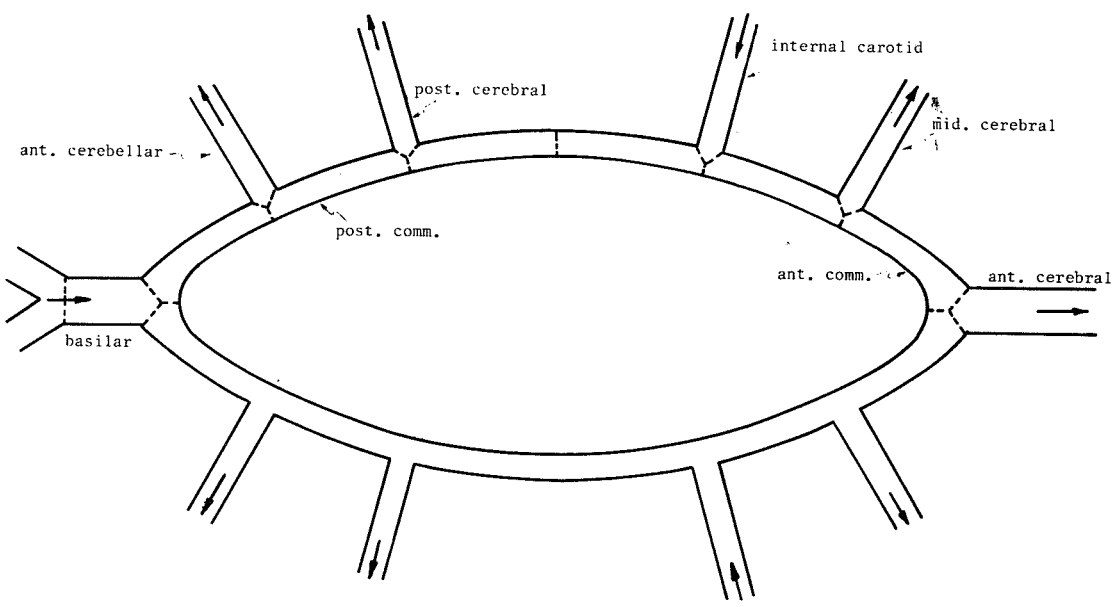


Fig.1 Sketch of the mathematical model

altered field of blood supply coincide with junctions of segments. The mathematical model is schematically shown in Fig.1.

Within each of the 22 segments, blood flow is assumed to be approximately laminar and incompressible. Disregarding body forces and the small tangential motion of the blood, the simplified Navier-Stokes equations in the cylindrical coordinate form can be written as:

$$\frac{\partial p}{\partial z} = -\rho\frac{\partial u}{\partial t} + \mu[\frac{1}{r}\frac{\partial u}{\partial r} + \frac{\partial^2 u}{\partial r^2}] \quad (1)$$

$$\frac{\partial p}{\partial r} = -\rho\frac{\partial w}{\partial t} + \mu[\frac{1}{r}\frac{\partial w}{\partial r} + \frac{\partial^2 w}{\partial r^2} - \frac{w}{r^2}] \quad (2)$$

Here p is pressure, z is the distance along the axis, ρ is fluid mass density, μ is the ordinary coefficient of viscosity, and u and w are longitudinal and radial velocity components, respectively.

If we further assume that the pressure is independent of r, a reasonable assumption for small radial velocity, equation (2) can be neglected initially. The value of w may then be determined from wall elasticity using the continuity relation.

Equation (1) may be converted from differential to difference form in the radial space dimensions by breaking up the segment radially into N concentric annular shells of equal thickness of Δr for all but the outmout annulus, which was chosen to have thickness $\Delta r/2$. Applying the simplest case, N = 2, Rideout and Dick (9) have converted equation (1) to the following form:

$$p_o - p_i = -\frac{9\rho\Delta z}{4\pi R^2}\frac{dq_i}{dt} - \frac{81\mu\Delta z}{8\pi R^4}q_i \quad (3)$$

Here Δz is the length of the segment, R is the tube radius, and q is the longitudinal flow rate. The subscripts i and o indicate the input and output at the entrance and the exit of the segment, respectively.

The coefficient of the second term on the right-hand side of equation (3) gives the fluid resistance of the segment,

$$R = \frac{81\mu\Delta z}{8\pi R^4}, \quad (4)$$

while the coefficient of the first term on the right-hand side of equation (3) gives the fluid inductance,

$$L = \frac{9\rho\Delta z}{4\pi R^2} \quad (5)$$

At the wall a radial velocity (w_R) is assumed. Here Womersley (10) and Jager, et al., (11) have shown

$$p = \rho h\frac{dw_R}{dt} + \beta w_R + \frac{Eh}{R^2(1-\sigma^2)}\int_0^t w_R dt \quad (6)$$

in which ρ is the wall density, β is the wall damping coefficient, E is Young's modulus of the wall, σ is the Poisson ratio of the wall, and h is wall thickness.

As discussed in reference (9), the first two terms in equation (6) can be ignored because radial velocities are small. Equation (6) then becomes

$$p = \frac{Eh}{R^2(1-\sigma^2)}\int_0^t w_R dt \quad (7)$$

Let the input and output flow at each end of the Δz length of the nth annulus be designated as q_{ni} and q_{no}, then the continuity relates such flows and the radial velocity at r = R as

$$\sum_{n=1}^{N-1}(q_{ni} - q_{no}) = 2\pi R w_R \Delta z \quad (8)$$

From equations (7) and (8),

$$p = \frac{Eh}{2\pi R^3(1-\sigma^2)\Delta z}\int_0^t \sum_{n=1}^{N-1}(q_{ni} - q_{no})dt \quad (9)$$

For the case of N=2, we have,

$$p = \frac{Eh}{2\pi R^3(1-\sigma^2)\Delta z}\int_0^t (q_{ni} - q_{no})dt \quad (10)$$

The fluid capacitance for the segment of length (Δz) is, therefore,

$$C = \frac{2\pi R^3(1-\sigma^2)\Delta z}{Eh} = \frac{3\pi R^3\Delta z}{2Eh} \quad (11)$$

take $\sigma = 1/2$.

Measuring the dimensions in CGS units and converting the blood pressure from the conventional unit of mm Hg to dyne/sq cm by 1 mm Hg = 1332 dyne/sq cm, the values of R, L, and C are calculated for each of the 22 segments. The values are shown in Table 1.

As shown in Fig.1, all the efferent branches are each represented by two segments in series. The distal part is accounted for 80 percent of the

Table 1 Characteristics of Arterial Segments

Segment	Radius (cm)	Length (cm)	R $(\frac{10^3 \text{dyne/cm}}{\text{ml/sec}})$	L $(\frac{10^3 \text{dyne/cm}^2}{\text{ml/ sec}})$	C $(10^{-8} \text{ml/dyne/cm})$
Basilar	.05	4.2	68.6	1.130	5.88
Post comm. 1	.035	.50	42.8	.310	.202
Post comm. 2	.035	.25	21.4	.156	.101
Post comm. 3	.035	.36	30.8	.223	.146
Post comm. 4	.035	.36	30.8	.223	.146
Ant. comm. 1	.042	.25	10.4	.107	.175
Ant. comm. 2	.042	.66	27.4	.244	.460
Ant. cerebellar	.032	.73	89.2	.536	.226
Post cerebral	.039	.41	22.7	.202	.228
Int. carotid	.06	4.2	41.6	.880	8.56
Mid. cerebral	.048	.81	19.6	.265	.841
Ant. cerebral	.06	1.12	10.4	.220	2.14

pressure drop. Hence, the lower resistance is four times the upper resistance; the lower compliance is six times the upper compliance.

Applying relations (4), (5), and (11) to a segment (m), which is located between sections (m) and (m+1), equations (3) and (10) can be rewritten as follows:

$$q_m = \frac{1}{L_m} \int_0^t [(p_{m-1} - p_m) - R_m q_m] dt \qquad (12)$$

and

$$p_m = \frac{1}{C_m} \int_0^t (q_m - q_{m+1}) dt \qquad (13)$$

The analog computer setup for the segment is shown in Fig.2.

The entire model including its input function, as mentioned in the next paragraph, has been set on the SS-100 Analog/Hybrid Computer in the Cullen College of Engineering.

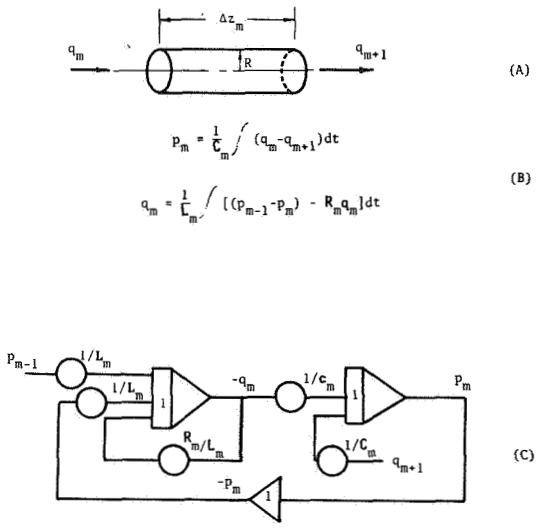
TESTING OF THE MODEL

Intravascular pulse pressure directly measured from a dog's right common carotid artery (12) was taken to test the model performance. At this preliminary phase of the study,[2] several assumptions, which had been commonly made for almost all previous models (1-5,7), were accepted to test our model. These assumptions are as follows: 1) the total flow through the system is estimated at 63 cc/100 gm of dog brain per minute, 2) the afferent flow is divided equally among the two carotids and the basilar, 3) the efferent flow is distributed in accordance with the weight of brain irrigated by each vessel, and 4) the input pressure at all three afferent arteries was the same function of time and there is no phase delay.

Pulse pressure and flow rate at several sections on the model were measured with a Polaroid camera which photographed the traces on the screen of a Textronix dual-beam oscilloscope. The measured pressure and flow rated at each of these sections are shown as functions of time in Figs.3 and 4, respectively.

Comparisons of the model output with the animal data (12,13) were rather encouraging because the relative pressure drop and flow distribution at each of the junctions were comparable

[2] A hybrid computer phase of the model with delayed individual input function for each of the three afferent arteries will be reported in subsequent papers.



(A) Arterial segment
(B) Integral equations representing (A)
(C) Analog computer setup for (A)

Fig.2

both in magnitude and wave forms. Although a point-to-point comparison was not made in this study due to the fact that different animals were involved, the similarity in trends indicated the possibility of satisfactory refinement of the modeling technique. The comparison of the changes in flow rate and pressure with the previous models was not possible since the flows in all other models were steady in nature. However, the ranges of magnitude showed good agreement between our model and the previous models.

CONCLUSION

The advantages of analog modeling of a complex circulation system, such as the circle of Willis at the base of the brain by integrated lumped circuits, are apparent. Not only does it offer a far greater versatility than a conventional fluid-model, but it also offers the convenience of recording the pressure and flow rates at almost any desired sections in the system; the ease of finding values of functions with varying parameters; and most of all, it permits the possibility of including the nonlinear terms.

The assumption of laminar flow in cerebral arteries may be regarded as a better approximation than is made for larger vessels (such as the aorta) because of the apparently smaller tube diameter and slower longitudinal velocities (14).

Study of brain circulation after various cervical arterial occlusions was not included in this paper. However, it is planned for our next stage of investigation. A hybrid computer extension of the present model is now under construction in our laboratory. In the model, the analog representation described in this paper will be combined with a digital representation of some of the slower acting control loops, such as the peripheral resistance and phase delay of pressure waves in different afferent arteries. Better conformity of model to animal prototype can be expected.

ACKNOWLEDGMENT

The assistance of Drs. Williamina A. Himwich and T. Iwabuchi in furnishing some of the animal data is greatly appreciated.

This work was partially supported by NASA Grant NGL-44-005-084.

REFERENCES

1 Kramer, S. P., "On the Function of the Circle of Willis," Journal of Experimental Medicine, Vol. 15, 1912, pp. 348-363.

2 Rogers, L., "The Function of the Circlus Arteriosus of Willis," Brain, Vol. 70, 1947, pp. 171-178.

3 Avman, N., and Berling, E. A., Jr., "A Plastic Model for the Study of Pressure Changes in the Circle of Willis and Major Cerebral Arteries Following Arterial Occlusion," Journal of Neurosurgery, Vol. 18, 1961, pp. 361-365.

4 Himwich, W. A., et al., "The Circle of Willis as Simulated by an Engineering Model," Arch. Neurol., Vol. 13, 1965, pp. 164-172.

5 Murray, K. D., "Dimensions of Circle of Willis and Dynamic Studies Using Electric Analog," Journal of Neurosurgery, Vol. 21, 1964, pp. 26-34.

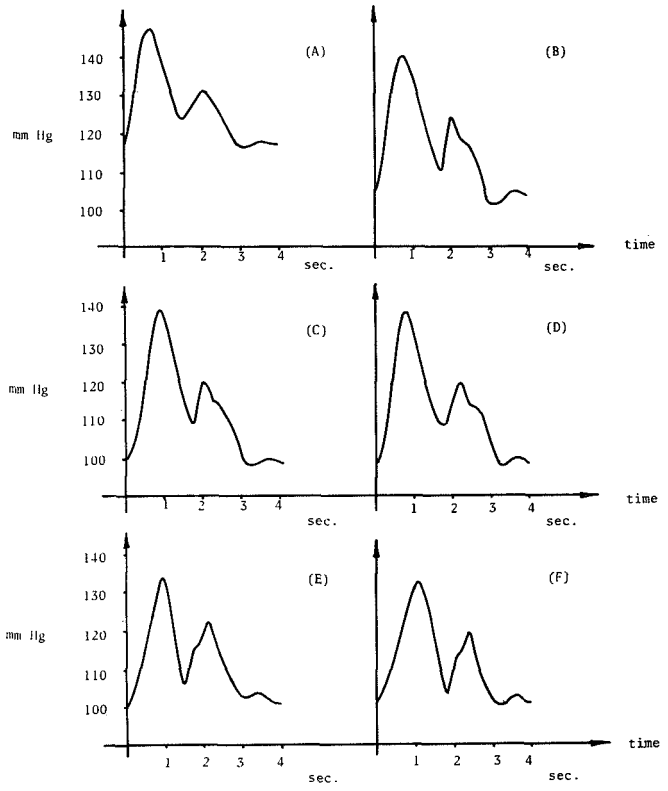
6 Fasano, V. A., Portalupi, A., and Broggi, G., "Aspetti Fisici Della Circolazione Nei Vasi Anastomotici Endocranici Basali e Meningo-Cortical: (Studio su un Modello Elettrico Analogico)," Sist. Nerv., Vol. 16, 1964, pp. 271-307.

7 Clark, M. E., et al., "Engineering Analysis of the Hemo-dynamics of the Circle of Willis," Arch. Neurol., Vol. 13, 1965, pp. 173-182.

8 Clark, M. E., Himwich, W. A., and Martin, J. D., "A Comparative Examination of Cerebral Circulation Models," Journal of Neurosurgery, Vol. 29, 1968, pp. 484-494.

9 Rideout, V. C., and Dick, D. E., "Difference-differential Equations for Fluid Flow in Distensible Tubes," IEEE Transactions Bio-Med. Engr., BEM-14, 1967, pp. 171-177.

10 Womersley, J. R., "An Elastic Tube Theory for Pulse Transmission and Occillatory Flow in Mammalian Arteries," Wright Air Div. Rept., WADC-TR 56-614, 1957.



(A) Input pressure function at the junction of vertebral and basilar

(B) Pressure at the junction of post. comm. and basilar

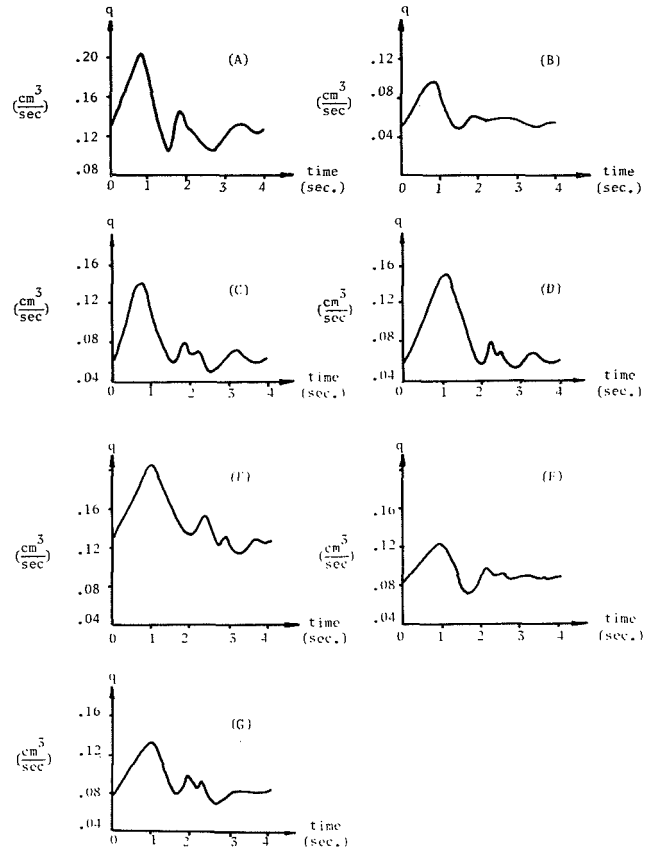
(C) Pressure at the junction of post. comm. and ant. cerebellar

(D) Pressure at the junction of post. comm. and post. cerebral

(E) Pressure at the junction of ant. comm. and mid. cerebral

(F) Pressure at the junction of ant. comm. and ant. cerebral

Fig.3 Pressure distribution at arterial junctions



(A) Flow rate at the junction of basilar and post. comm. (in the direction of post. comm. artery)

(B) Flow rate at the junction of post. comm. and post. cerebral (in the direction of post. cerebral)

(C) Flow rate at the junction of post. comm. and post. cerebral (in the direction of post. comm.)

(D) Flow rate at the mid. point of the post. comm. (between carotid and post. cerebral)

(E) Flow rate at the junction of internal carotid and comm. (in the direction of anterior comm.)

(F) Flow rate at the junction of mid. cerebral and anterior comm. (in the direction of mid. cerebral)

(G) Flow rate at the junction of anterior comm. and anterior cerebral (in the direction of anterior cerebral)

Fig.4 Flow rate distribution in the circle of Willis

11 Jager, G. N., Westerhof, N., and Noordergraaf, A., "Oscillatory Flow Impedance in Electrical Analog of Arterial System," Circ. Res., Vol. 16, 1963, pp. 121-133.

12 Himwich, W. A., and Spurgeon, H. A., "Pulse Pressure Contours in Cerebral Arteries," Acta Neurol. Scandinav., Vol. 44, 1968, pp. 43-56.

13 Personal communication with W. A. Himwich.

14 Hwang, N. H. C., and Chao, J. C., "Turbulent Characteristics of Arterial Flow," Proceedings of the Annual Conference on Engineering in Medicine and Biology, Vol. 10, 1968, 16.5.

AN ANALOG COMPUTER STUDY
OF MULTIVARIABLE MODEL SIMPLIFICATION

A Thesis

Presented to

the Faculty of the Department of Electrical Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

---

by

Terry Edward Edgerton

December 1970

AN ANALOG COMPUTER STUDY

OF MULTIVARIABLE MODEL SIMPLIFICATION

Terry E. Edgerton

Approved:

Director and Chairman

of the Committee:

C. F. Chen

Associate Advisor:

William P. Schneider

Associate Advisor:

R. L. Motard

Dean, College of

Engineering

C. V. Kirkpatrick

ACKNOWLEDGEMENT

This work was supported in part by National Science Foundation Grant GK2436 and National Aeronautics and Space Administration Grant N44-005-084.

TABLE OF CONTENTS

CHAPTER		PAGE
I.	INTRODUCTION	1
	The Various Techniques	1
	The Most Flexible Technique	2
II.	DISCUSSION OF THE VARIOUS SIMPLIFICATION TECHNIQUES	3
	The Method Based on Dominant Roots	3
	The Method Based on the Fractional Form	9
III.	DISCUSSION OF THE SECOND CAUER CONTINUED FRACTION FORM	10
	Development of the Second Cauer Form	11
	Implementation of the Second Cauer Form	13
IV.	PRESENTATION OF SINGLE INPUT SINGLE OUTPUT EXAMPLES	18
V.	THE CAUER MODEL FOR MULTIVARIABLE SYSTEMS	26
	Development of the Cauer Model	26
	Development of the Cauer Model	28
	Analog Program for Multivariable Systems	28
VI.	PRESENTATION OF TWO INPUT-TWO OUTPUT EXAMPLE	33
VII.	CONCLUSIONS	46
BIBLIOGRAPHY		48
APPENDIX		49

LIST OF FIGURES

FIGURE		PAGE
1.	Pole Zero Plot of Equation 1	4
2.	Block Diagram of Equation 1	6
3.	Analog Program of Equation 1	7
4.	Transient Response for Equations 2, 2A, 2B	8
5.	Block Diagram of Equation 7	12
6.	Analog Program for Equation 7	12
7.	Block Diagram of Equation 9	14
8.	Analog Program for Equation 9	14
9.	Block Diagram of Equation 11	15
10.	Analog Program for Equation 11	17
11.	Analog Program for First Example	19
12.	Response Curves for First Example	21
13.	Response Curves for First Example	21
14.	Analog Program for Second Example	24
15.	Response Curves of the Second Example	25
16.	Response Curves of the Second Example	25
17.	Block Diagram of Equation 24	29
18.	Block Diagram of Equation 22	30
19.	Basic Program Block for Two Input/Output System	31
20.	Interconnection Diagram of Two Input/Two Output System with Terms Through H_{4ij}	32
21.	Analog Program for Third Example	35-38

FIGURE PAGE

22. C_1 Responses for Example 3 ($R_1=R_2=\frac{-50}{S}$) 40

23. C_2 Responses for Example 3 ($R_1=R_2=\frac{-50}{S}$) 40

24. C_2 Responses for Example 3 ($R_1=R_2=\frac{-50}{S}$) 41

25. C_1 Responses for Example 3 ($R_1=\frac{-50}{S}$, $R_2=\frac{-70}{S}$) 42

26. C_2 Responses for Example 3 ($R_1=\frac{-50}{S}$, $R_2=\frac{-70}{S}$) 42

27. C_1 Responses for Example 3 ($R_1=-28.4$, $R_2=\frac{-50}{S}$) 43

28. C_2 Responses for Example 3 ($R_1=-28.4$, $R_2=\frac{-50}{S}$) 43

CHAPTER I

INTRODUCTION

I. THE VARIOUS TECHNIQUES

In classical techniques, there are several methods for simplifying a high order system with a lower order one. Namely:

(1) The Nixon method [1] - by ignoring certain coefficients.

(2) The Dominant Roots method [2] - by ignoring certain roots.

(3) The K. Chen method [3] - by ignoring certain parts of a Bode diagram.

In the modern techniques, there are also several methods for reducing a high dimensional state space into a lower dimensional one. Namely:

(4) The Davison method [4] - emphasizing on dominant eigenvalues.

(5) The Marshall method [5] - emphasizing on the steady state values.

(6) The Chen and Shieh method [6] - emphasizing on the later part of a transient curve.

II. THE MOST FLEXIBLE TECHNIQUE

In the six methods mentioned, the Chen and Shieh method [6] is the most flexible because, by using it one can either work in the state space or work in the transfer function (s) domain.

The Chen and Shieh method [6] is essentially the second Cauer form of the continued fraction expansion; however, the interpretation of the second cauer form is quite different from the one used in circuit theory. The method can be summarized as follows:

A. Put the denominator and the numerator of the transfer function into ascending order.

B. Expand the transfer function into continued fraction form.

C. Discard certain quotients.

D. Invert the simplified continued fraction into a transfer function which is the required simplified transfer function.

The procedure is extremely simple and the results are extremely satisfactory; however, Chen and Shieh's method [6] is limited to single input - single output systems. In other words, the method needs to be extended to multi-input/multi-output systems so that the method will be more versatile. This paper will use the analog computer to investigate the multi-input/multi-output cauer form. Theoretically, this is a natural extension of the scalar cauer form.

CHAPTER II

DISCUSSION OF THE VARIOUS SIMPLIFICATION TECHNIQUES

The six existing simplification methods can be regrouped into two schools.

(A) the school based on dominant roots and (B) the school based on fractional form. Among the methods mentioned in Chapter I, (2), (4) and (5) belong to (A); while (1), (3) and (6) belong to (B). The group (A) is good for certain systems and the group (B) is good for other systems. As far as control systems are concerned, group (B) seems to be more powerful.

I. THE METHOD BASED ON DOMINANT ROOTS

Group (A) can be best illustrated by a simple example.. Consider the transfer function

$$F(s) = \frac{S^3 + 52S^2 + 102S + 100}{S^5 + 35S^4 + 396S^3 + 1538S^2 + 3048S + 1872} \qquad (1)$$

Which factor as follows

$$F(s) = \frac{(S+50)\ (S+1+J1)\ (S+1-J1)}{(S+1)\ (S+2+J2)\ (S+2-J2)\ (S+15+J3)\ (S+15-J3)} \qquad (1A)$$

The pole-zero form of the transfer function is shown in (1A) and can be expressed in the S-plane as shown in Figure 1. In the pole-

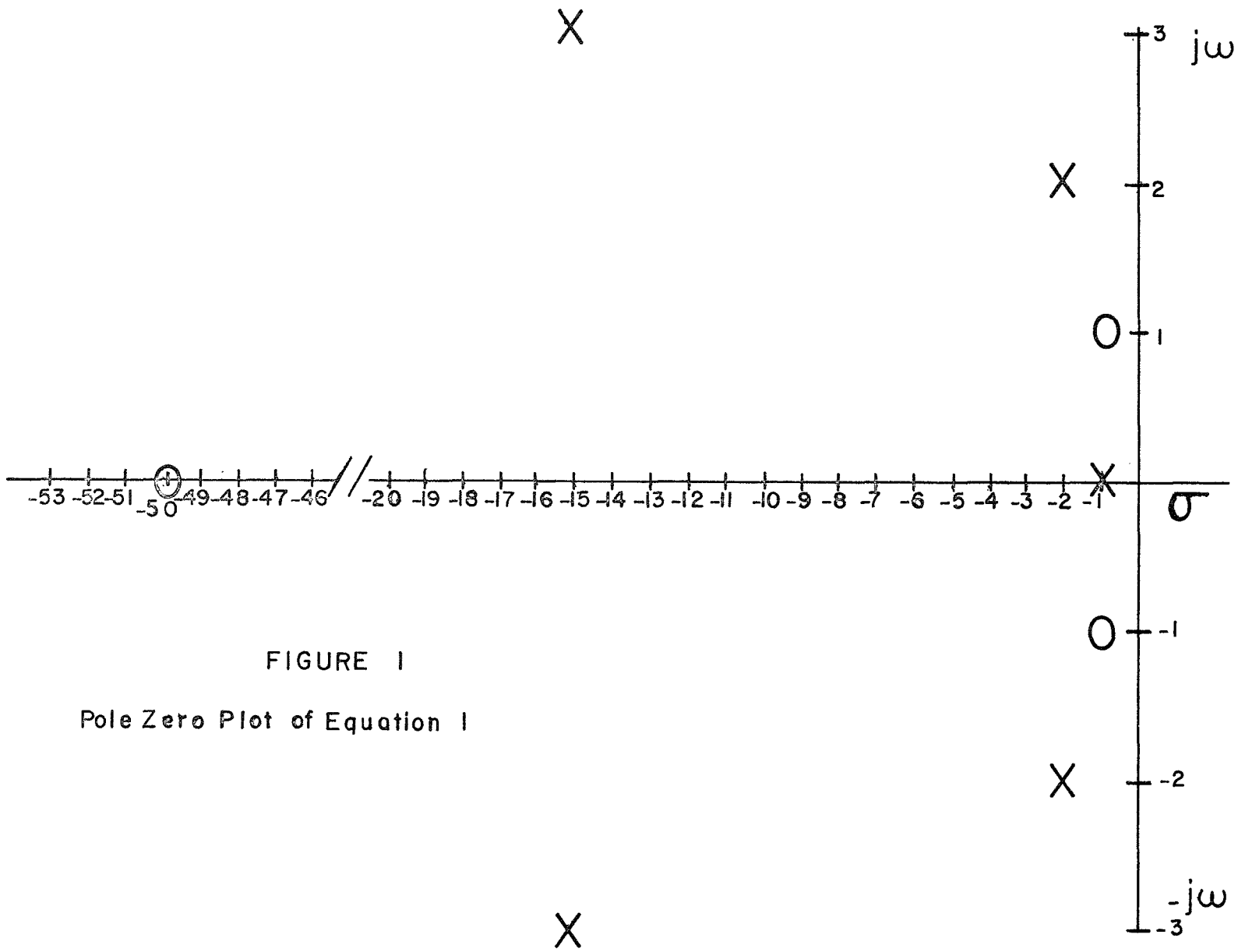


FIGURE 1

Pole Zero Plot of Equation 1

zero configuration, one knows that the poles near the imaginary axis have the most influence on the system's behavior. Based on this well known knowledge, one can divide the poles into three groups and write the transfer function as follows:

$$F(s) = \frac{1}{S+1} \cdot \frac{S^2 + 2S + 2}{S^2 + 4S + 8} \cdot \frac{S + 50}{S^2 + 30S + 234} \qquad (1B)$$

The corresponding block diagram is shown in Figure 2. It is straight forward to change the block diagram into an analog computer program as shown in Figure 3.

When we study the behavior of the system using the dominant root method, one simulates the transfer function by one of the following:

$$F(s) = \frac{1}{S+1} \cdot .05342 \qquad (2)$$

$$F(s) = \frac{1}{S+1} \quad \frac{S^2 + 2S + 2}{S^2 + 4S + 8} \quad .2137 \qquad (2A)$$

$$F(s) = \frac{1}{S+1} \cdot \frac{S^2 + 2S + 2}{S^2 + 4S + 8} \quad \frac{S + 50}{S^2 + 30S + 234} \qquad (2B)$$

The two approximations (2 and 2A) can be realized by disconnecting, first the last stage parts and then disconnecting the second stage of the analog computer program. The three different responses are shown in Figure 4.

This method is not very satisfactory in many senses. Generally the approximations have poor transient response. It is well known that in control system analysis one of the most important considerations is

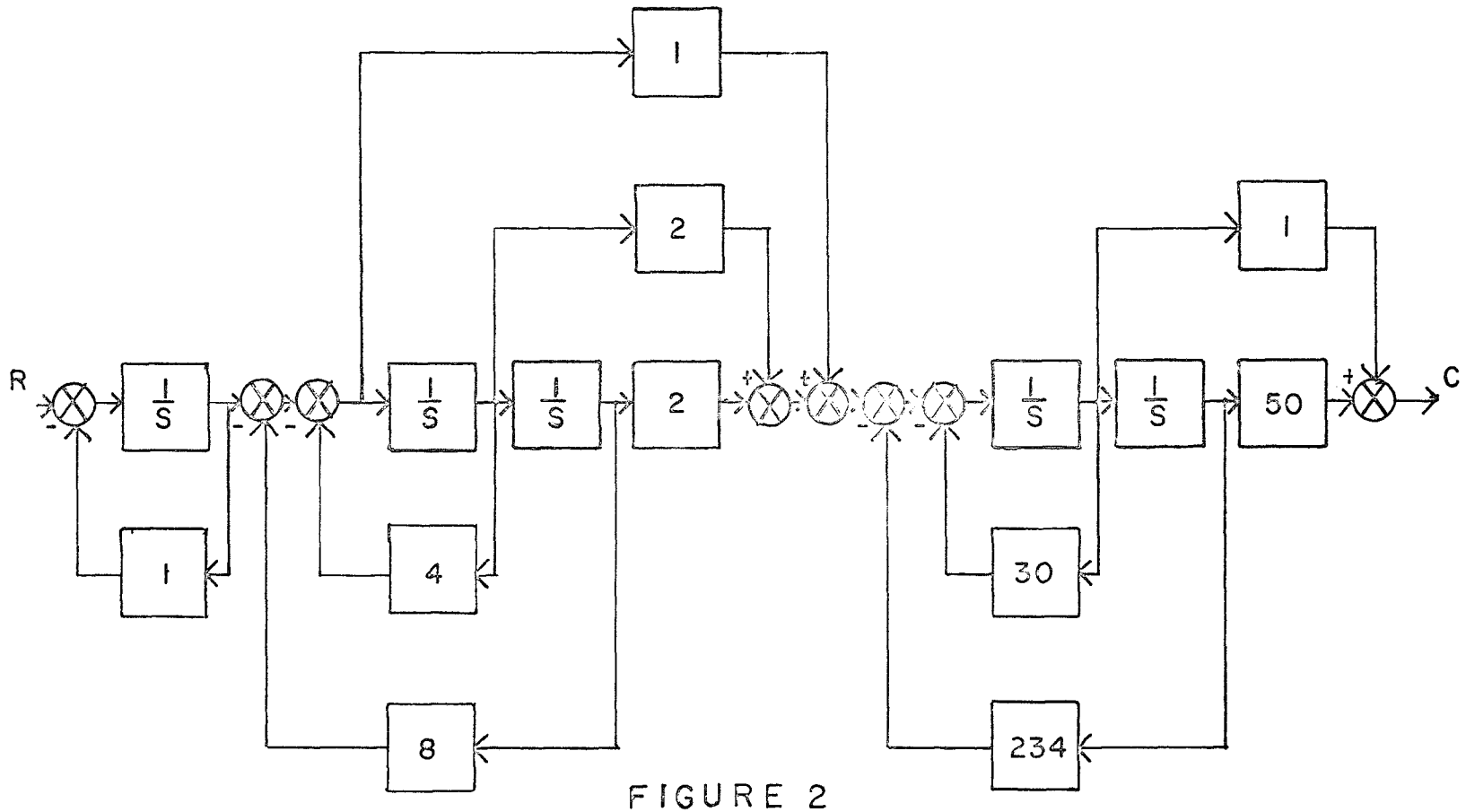


FIGURE 2

Block Diagram of Equation 1.

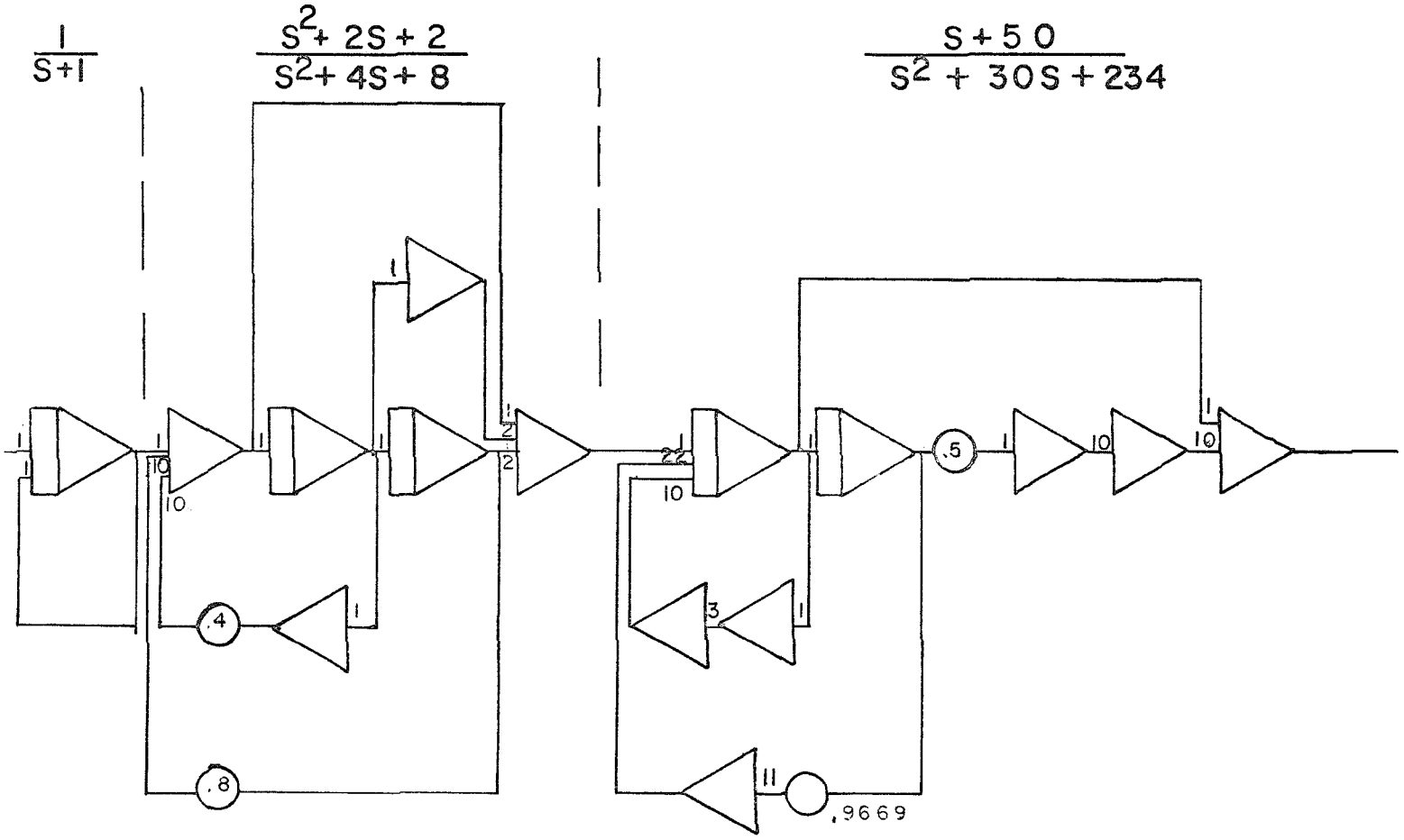


FIGURE 3

Analog Program for Equation 1

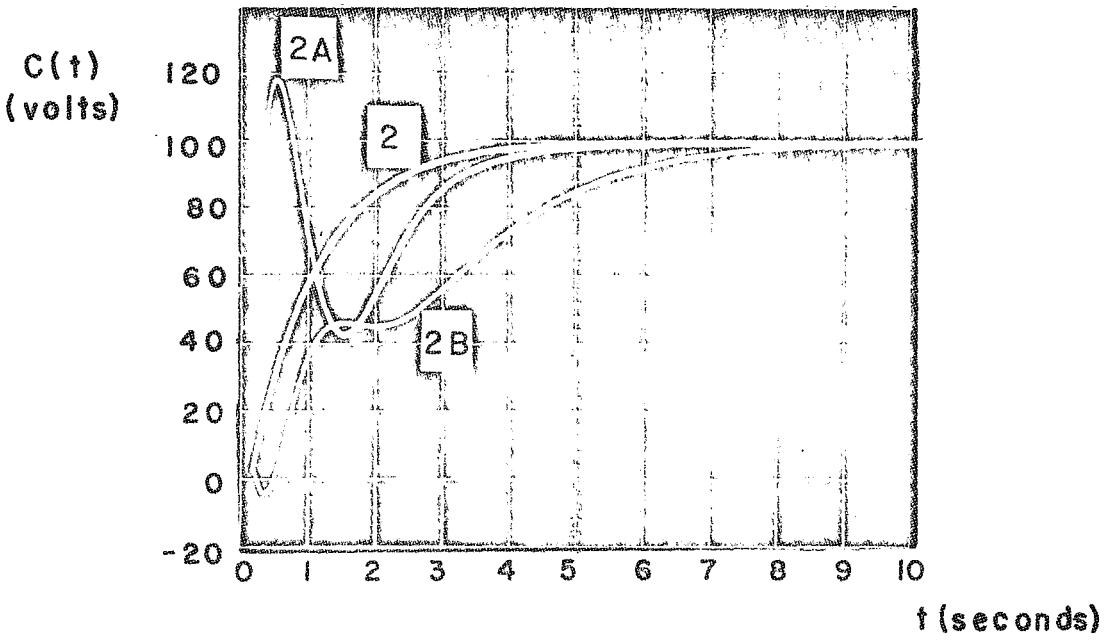


FIGURE 4

Transient Response for Equations

2, 2A, and 2B

transient response.

By observing the response curves of Figure 4 one can quickly see that the approximation, using equation 2, is not even close to the total, but it is much better than the approximation using equation 2A. It is interesting to note that equation 2A was a higher order approximation than equation 2.

This example readily points to the pitfalls of the pole-zero reduction techniques.

II. THE METHOD BASED ON THE FRACTIONAL FORM

K. Chen [3] was one of the first to realize that the dominant roots method has shortcomings. He uses a completely different approach by considering a transfer function as a feedback system and then attempting to simplify the open loop transfer function. In other words, he decomposes the system into a feedback system of the form

$$F(s) = \frac{1}{1 + G(s)} \tag{3}$$

where G(s) is the open loop transfer function. G(jw) is then plotted in Bode diagram. K. Chen considered that any poles above +15db or below -15db can be ignored. K. Chen felt the importance of the feedback loop, but he restricted himself to systems with unity feedback. Consequently, his approach is only good for the unity feedback system.

CHAPTER III

DISCUSSION OF THE SECOND CAUER CONTINUED FRACTION FORM

C. F. Chen and Shieh [6] extended and modified K. Chen's work [3] by the following procedure. Consider a general transfer function

$$T(s) = \frac{b_1 s^{n-1} + b_2 s^{n-2} + \cdots + b_n}{s^n + a_1 s^{n-1} + a_2 s^{n-2} + \cdots + a_n} \tag{4}$$

First of all, the two polynomials of the transfer function should be arranged in ascending order as

$$T(s) = \frac{b_n + b_{n-1} s + \cdots + b_1 s^{n-1}}{a_n + a_{n-1} s + \cdots + s^n} \tag{4A}$$

then expanded into a continued fraction as follows:

$$T(s) = \cfrac{1}{H_1 + \cfrac{1}{\frac{H_2}{s} + \cfrac{1}{H_3 + \cfrac{1}{\frac{H_4}{s} + \ddots + \cfrac{1}{\frac{H_{2n}}{s}}}}}} \tag{4B}$$

Next truncate (4B) by discarding certain quotients. For example,

$$T(s) \simeq \cfrac{1}{H_1 + \cfrac{1}{\frac{H_2}{s}}} = \frac{H_2}{H_1 H_2 + s} \tag{5}$$

This is the simplest reduction of (4B).

Or, one can approximate (4) by

$$T(s) \simeq \cfrac{1}{H_1 + \cfrac{1}{\cfrac{H_2}{S} + \cfrac{1}{H_3 + \cfrac{1}{\cfrac{H_4}{S}}}}} = \tag{6}$$

$$\frac{H_2H_3H_4 + (H_2 + H_4)S}{H_1H_2H_3H_4 + [H_1(H_2 + H_4) + H_3H_4]S + S^2}$$

I. DEVELOPMENT OF THE SECOND CAUER FORM

The form of a general transfer function is

$$\frac{C}{R} = \frac{G}{1 + GH} \tag{7}$$

which, after rearranging, is

$$\frac{C}{R} = \frac{1}{H + \frac{1}{G}} \tag{7A}$$

Equation (4B) is a natural generalization of (7A). The block diagram for (7A) is presented in Figure 5.

Now, letting $G = -1/s$ and $H = -a$ produces the transfer function

$$F(s) = \frac{C}{R} = \frac{-1}{s + a} \tag{8}$$

which has the analog program shown in Figure 6.

To complete the basics in converting fraction expansion form

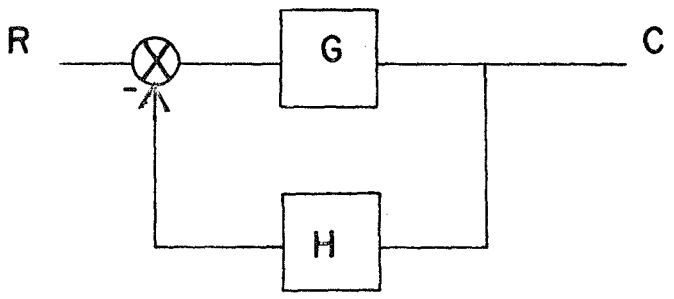


FIGURE 5

Block Diagram of Equation 7

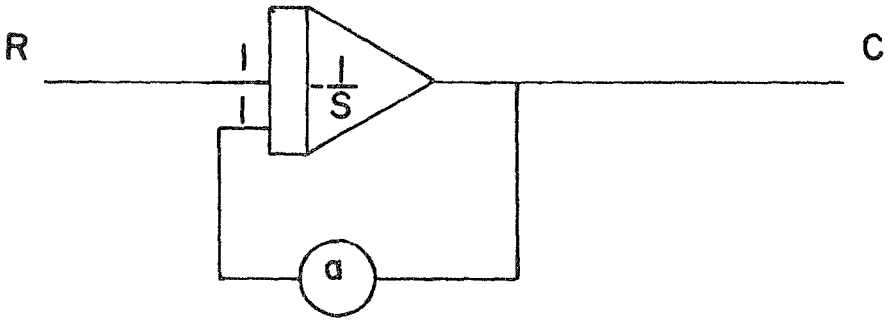


FIGURE 6

Analog Program for Equation 7

to an analog program, the last block is a feed-forward loop. This can be illustrated with the transfer function

$$\frac{C}{R} = \frac{G_1 + F_1}{1 + (G_1 + F_1)H} \tag{9}$$

which, after rearranging, is

$$\frac{C}{R} = \cfrac{1}{H + \cfrac{1}{G_1 + F_1}} \tag{10}$$

The block diagram of equation 9 is shown in Figure 7.

Again, let $G = -1/s$ and $H = -a$, and let $F_1 = -F(s)$ then the analog program is as shown in Figure 8.

II. IMPLEMENTATION OF THE SECOND CAUER FORM

In general, the form of the second Cauer continued fraction is

$$\frac{C}{R} = F(s) = \cfrac{1}{H_1 + \cfrac{1}{\frac{H_2}{s} + \cfrac{1}{H_3 + \cfrac{1}{\frac{H_4}{s} + \cfrac{1}{H_5 + \ddots \cfrac{1}{\frac{H_{2n}}{s}}}}}}} \tag{11}$$

The block diagram for equation (11) is shown in Figure 9.

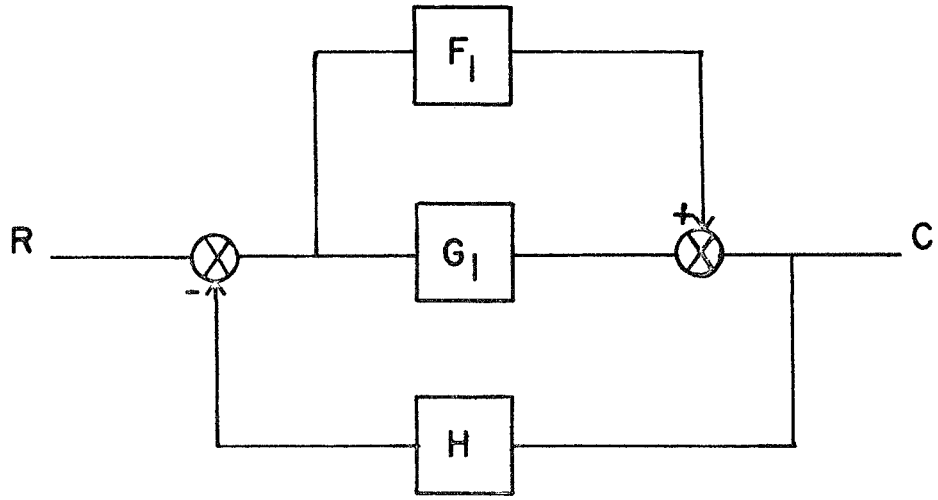


FIGURE 7

Block Diagram of Equation 9

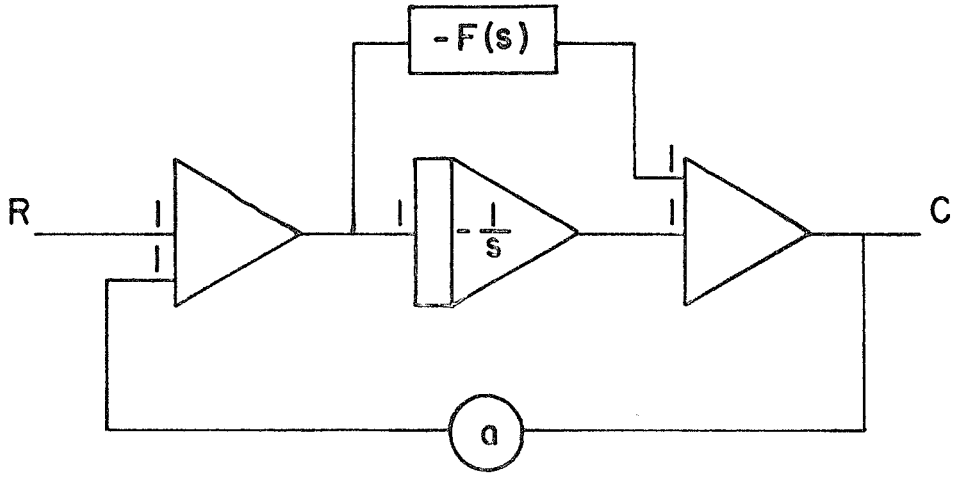


FIGURE 8

Analog Program for Equation 9

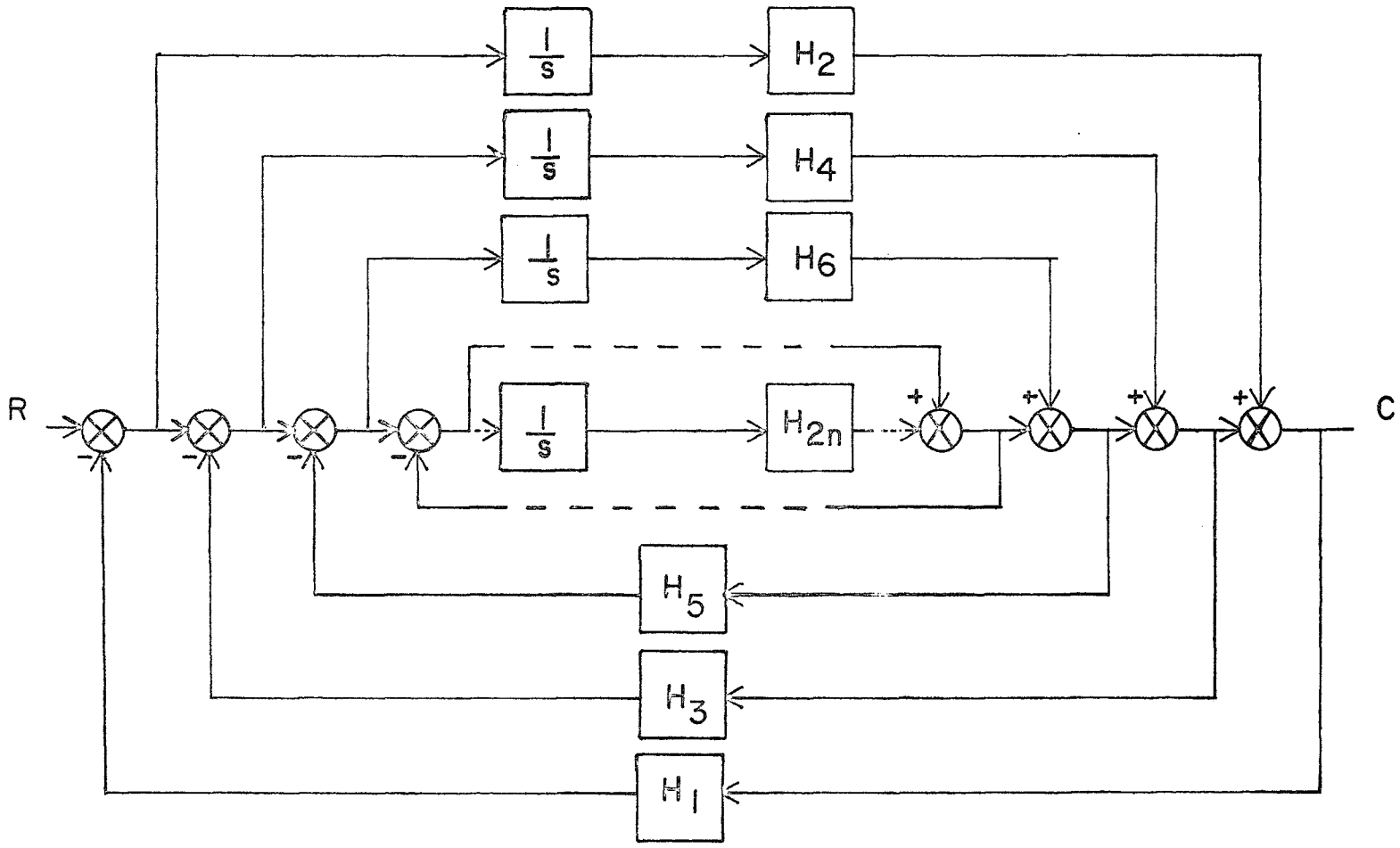


FIGURE 9

Block Diagram of Equation 11

The analog program through the H6/s term is shown in Figure 10. H terms greater than 1 require amplifiers in cascade with the integrators. Negative H terms require an odd number of cascaded amplifiers not counting the integrators. If an H term requires amplification but does not have a negative sign, then an even number of amplifiers in cascade are required. A completely general analog program is presented in appendix A.

Once the system is programmed on the analog computer it is a simple task to simplify the model as presented by Chen and Shieh [6], who state that "the most dominant term is H1 and the second influence term is H2/s. As the quotients in the continued fraction descend lower .... they are less .... important .... on the performance of the system." With this in mind, all one has to do is remove the inner most loops of the system (while observing the transient response) until the system response is degraded past a desired accuracy; and then replace the last loop removed.

Using the analog computer for this task affords distinct advantages over digital computer compilation or other methods of system response calculations. One advantage is an intimate feel for the system under examination. No one can deny the advantage of actually seeing the response, of their system model, traced out on some display device, and then being able to simplify the system by unpatching a loop, and then watching the simplified model response.

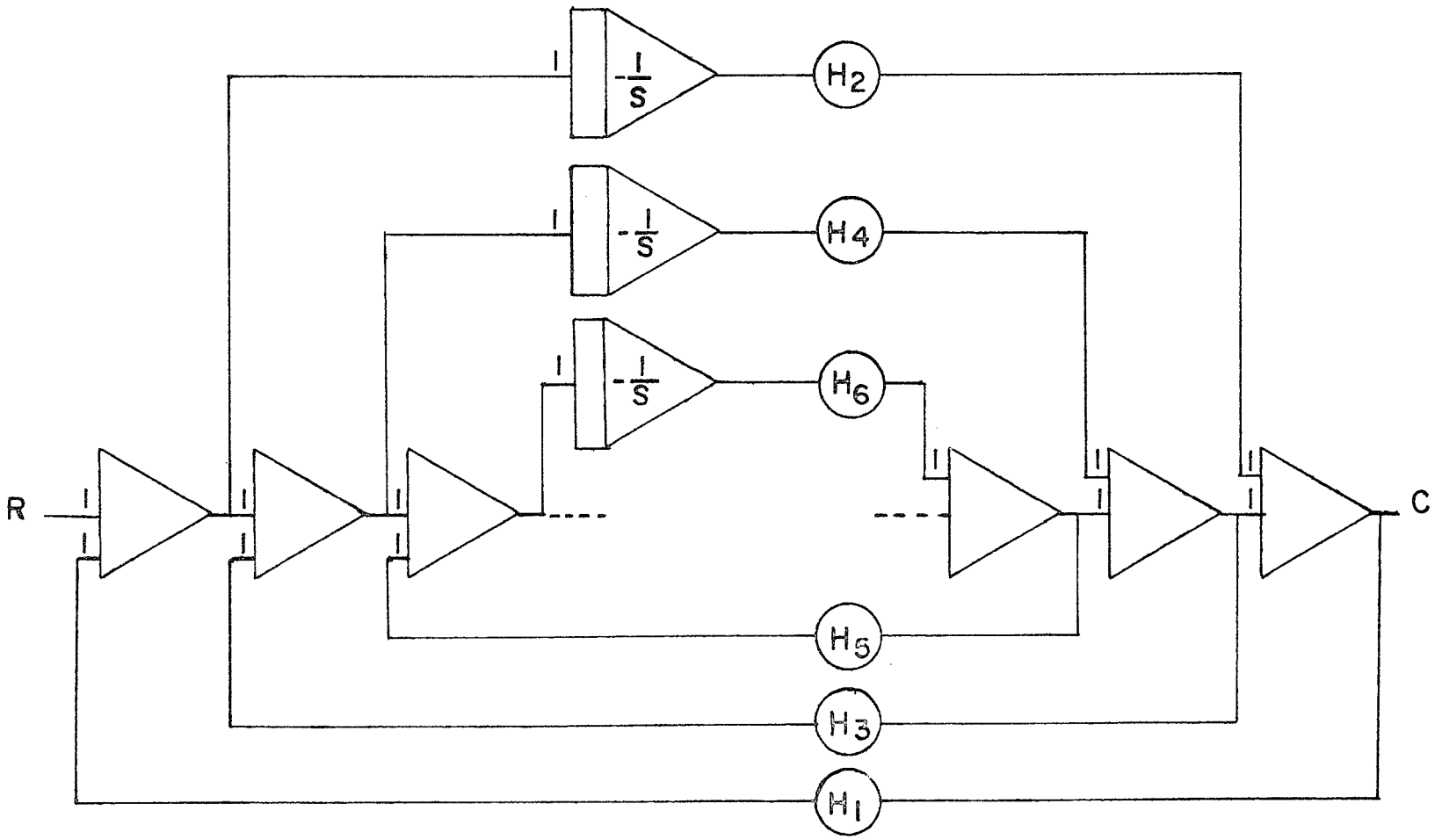


FIGURE 10

Analog Program for Equation 11

CHAPTER IV

PRESENTATION OF SINGLE INPUT SINGLE OUTPUT EXAMPLES

Using the techniques outlined in the foregoing chapter, two example systems were analyzed. The first is a practical example [7].

$$T_1\ (s) = \frac{1441.53s^3 + 78319.s^2 + 525286.125s + 607693.25}{s^7+112.04s^6+3755.92s^5+39736.73s^4+363650.56s^3+759894.19s^2+683656.25s+617497.375} \tag{12}$$

by arranging the polynomial in ascending order and then expanding into the continual fraction, $T_1(s)$ can be expressed as

$$T_1\ (s) = \cfrac{1}{1.016+\cfrac{1}{\frac{4.054}{s}+\cfrac{1}{-.0666+\cfrac{1}{\frac{-3.803}{s}+\cfrac{1}{-10.27+\cfrac{1}{\frac{-.2572}{s}+\cfrac{1}{-42.77+\cfrac{1}{\frac{-.2591}{s}+\cfrac{1}{29.01+\cfrac{1}{\frac{.2695}{s}+\cfrac{1}{-15320.+\cfrac{1}{\frac{-.0052}{s}+\cfrac{1}{47760.+\cfrac{1}{\frac{.0007}{s}}}}}}}}}}}}}} \tag{13}$$

The analog program for equation (13) is shown in Figure 11.

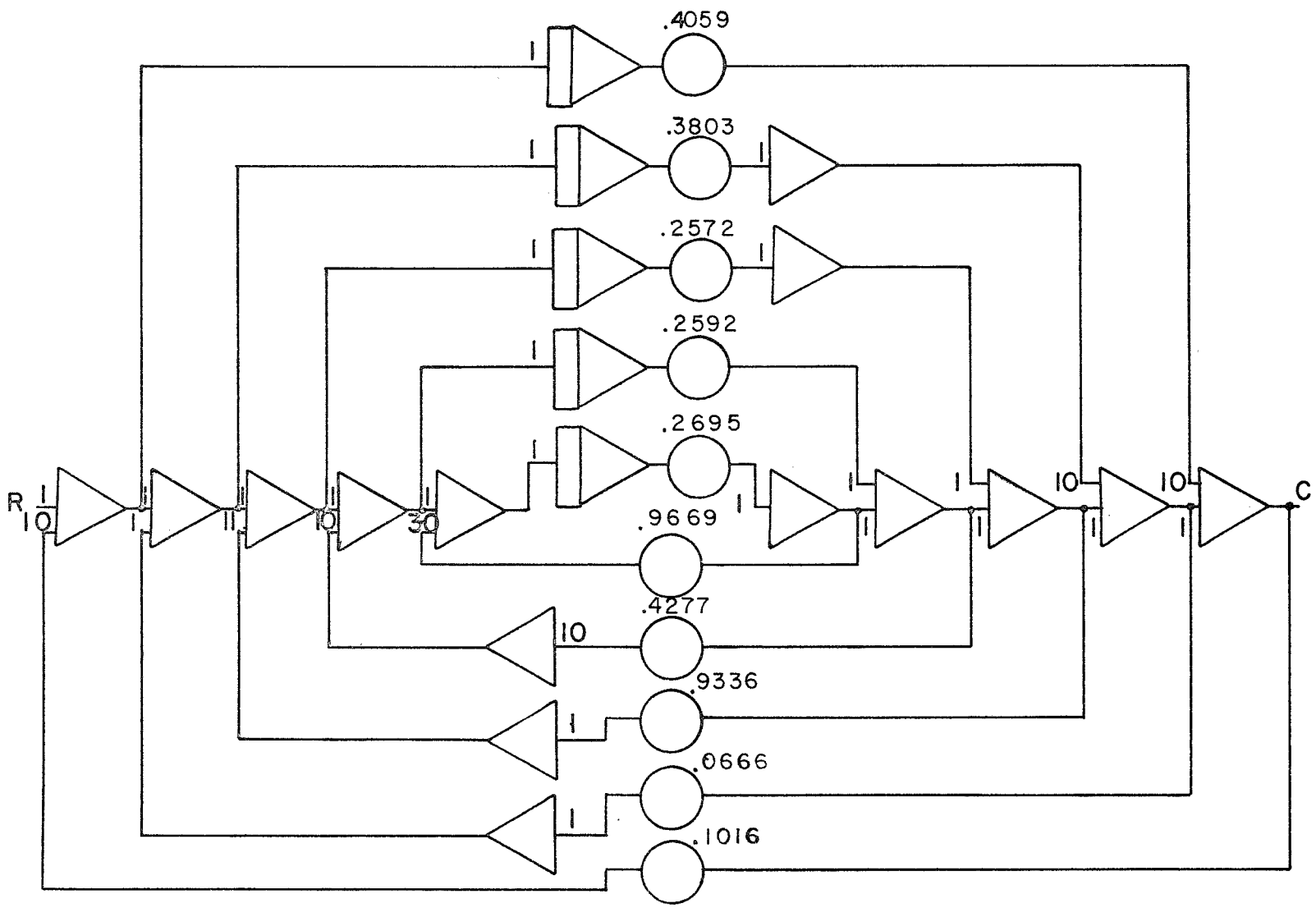


FIGURE II

Analog Program for First Example

The analog program was implemented on a Hybrid System SS-100 (100 volt analog computer). The input was a step function of 70 volts. This function is easily implemented with the program as shown in Figure 11; the input (R) is patched to a potentiometer set at .7000 and the potentiometer is patched to Reference (+100V). Thus, whenever the integrators are switched from IC to OP, the system will see a step function of 70 volts because the integrators are in series with every feed forward section of the program. The system was implemented only through the $\frac{H_{10}}{S}$ term because the H_{11} and H_{13} terms require gains that are impractical to implement on an analog computer. However, as shown in Figures 12 and 13, the omission of these terms did not impair the system modeling. As seen in Figure 12, one cannot distinguish between the third, fourth, and fifth order system approximations. Only in Figure 13 (an expanded view of the origin) can one see a difference between the third, fourth and fifth order approximations.

The region around the origin of a transient response curve is dominated by the higher frequency characteristics of the system. Therefore, in leaving off the H_{11} through $H_{14/s}$ terms, one has only altered the higher frequency characteristics of the system. Therein lies the beauty of the second Cauer form; one has at his disposal the tool which only reduces the upper frequency characteristics of a system and then only to the degree one desires. The power of this tool is seen when the results obtained are compared to the results

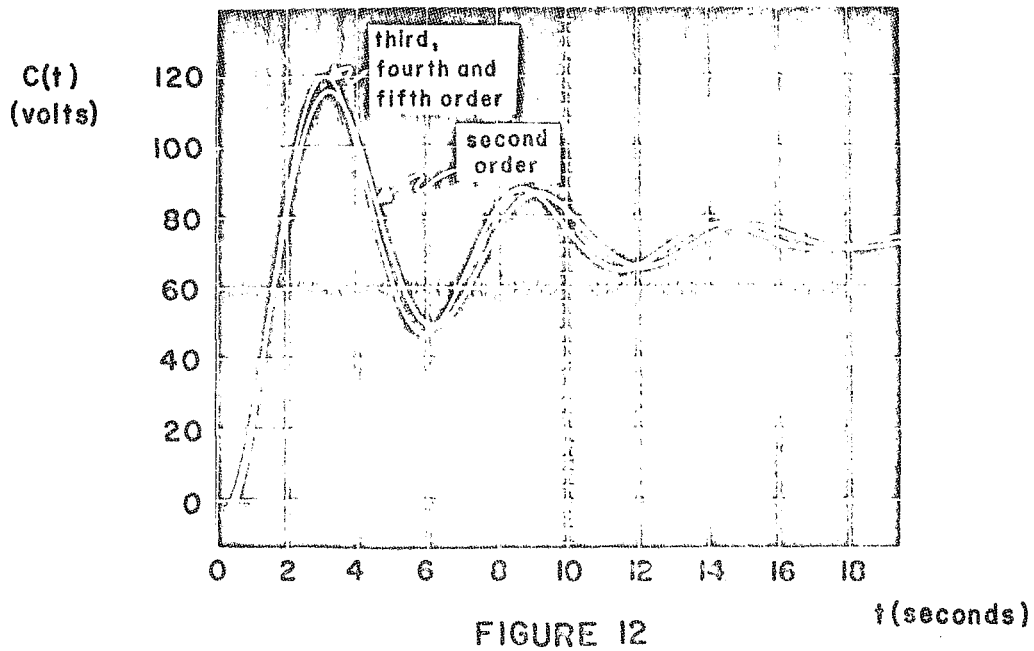


FIGURE 12

Response Curves for first Example

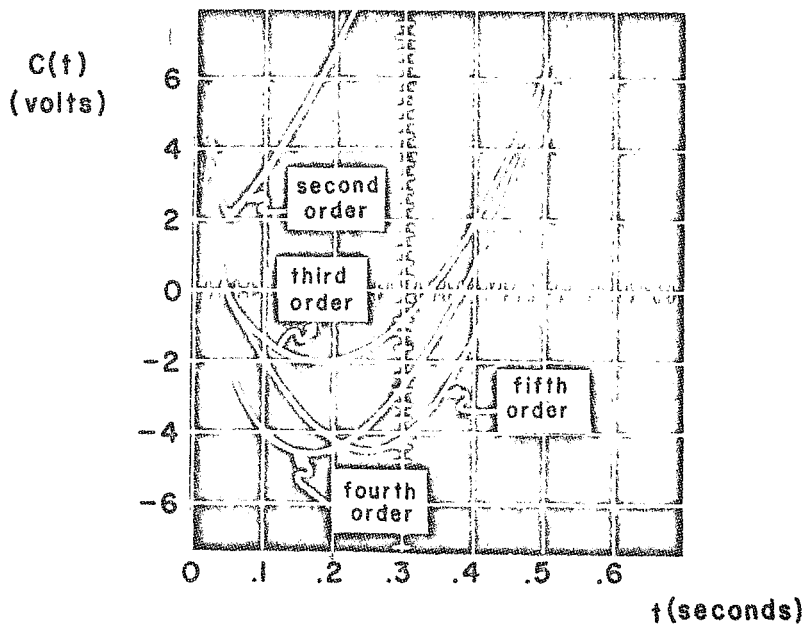


FIGURE 13

Response Curves for first Example

obtained using other techniques such as the pole-zero reduction technique. In the pole-zero reduction technique one can not limit the effect of simplification to just the upper frequencies; on the contrary, as indirectly shown by Figure 4, the effect is spread over quite a large range of frequencies.

The second example demonstrates the simplification technique of a system which is not easily reduced by conventional methods. [6]

The transfer function

$$T_2(s) = \frac{s^2 + 15s + 50}{s^4 + 5s^3 + 33s^2 + 79s + 50} \tag{14}$$

can be factored into

$$T_2(s) = \frac{(s + 5)\ (s + 10)}{(s + 1)\ (s + 2)\ (s + 1 + j4.9)\ (s + 1 - j4.9)} \tag{15}$$

The problem with using existing pole-zero reduction techniques on this system is readily seen; three poles have the same real part.

Expanding $T_2(s)$ into continued fraction expansion one has (16)

$$T_2(s) = \cfrac{1}{1+\cfrac{1}{\frac{.7812}{s}+\cfrac{1}{-6.400+\cfrac{1}{\frac{-.7463}{s}+\cfrac{1}{-4.611+\cfrac{1}{\frac{18.02}{s}+\cfrac{1}{.0089+\cfrac{1}{\frac{-18.05}{s}}}}}}}}}$$

The analog program for equation 16 is presented in Figure 14.

The second example response curves were obtained in the same manner as used in the previous example except that the input was changed to a step function of 60 volts.

As shown in Figures 15 and 16, the reduction on the system transfer function of the second example was a success. In Figure 16 one can not even see a difference in the two curves because of the width of the oscilloscope trace.

The simplification of this system could only be performed from a fourth order to a second order system because the original system has a pair of complex conjugate poles and the simplification to a third order yields a condition where the system has a pole in the right half of the s-plane.

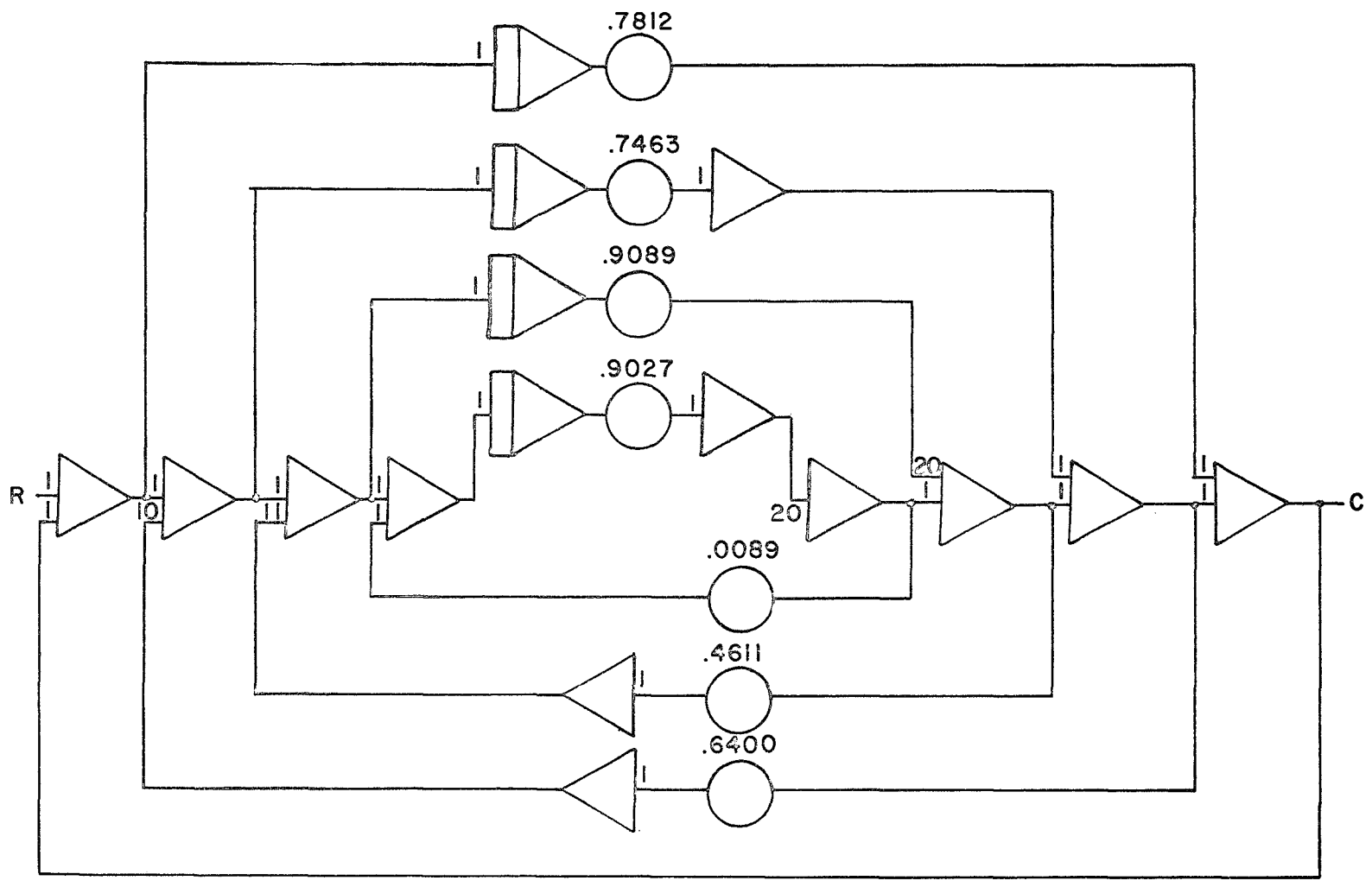


FIGURE 14

Analog Program for Second Example

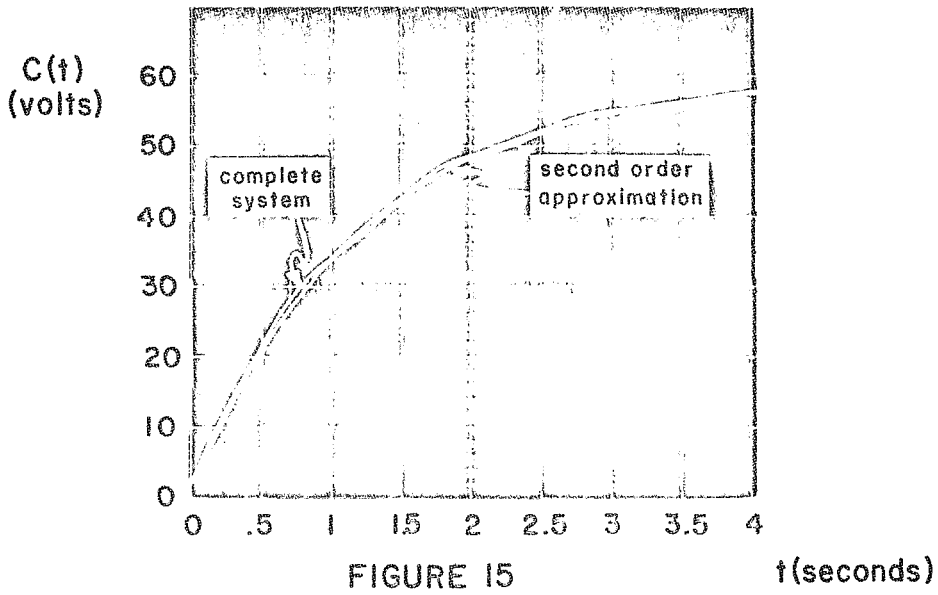


FIGURE 15

Response Curves of the Second Example

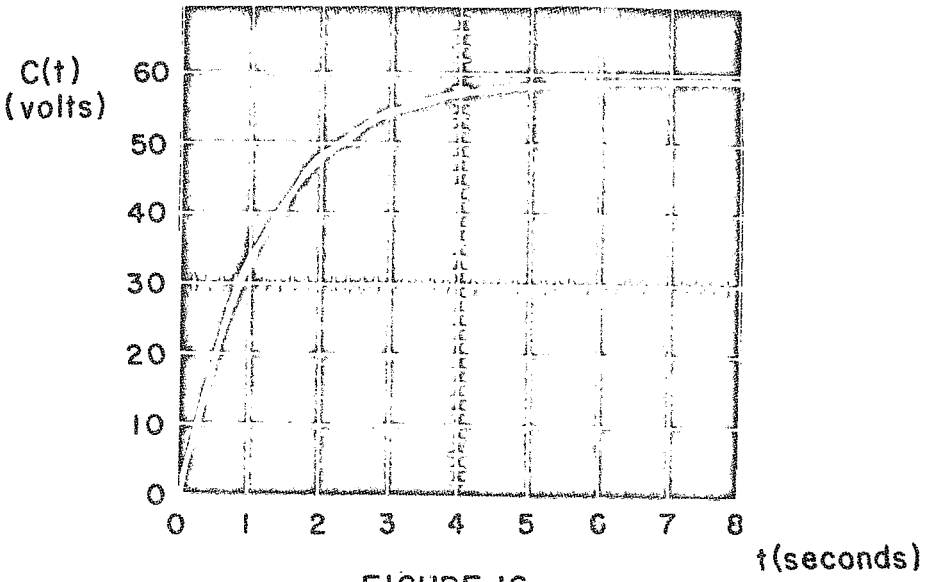


FIGURE 16

Response Curves of the Second Example

CHAPTER V

THE CAUER MODEL FOR MULTIVARIABLE SYSTEMS

The techniques developed for the single input, single output system in the previous section can be directly extended to the multivariable case. All one has to do is consider that the input and the output are two vectors.

I. DEVELOPMENT OF THE CAUER MODEL

One writes the transfer matrix of the overall system as follows:

$$[C] = [T(s)]\ [R] \tag{17}$$

where $[T(s)]$ is a square matix and each element of the transfer matrix is a transfer function. Without loss of generality, one can use a two by two matrix as an example.

$$[T(S)] = \begin{bmatrix} \dfrac{B_{11}\ (S)}{\Delta(S)} & \dfrac{B_{12}\ (S)}{\Delta(S)} \\ \\ \dfrac{B_{21}\ (S)}{\Delta(S)} & \dfrac{B_{22}\ (S)}{\Delta(S)} \end{bmatrix} \tag{18}$$

where $\Delta(S) = A_n + A_{n-1}S + \ . \ . \ . + S^n$ (19)

and $B_{ij}(S) = b_{ijn} + b_{ij(n-1)}S + \ . \ . \ . \ . + b_{ij1}S^{n-1}$ (20)

Taking the inverse of equation (18) and then separating the constant

from the remaining parts, one has

$$[T(S)] = [H_1 + S[. . . . . .]^{-1}]^{-1} \tag{21}$$

where $[H_1]$ is a constant matrix and the remaining parts have a common factor S. Then, after factoring out the S, one repeats the process. Finally, one has

$$[T(S)] = [H_1 + S[H_2 + S[H_3 + \ldots S[H_{n-1} + S[H_n]^{-1}]^{-1} \ldots]^{-1}]^{-1}]^{-1} \tag{22}$$

Equation (22) is the general matrix Cauer form.

The crudest approximation can be obtained by keeping only two terms $[H_1]$ and $[H_2]$ and discarding all other matrices.

$$[T(S)]^{1} = [H_1 + S[H_2]^{-1}]^{-1} \tag{23}$$

There is only one "S" involved in the matrix; however, even this simple first order approximation can express the exact steady state of the system shown in (17).

A better model can be obtained by keeping two more terms; one has

$$[T(S)]^{11} = [H_1 + S[H_2 + S[H_2 + S[H_4]^{-1}]^{-1}]^{-1}]^{-1} \tag{24}$$

The model shown in equation (24) is quite good for approximating most systems. Of course, if we take more and more terms and we get better and better approximations; however, in practice equation (24) is usually sufficient. This conclusion is a natural extension of the single input, single output case and will be demonstrated by the use of an example in the next chapter.

The block diagram representation of equation (24) can be expressed as shown in Figure 17.

The general matrix Cauer formula [equation (22)] is also easily expressed by a block diagram. See Figure 18. If one disconnects all the inner loops in figure 18, except $[H_1]$ and $[H_2]$, one has the simplified model corresponding to (23). If one also keeps $[H_3]$ and $[H_4]$ and disconnects their inner loops, then one has a better simplified model, corresponding to equation 24.

II. DEVELOPMENT OF A CAUER MODEL ANALOG PROGRAM FOR MULTIVARIABLE SYSTEMS

The next step is to build a basic analog computer program block for the two input two output system; this block is presented in Figure 19.

The derivation of this program was quite simple. If one examines only the circuit containing the first three amplifiers (A_1, A_2 and A_3) and the first integrator (I_1), then it is clear that the program is the same as the program used for the single input/output system. The only difference is the cross-coupling terms (H_{n12} and H_{m12}) which were added for the two input - two output system.

The interconnection of basic block diagrams for two input - two output system is presented in Figure 20.

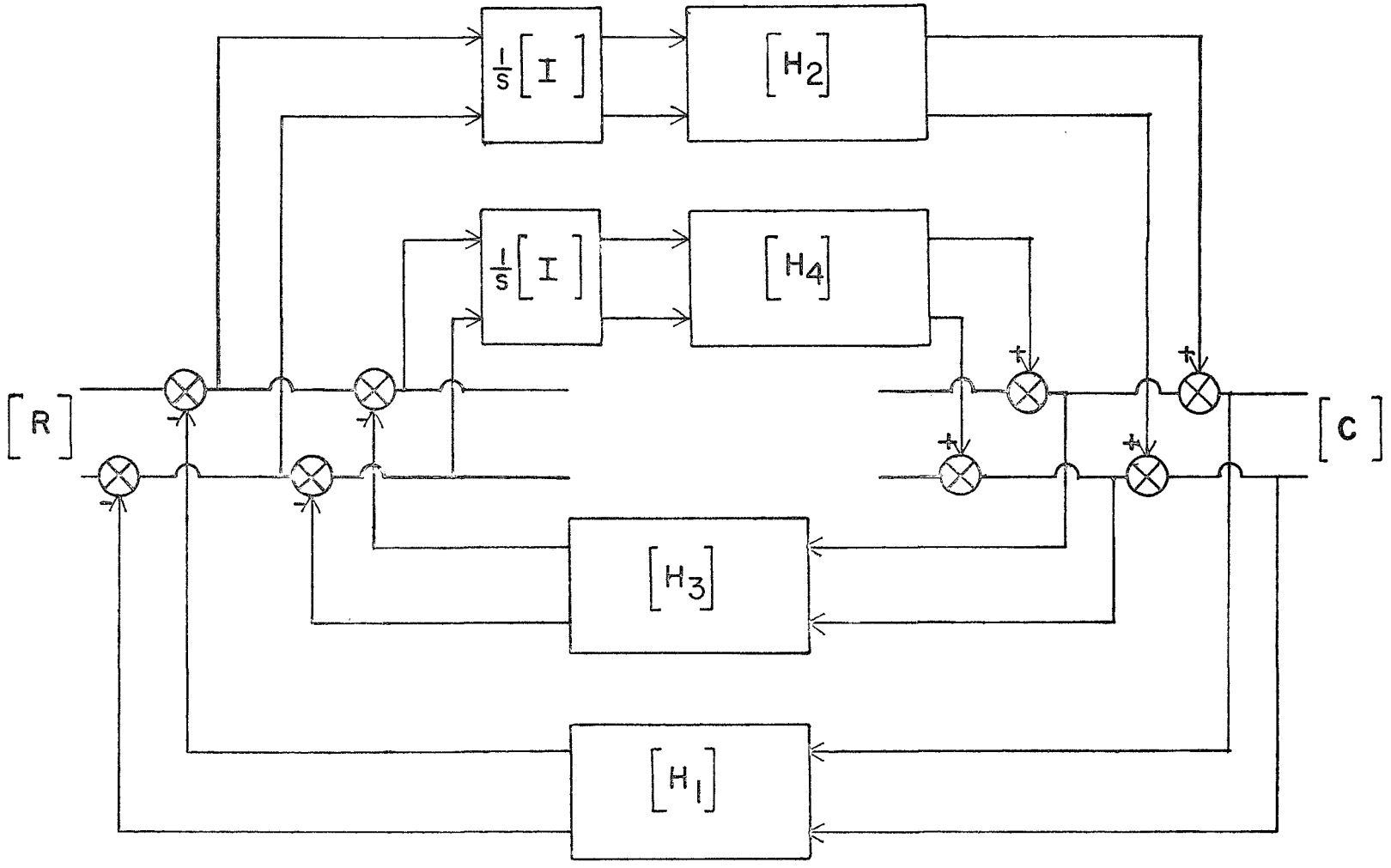


FIGURE 17

Block Diagram of Equation 24

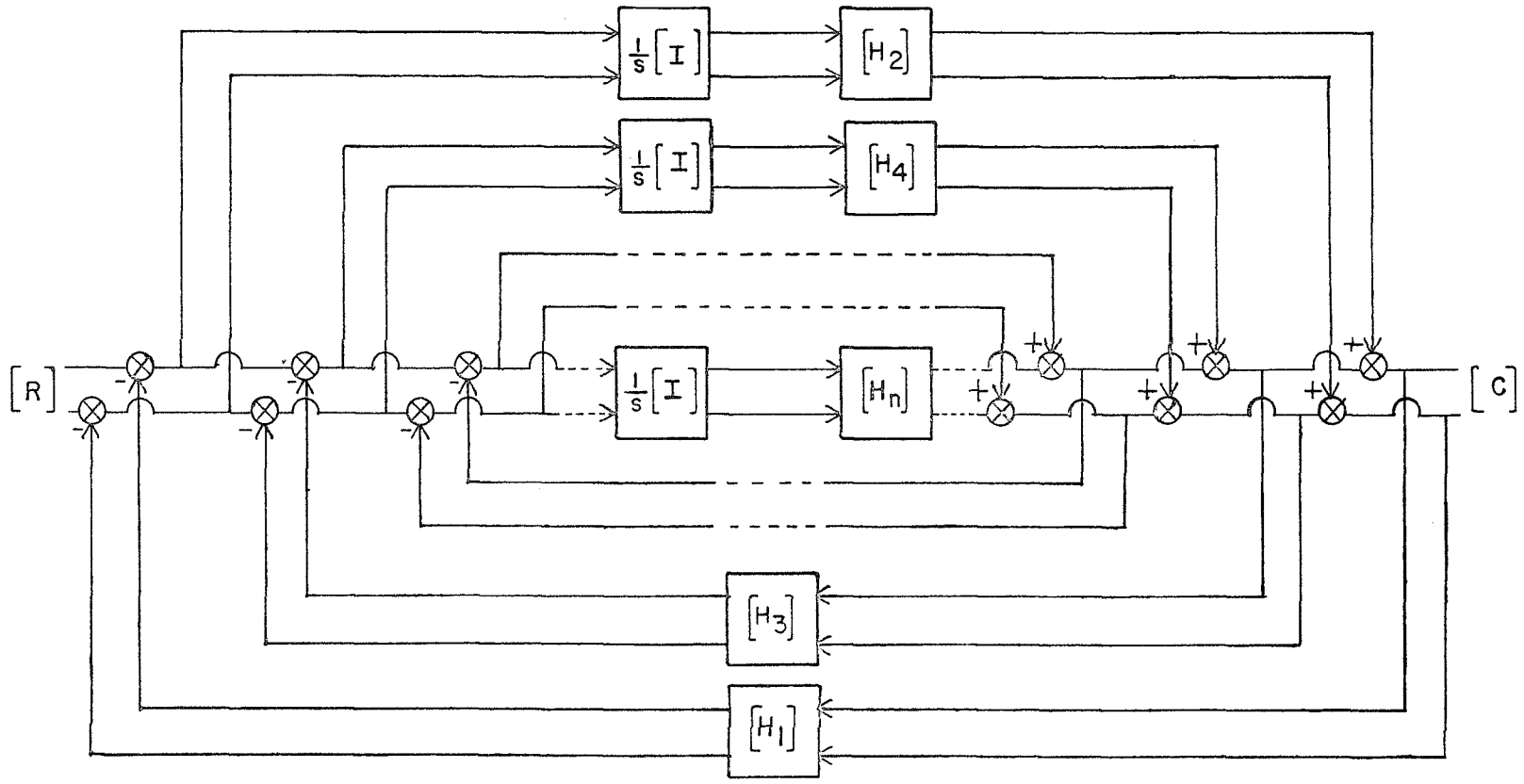


FIGURE 18

Block Diagram of Equation 22

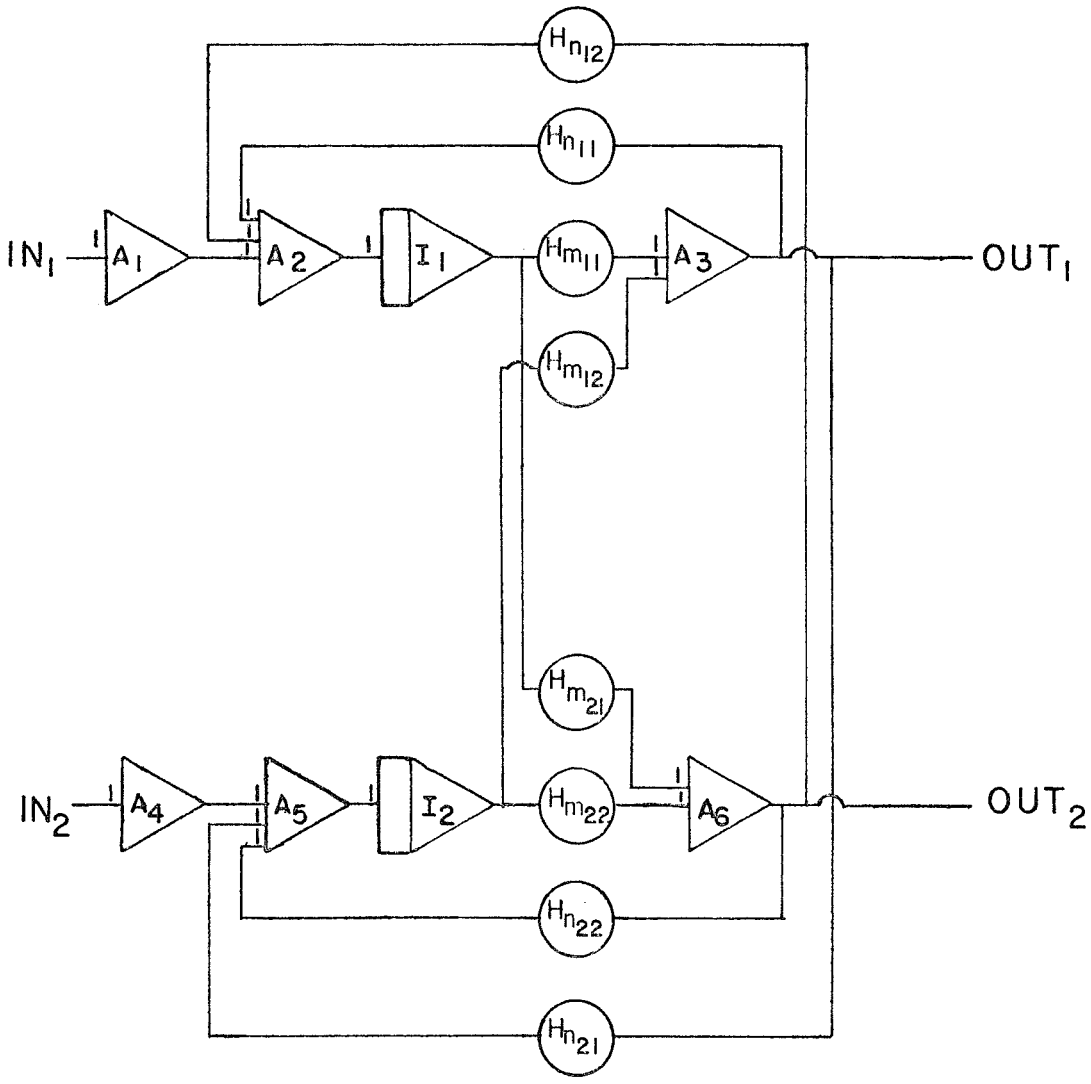


FIGURE 19

Basic Program Block for Two Input/Output System

note: n is any odd number and m is the next even number

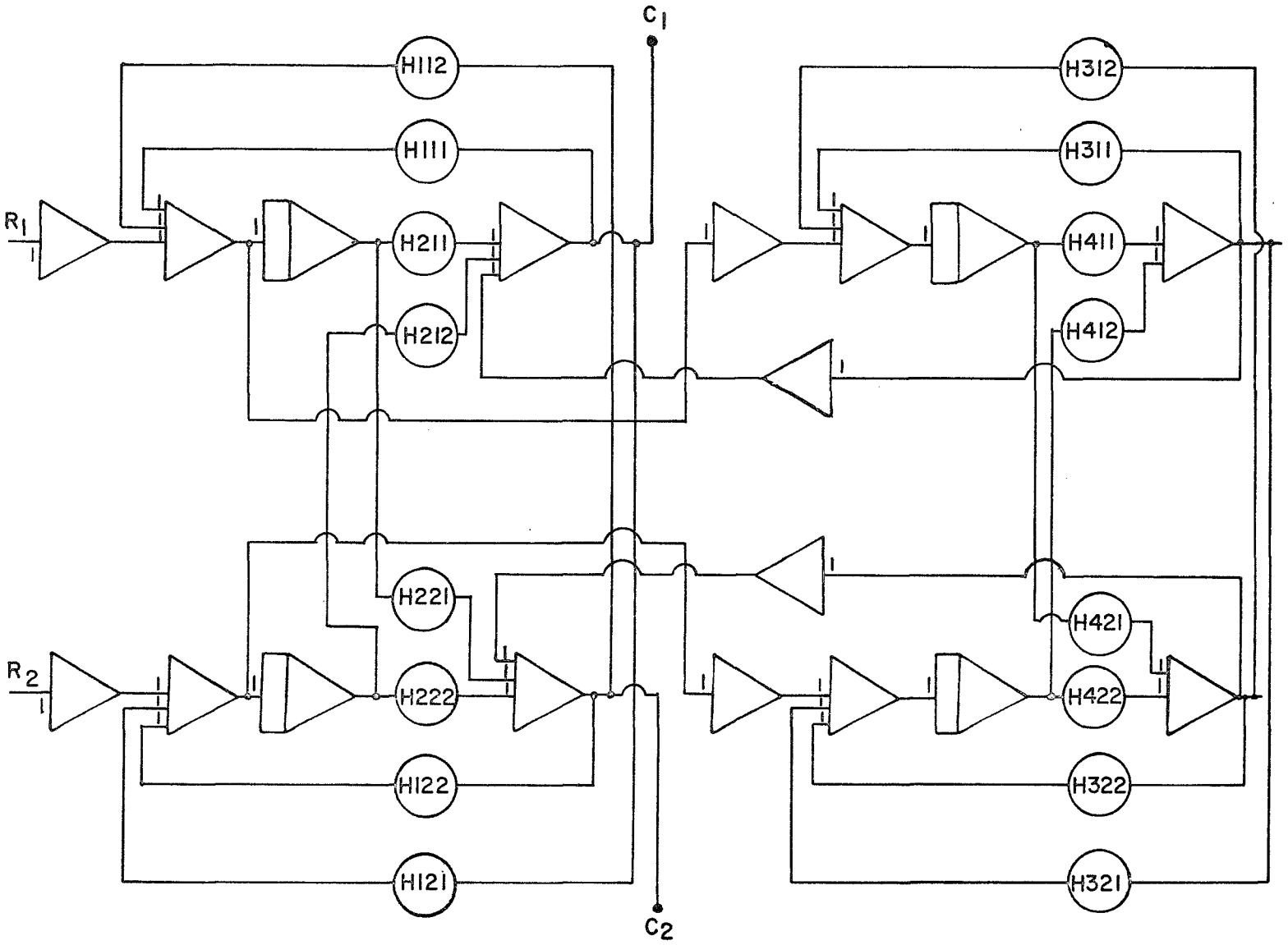


FIGURE 20

Interconnection Diagram of Two Input Two Output System with Terms Through H_{4ij}

CHAPTER VI

PRESENTATION OF TWO INPUT-TWO OUTPUT EXAMPLE

Example 3 is a two input-two output system which has the transfer function

$$[T(s)] = \frac{1}{\Delta(s)} \begin{bmatrix} T_{11}(s) & T_{12}(s) \\ T_{21}(s) & T_{22}(s) \end{bmatrix} \tag{25}$$

where $T_{11}(s) =$ (25A)

$$-5.3350(s^7+469.26s^6+32141.s^5+375100.s^4+764560.s^3+3586600.s^2+8801500.s+4763000.$$

$T_{21}(s) =$ (25B)

$$7.7620(s^7+332.01s^6+22322.s^5+252910s^4+377990.s^3+1805,000s^2+5,088,600s+2,864,400.$$

$T_{12}(s) =$ (25C)

$$5.6698(s^7+521.14s^6+37192.s^5+540,450.s^4+813230.s^3+183200.s^2+5,651,000.s+3,361,200.$$

$T_{22}(s) =$ (25D)

$$-3.9271(s^7+353.00s^6+45952.s^5+791,180.s^4+979,940.s^3+1,399,000.s^2+6,259,800.s+4,044,000.$$

and $\Delta(s) = s^8+581.58s^7+43543.s^6+667,870.s^5+3,562,300.s^4+12,832,000.s^3+24,944,000.s^2+21,288,000.s+6,362,700.$ (25E)

The methods described in Chapter V for obtaining the Second Cauer matrix form are now applied to the system defined by equation (25). The result is the second Cauer matrix equation (26)

$$[T(S)] = \left[\begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} + S \left[\begin{pmatrix} -2 & 1 \\ 1.5 & -.5 \end{pmatrix} + S \left[\begin{pmatrix} 25 & 20 \\ 4 & 1 \end{pmatrix} + S \left[\begin{pmatrix} -2 & 2 \\ 2.5 & -1 \end{pmatrix} + S \left[\begin{pmatrix} 1 & 10 \\ 3 & 7 \end{pmatrix} + S \left[\begin{pmatrix} -.2 & .4 \\ .3 & -.1 \end{pmatrix} + S \left[\begin{pmatrix} 1 & 100 \\ 20 & 10 \end{pmatrix} + S \left[\begin{pmatrix} -.1 & .2 \\ .3 & -.2 \end{pmatrix}\right]^{-1}\right]^{-1}\right]^{-1}\right]^{-1}\right]^{-1}\right]^{-1}\right]^{-1}\right]^{-1}$$

Once equation (26) is found, it is a simple matter to plug the coefficients into the general analog program blocks shown in figure 19 and then interconnect the blocks as shown in figure 20. The analog program for the third example is shown in figure 21.

Page 1 of figure 21 contains the H_1 and H_2 terms which are

$$[H_1] = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \tag{27}$$

$$[H_2] = \begin{bmatrix} -2 & 1 \\ 1.5 & -.5 \end{bmatrix} \tag{28}$$

Page 2 of figure 21 contains the H_3 and H_4 terms which are

$$[H_3] = \begin{bmatrix} 25 & 20 \\ 4 & 1 \end{bmatrix} \tag{29}$$

$$[H_4] = \begin{bmatrix} -2 & 2 \\ 2.5 & -1 \end{bmatrix} \tag{30}$$

Page 3 of figure 21 contains the H_5 and H_6 terms which are

$$[H_5] = \begin{bmatrix} 1 & 10 \\ 3 & 7 \end{bmatrix} \tag{31}$$

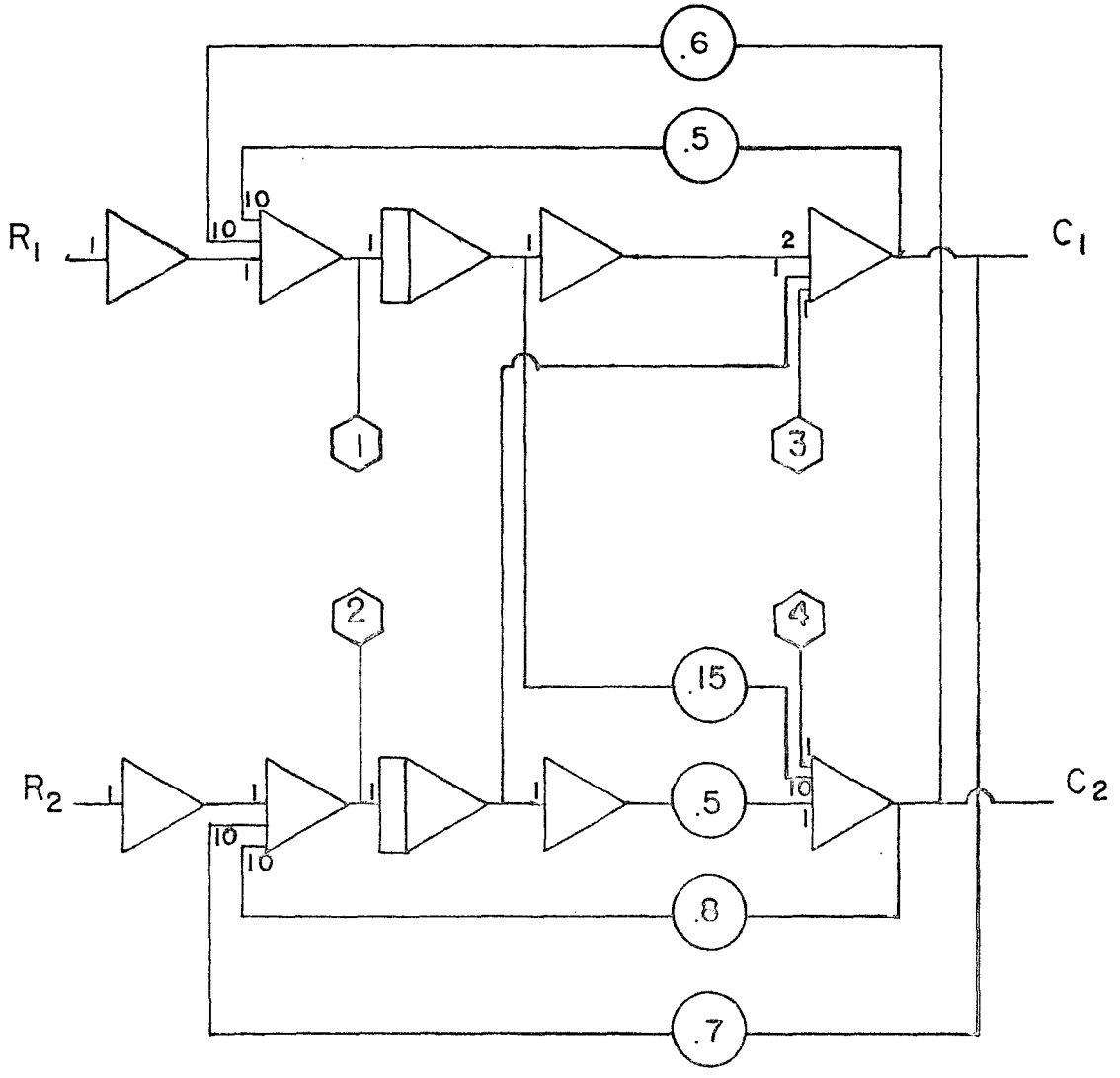


FIGURE 21

Analog Program for Third Example (equation 26)

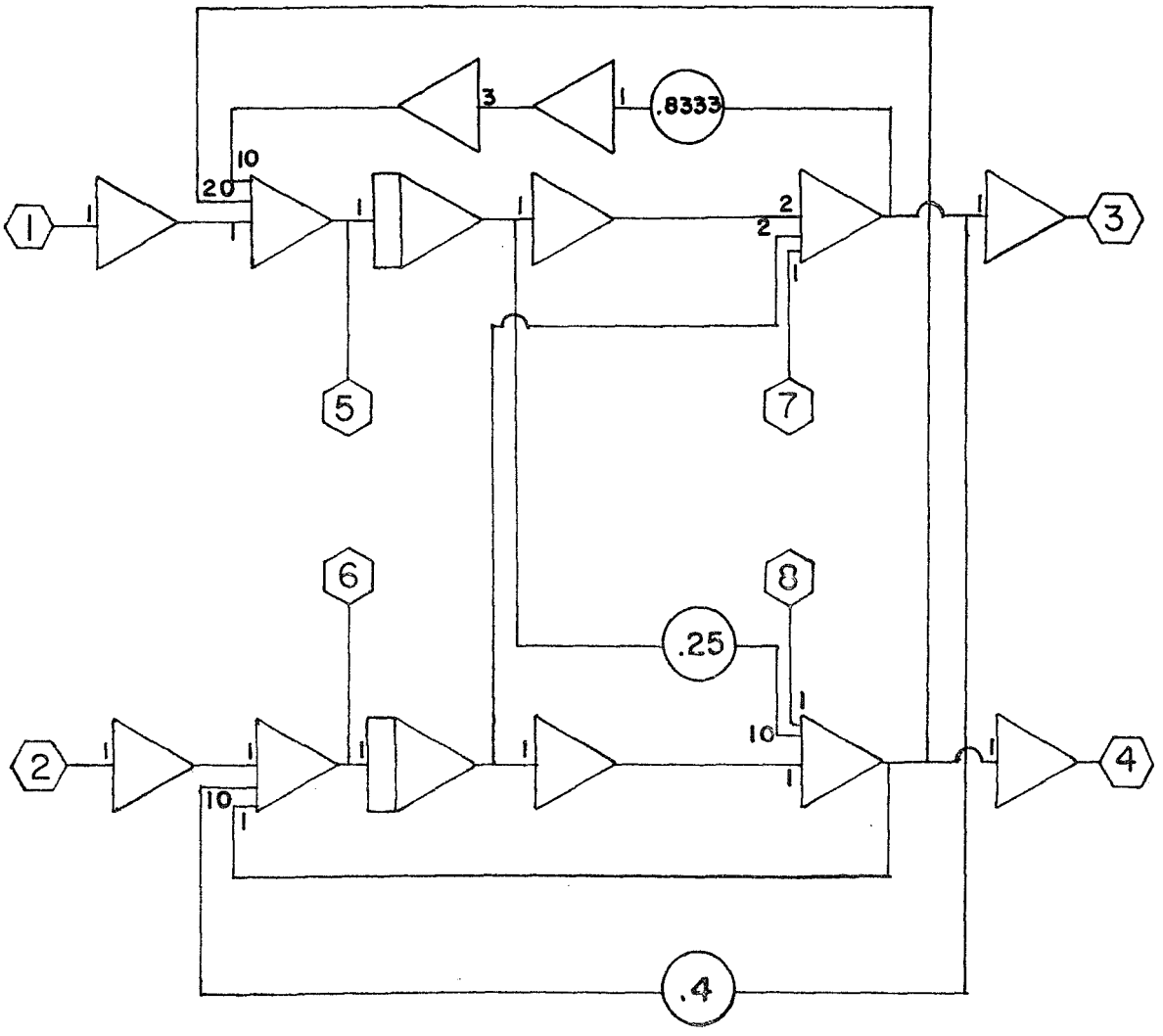


FIGURE 21 (continued)

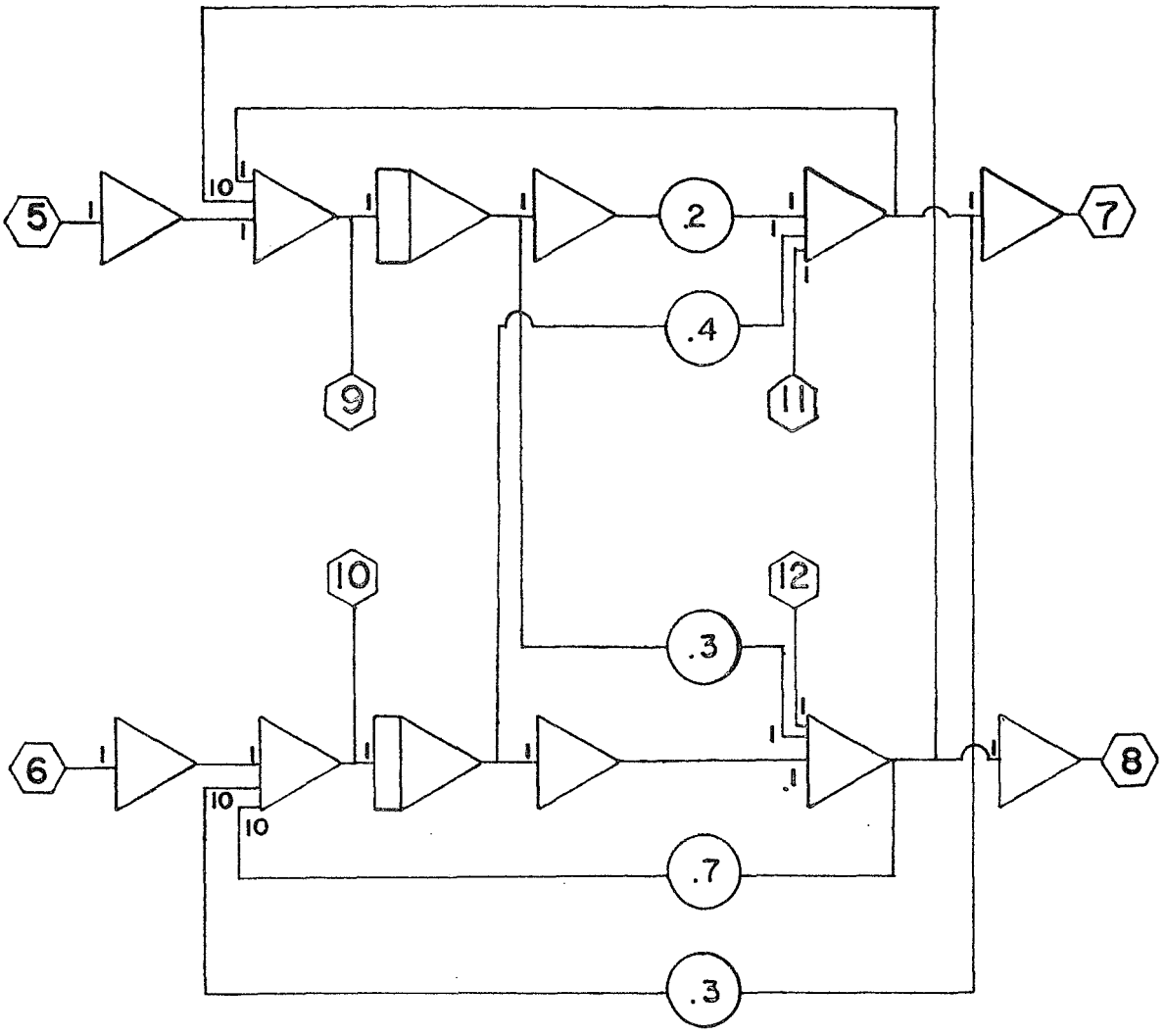


FIGURE 21 (continued)

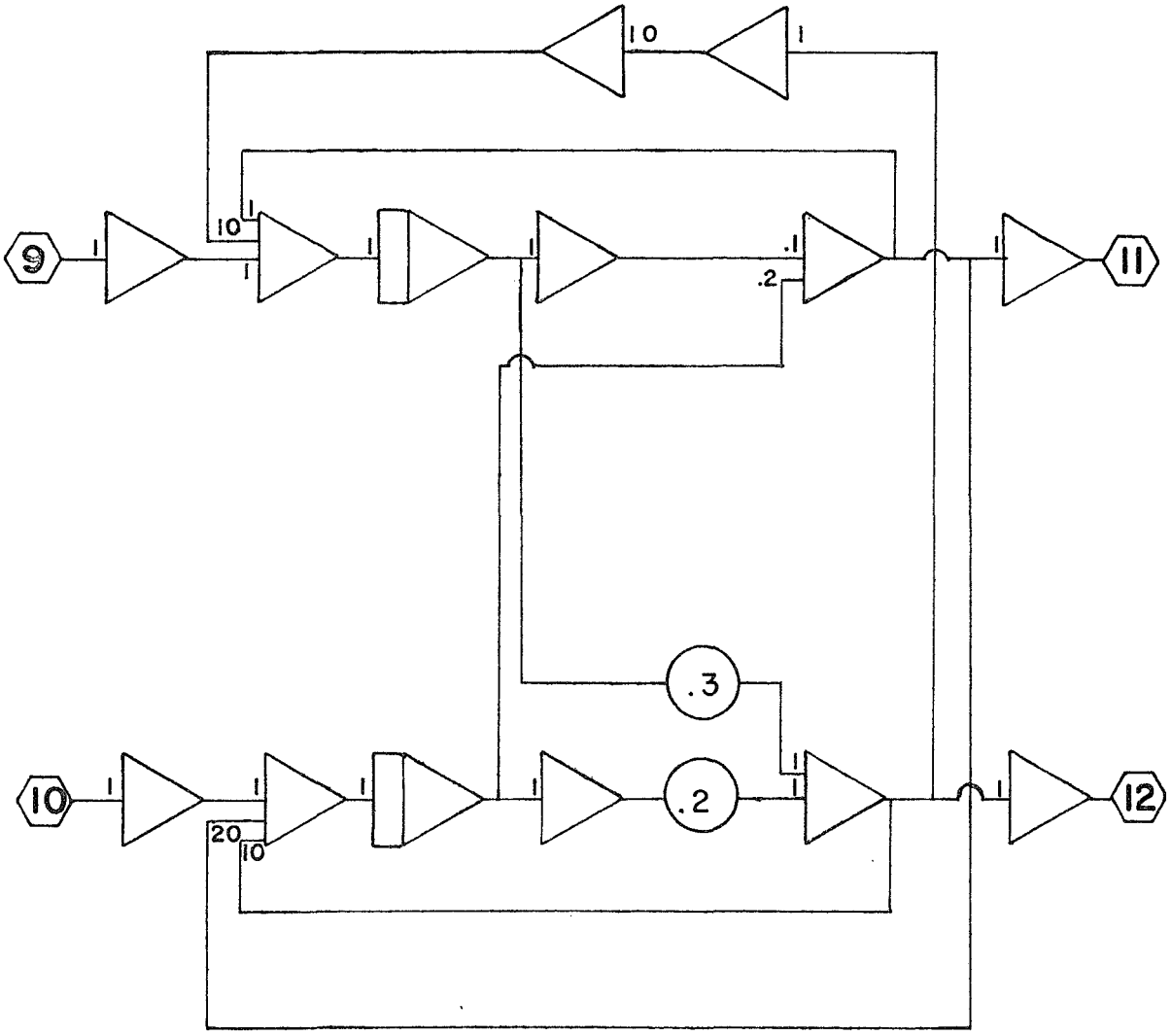


FIGURE 21 (continued)

$$[H_6] = \begin{bmatrix} -.2 & .4 \\ .3 & -.1 \end{bmatrix} \tag{32}$$

Page 4 of figure 21 contains the H_7 and H_8 terms which are

$$[H_7] = \begin{bmatrix} 1 & 100 \\ 20 & 10 \end{bmatrix} \tag{33}$$

$$[H_8] = \begin{bmatrix} -.1 & .2 \\ .3 & -.2 \end{bmatrix} \tag{34}$$

The program blocks are connected by patching the like numbered points together. Once the analog program was set up and debugged, the results as shown in figure 22 thru figure 28 were obtained. In the figures, H_8 represents the entire function, H_6 represents an approximation through the H_6 term and H_4 represents an approximation through the H_4 term. No results are presented for an approximation through the H_2 term because a first order solution for this high order system is of no interest.

In addition to the results presented, a test run of the system was compared to digital computer generated data. The analog program response's overall accuracy was within .5% with the majority of the data within .1% accuracy. In most cases an accuracy of 5% would be sufficient because one is not interested in accuracy, but relative degradation of each approximation as compared to the original system.

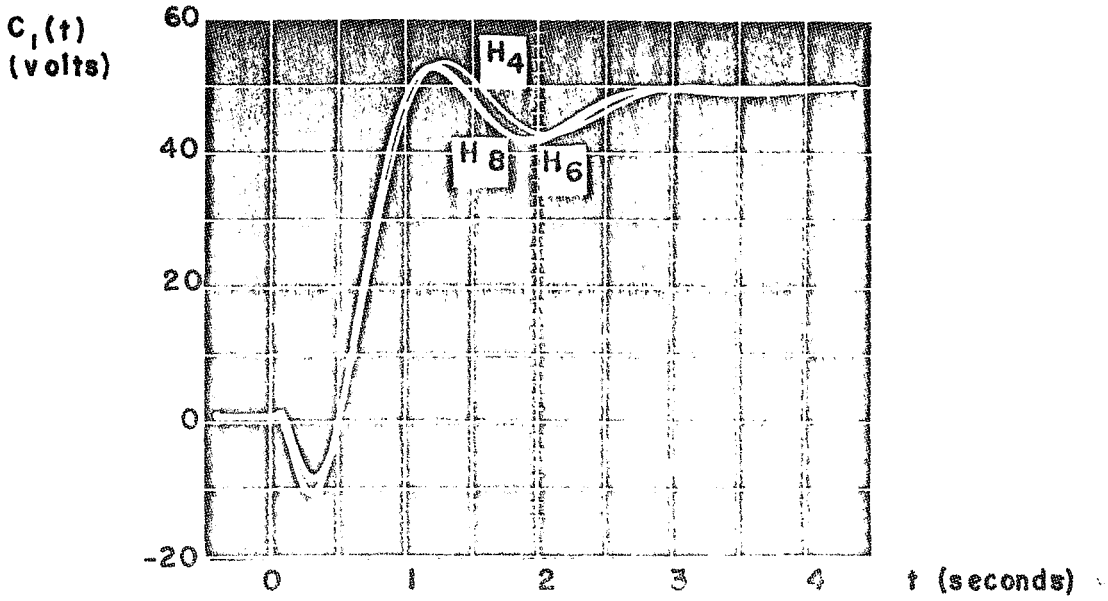


FIGURE 22

C_1 Responses for Example 3 $(R_1 = R_2 = -\frac{50}{s})$

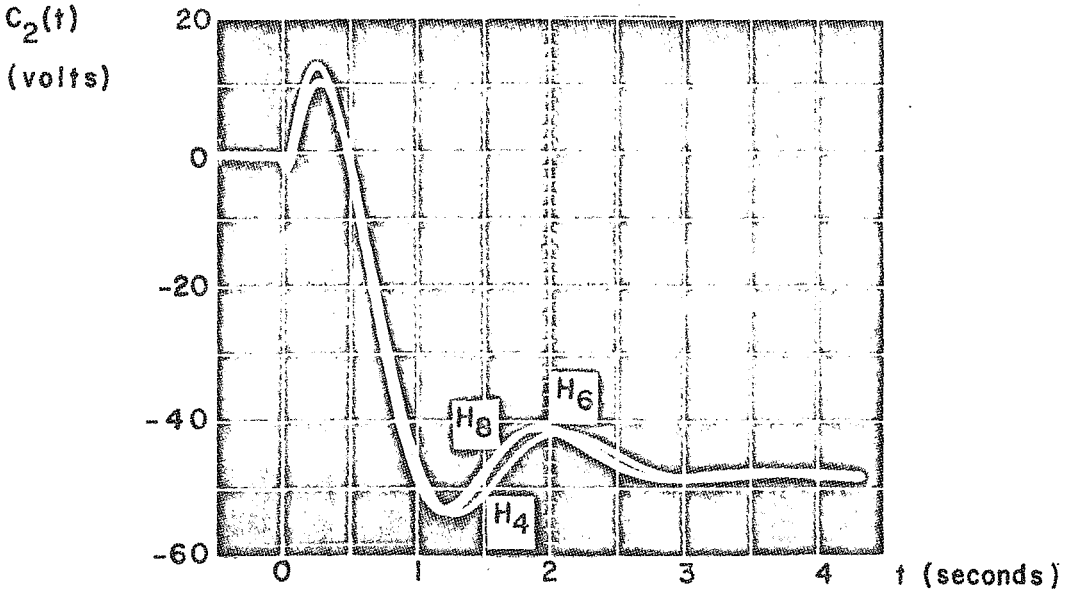


FIGURE 23

C_2 Responses for Example 3 $(R_1 = R_2 = -\frac{50}{s})$

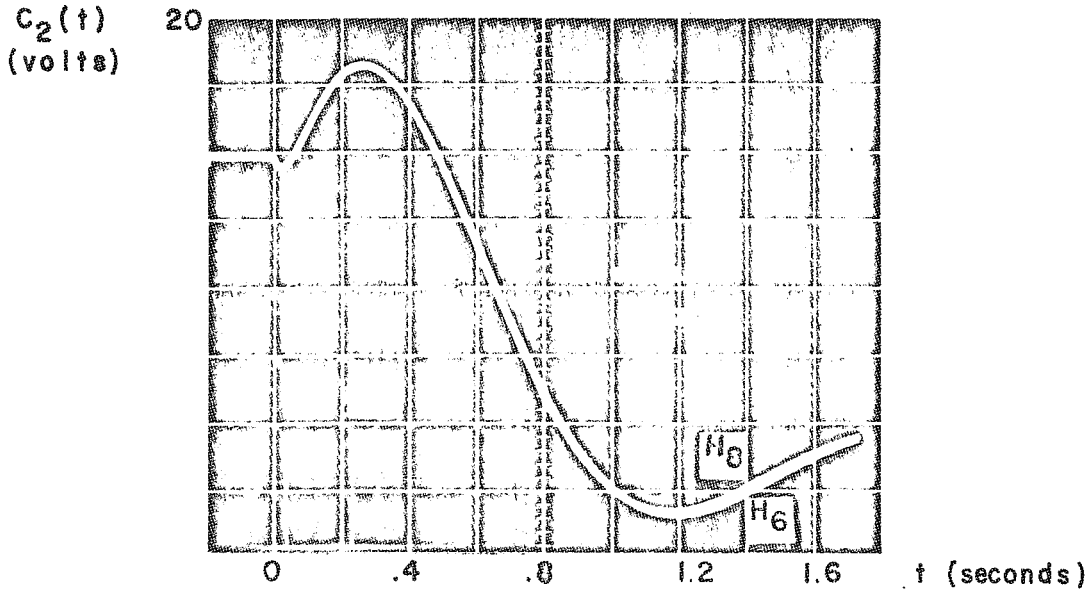


FIGURE 24

C_2 Responses for Example 3 $(R_1 = R_2 = -\frac{50}{s})$

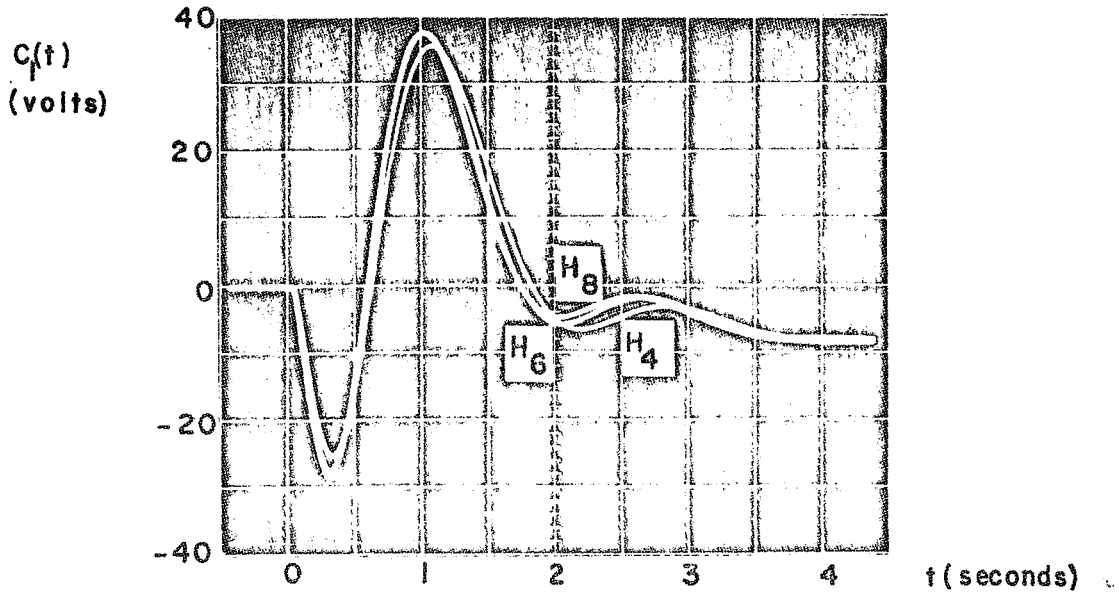


FIGURE 25

C_1 Responses for Example 3 ($R_1 = -\frac{50}{s}$, $R_2 = -\frac{70}{s}$)

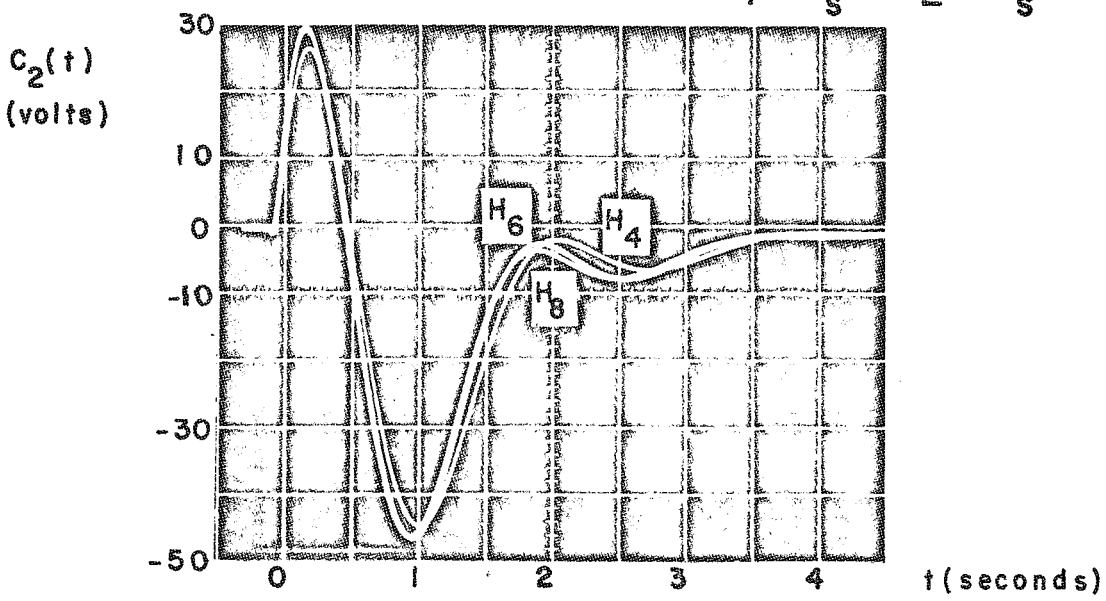


FIGURE 26

C_2 Responses for Example 3 ($R_1 = -\frac{50}{s}$, $R_2 = -\frac{70}{s}$)

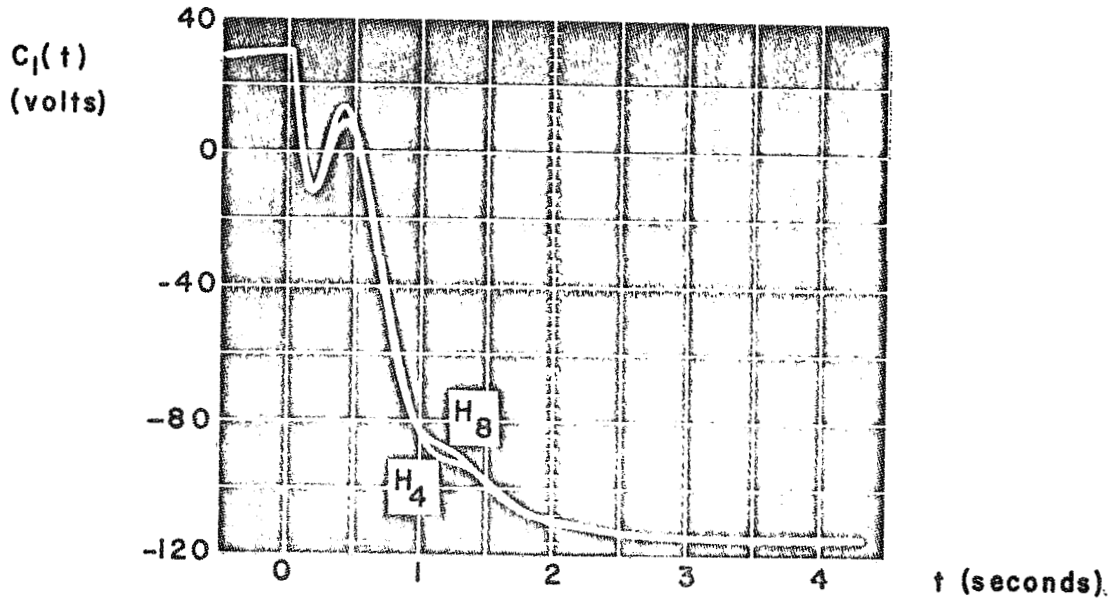


FIGURE 27

C_1 Responses for Example 3 $(R_1 = -28.4, R_2 = -\frac{50}{s})$

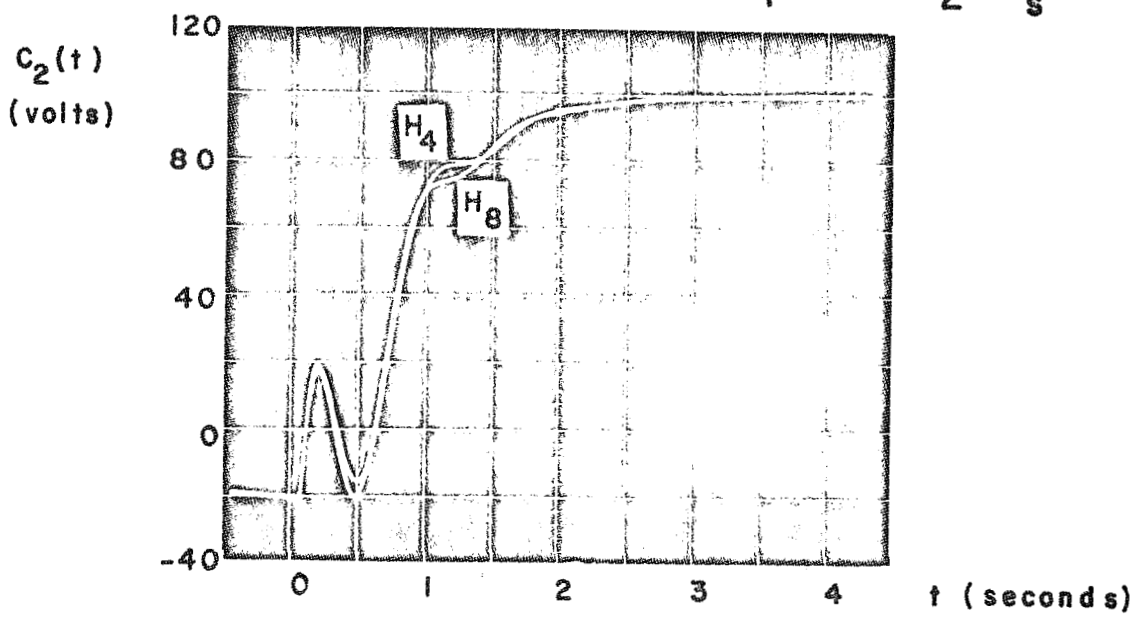


FIGURE 28

C_2 Responses for Example 3 $(R_1 = -28.4, R_2 = -\frac{50}{s})$

Figures 22 and 23 present the C_1 and C_2 output responses for -50 volt steps on both the R_1 and R_2 inputs. Figure 24 presents an expanded view of the C_2 output as shown in Figure 23 except the H_4 approximation is not shown. The results as presented by these figures are extremely good. Only in Figure 24 can one see the difference the H_6 approximation has on the response. Even the H_4 approximation is within 5% of the original system.

C_1 and C_2 responses to unequal step inputs are presented in Figures 25 and 26. The R_1 input has a step of -50 volts applied, while the R_2 input has a step of -70 volts applied. The results again are excellent; however, one can see a difference in the responses of the original system (H_8) and the H_6 approximation more clearly than in Figures 22 and 23.

The last set of responses (Figures 27 and 28) were run to determine if any cross-coupling degradation could be caused by the approximation techniques. To accomplish this, the R_1 input had a -28.4 volt step applied and the system was allowed to reach a steady state (t = 0 on the figures). At t=0 in figures 27 and 28, a step of -50 volts was applied to R_2. This procedure allows one to isolate the effects of the cross-coupling terms. The complete system (H_8) was plotted and the system approximation was performed through the H_4 term. The results are self evident. There was no cross-coupling degradation from the

approximation techniques. Again, the response of the approximated system are extremely close to the original system.

CHAPTER VII

CONCLUSIONS

An analog study of the matrix Cauer model has been presented. It is a natural extension of Chen-Shieh [6] technique for model simplification. The advantages of the method are that one has several approximations after once performing the expansion and the accuracies of the approximations are excellent. The simplest model is an approximation of the first order (H_1 and H_2) only; however, in practice, H_1 through H_4 would be as simple an approximation as one will use. When one takes more and more terms of the expansion, better and better results are obtained.

Multiple input/output systems are important in practice and difficult in analysis and they are almost impossible to design theoretically. This thesis offers a decomposition method to simplify the problem and to make the simulation practicable.

In introducing the matrix Cauer model,analog computer techniques were evolved for the scalar Cauer model. These techniques, as well as matrix model techniques, should be helpful to the designer from the standpoint of ease of simulation. Once the general programs are patched it is a simple task to plug in the values for a system and view the model's responses.

The analog computer programs generated using the second Cauer form are well suited to "canned program" environments. This is evidenced by

the lack of need to scale the examples because the second Cauer form inner loops are only active during transient conditions; therefore, if the outer loop is within the capabilities of the analog computer, scaling will probably not be needed. If the inner loops overload, the input amplitude can be reduced.

Hopefully, this paper will help take the analog computer out of the hands of the professional analog programmer and put it where it has always belonged --- in the hands of the engineer.

BIBLIOGRAPHY

[1] Nixon, F. E. Principles of Automatic Control. New Jersey: Prentice-Hall Publishing Company, 1959.

[2] Evans, W. R. Control System Dynamics. New York: McGraw-Hill Book Company, 1954.

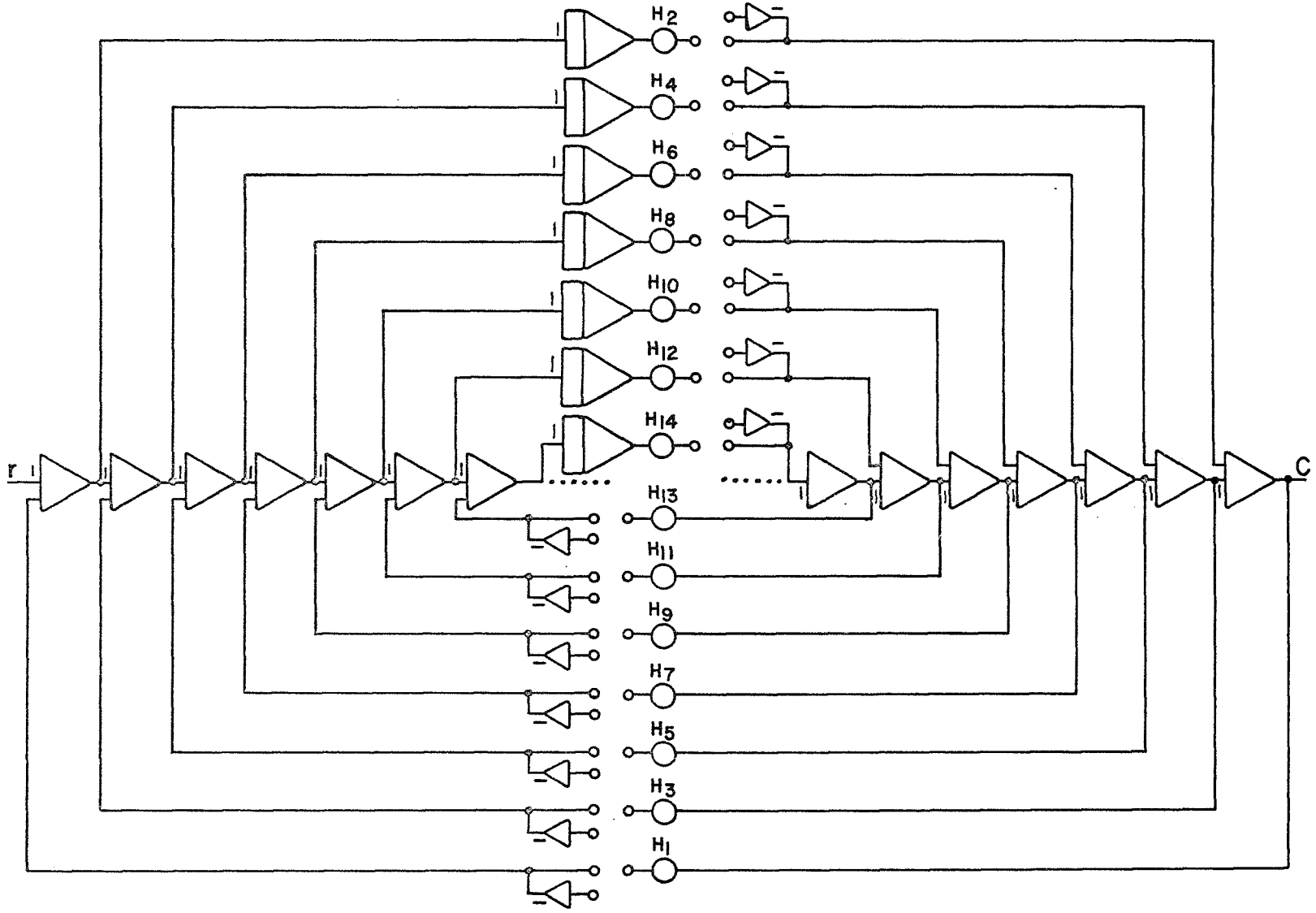
[3] Chen, K. Trans. Am. Inst. Elect. Engrs. PP 76-80. 1957.

[4] Davison, E. J., I.E.E.E. Trans. Autom. Control, 11-102.

[5] Marshall, S. A. Control, p. 642. 1966.

[6] Chen, C. F. and Shieh, L. S. "A Novel Approach to Linear Model Simplification," 1968 Joint Automatic Control Conference. Michigan: pp 454-461.

[7] Chestnut, H. and Mayer, R. W. Servomechanisms and Regulating System Design, pp. 477. New York: John Wiley & Sons, 1959.



APPENDIX A

General Analog Program for Scalar Cauer Form