COMMON

Proceedings of the San Francisco Meeting
December 11-13, 1967

GENERAL MEETING INFORMATION

Meeting Rooms

All meeting rooms will be in the Sheraton Palace Hotel, San Francisco. Specific room assignments are noted in the agenda.

Room Reservations

Reservations must be made by submitting the reservation card directly to the hotel.

Registration

Registration fees for COMMON cover all publication and operating costs and represent the only source of funds for the organization. Registration for this meeting has been set at $30 for the entire meeting, which includes the cost of the luncheon on Tuesday. The registration desk at the hotel will be open during the following hours:

Sunday,	December 10, 1967	5:00 p.m. - 9:30 p.m.
Monday,	December 11, 1967	7:45 a.m. - 5:00 p.m.
Tuesday,	December 12, 1967	8:00 a.m. - 3:00 p.m.

Birds - of - a - Feather Sessions

Attendees may schedule additional informal sessions by posting a notice on the bulletin board.

Systems Reference Library Publications (SRL)

A selection of 1620, 1130, 1800, and S/360 publications are on display in the registration area. You may order an SRL publication by filling out an order form provided. You should expect to receive the items ordered in about 3 weeks.

Computer Equipment Displays

Arrangements have been made for delegates to attend demonstrations and receive time on the San Francisco IBM Data Center machines including the 1130, 360/20, 360/30, and 360/40.

Meeting Headquarters

Typing and reproduction services will be available in the headquarters room: Regency

Agenda Notes

The agenda listed herein represents the program at the time the booklet went to press. Modifications and/or additions will be noted on the information board in the coffee area. Abstracts for the papers listed in the program appear at the end of this booklet.

LETTER TO MEMBERSHIP
FROM PRESIDENT OF COMMON

Members of COMMON:

With this letter is the preliminary agenda for the December meeting of COMMON, in San Francisco. The September meeting in Cincinnati was a good one; we expect the San Francisco meeting to be better. With your help and cooperation, we believe that each meeting can continue to be better than its predecessors.

There are several ways in which COMMON can affect IBM and the world in general; the majority of them, branch office, project liaison, etc., are for today's problems, but two are specifically for the future. The first is our comments and suggestions to the U.S.A. Standards Institute (USASI). Lynn Yarbrough, of USASI's X3.4 Committee on Programming Languages, will have a session on the Standards Institute: its organization, how it affects us, and what we can do to influence it. There will be time for questions and comments.

We will have a brief discussion of the Systems Objectives and Requirements Committee (SORC). This is a committee of user and IBM representatives, whose purpose is to provide IBM with the benefit of informed user opinion in determining the requirements for the next cycle. There will not be a formal session on SORC; their purpose is to acquire information, and not to dispense it. There will be an explanation of it at the general assembly, and there will be opportunities to discuss your ideas with SORC representatives. There is a serious lack of representation in SORC of the small computer user and process control installation. Remember that SORC is concerned almost exclusively with the problems of the future.

We have vociferously exercised our right to criticize the products of both IBM and USAI; we are morally obligated to do what we can to alleviate the situation for the next generation. And our most important contribution is information about our requirements.

See you in San Francisco.

James C. Stansbury

During the first half of the 1960's while IBM built and marketed the "1620" Computer, COMMON was known as "The 1620 Users Group." It was the "SHARE" of the small scientific computer user. All "1620" installations became members by applying and sending a representative to any one of a dozen or so regional or national meetings occurring during any two year period.

As the third generation computers began to appear and it became apparent that the days of the IBM "1620" were numbered, the name of organization was changed to "COMMON" and a myriad of regional and national meetings was reduced to three national meetings per year.

The COMMON Executive Board met in Chicago on September 15 and 16, 1966, to discuss ways of improving the effectiveness of the organization. The board members felt that an organizational change was necessary to cope with the imminent growth of COMMON, the integration into the organization of multiple machine types (1130, 1800, 1620, 360/30, 360/40 and 360/44), and the development of increasing numbers of interest areas by the members.

It was decided to break the organization down into four main divisions along the lines of the other two IBM User Groups "SHARE" and "GUIDE". Each division would be divided into projects and the projects would be further subdivided into working committees as needed.

The divisions established were as follows:

1. Systems Division. This division has broad responsibility for coordination of user activity and opinions in the areas of software and hardware regardless of application.

2. Administrative Division. This division has responsibility for the various activities which are necessary to operate COMMON as an organization, both internally and in relation to other organizations. These include the newsletter, CAST, bylaws, membership list, program library, meeting plans, etc.

3. Installation Management Division. This division has responsibility for coordinating those user activities which are concerned with the management of a computer installation. These include personnel training, standards, computing center operation, etc.

4. Applications Division. This division has responsibility for coordinating user activities which are application or industry oriented.

Each division is headed by a Division Manager who supervises the operation of his division and reports the activities and requirements of his division to the Executive Board through the Executive Vice President.

Each division is composed of a number of projects. Each project is headed by a Project Manager who is responsible for specific areas of interest within the division and who reports to the Dvision Manager.

Each project is divided into committees each of which is headed by a Committee Chairman. The committee chairmen have responsiblity for seeing that their committees carry out the duties within their areas of interest and for seeing that the consensus of opinion of the users at the committee meetings are transmitted to the Project Managers.

Although in some areas committees may be divided into special working subcommittees dealing with specific areas, it is intended that the participation of users in the activities of COMMON be carried out at the committee level. I.B.M. supplies liaison people to the established committees so that user's opinions and requests are transmitted to I.B.M.

The committees meet at every COMMON meeting and these sessions are open to any user who has an interest in the topics covered by the committee. Some meetings on the agenda are listed as project meetings. Most of these are intended to be general discussions by groups of users having the same machine types and are open to all users. These sessions replace the general hardware meetings and problem-solving sessions that occupied much of the 1620 Users Group meetings.

There are also opportunities at every COMMON meeting for users with special interests for which no formal project or committee exist to get together and discuss their problems and techniques. These are called "birds-of-a-feather" sessions and are organized by groups of users posting a notice on the bulletin board and holding a meeting. If enough interest persists, a committee may be formally created and meetings scheduled at future Common meetings.

COMMON OFFICERS

President

James C. Stansbury
Halcon International
Two Park Avenue
New York, New York 10016

Secretary - Treasurer

Charles E. Maudlin, Jr.
Computer Center
Texas Woman's University
Denton, Texas 76204

Eastern Region President

Norman Goldman
Boston University
111 Cummington St.
Boston, Massachusetts 02215

Western Region President

William G. Lane
Director of Computer Sciences
Chico State College
Chico, California 95926

European Region President (Acting)

Dr. Hans Tompa
European Research Associates S.A.
95 Rue Gatti de Gamond
Brussels 18, Belgium

Mid-West Region President

Mrs. Laura B. Austin
General Supervisor
Computer Services
General Motors Institute
1700 W. Third Avenue
Flint, Michigan 48502

Canadian Region President

Stuart D. Baxter
National Research Council of Canada
Computation Centre M23A
Montreal Road
Ottawa, Ontario, Canada

Board Members at Large

Frank H. Maskiell
McGraw-Edison Power Systems Division
Box 330
Cannonsburg, Pa. 15317

Richard L. Pratt
Data Corporation
7500 Old Xenia Pike
Dayton, Ohio 45432

Paul A. Bickford
Computer Center
DePauw University
Greencastle, Indiana

COMMON COMMITTEE OFFICERS

ADMINISTRATIVE DIVISION

Manager - Laura B. Austin (3070) General
General Motors

Reference Manual

Chairman - Dr. Raymond E. Roth (1715)
State University College

Contributed Program Library

Chairman - Miss B. Baber (1454)
National Education Assn.

(Prep Forms)

CPL Shipment Analysis (1303)
Chairman - J. H. Keith
Miami - Dade Junior College North

Jug Inter-Library Exchange

Chairman - W. A. DeLagall (1582)
Schering Corp.

APPLICATIONS DIVISION

Manager - F. H. Maskiell (3081)
McGraw-Edison Co.

Techniques Project

Chairman - W. Pease, Jr. (1516)
West Virginia Pulp & Paper Co.

Utilities

Chairman - G. Haralampu (1041)
New England Electric System

Civil Mechanical Engineering

Chairman - L. W. Brown (3301)
International Paper Co.

Chemical Engineering

Chairman - G. Hertel
U. S. Rubber Co.

Education

Chairman - M. Goldberg (1416)
Fordham University

INSTALLATION MANAGEMENT DIVISION

Manager - C. Baker
Pioneer Hi-Bred Corn Company

Personnel

Chairman - R. B. Thomas (3038)
Federal Reserve Bank

Operations

Chairman - R. J. Snailer (1495)
Metropolitan Life Ins. Co.

Standards (Inactive)

SYSTEMS DIVISION

Manager - J. S. Taylor (3121)
Data Corporation

System/360

Project Manager - R.L. Pratt (3121)
Data Corporation

DOS

Committee Chairman - D. R. McIlvain (1100)
Catalytic Construction Co.

OS

Committee Chairman - Wade Norton
Southern Services, Inc.

S/360-44

Committee Chairman - A. G. Wigdahl (3134)
Allen Brady Company

Hardware

Committee Chairman - J. L. Tunney, Jr. (3076)
James R. Ahart & Associates

360 Commercial Users

Chairman - Mr. F. Hatfield (3008)
Line Materials Industries

1800

Project Manager - C. R. Pearson (1497)
J. P. Stevens & Co., Inc.

(Non-Control)

Software

Chairman - R. Cox (1132)
Humble Oil Company

Applications

Chairman - J. C. Deck (3237)
Inland Steel Co.

Hardware & RPQs

Chairman - W. Barnes (531)
Pacific Gas & Electric

(Control)

Process Systems

Chairman - L. Jones (5046)
Bonneville Power Adm.

Software

Chairman - J. K. Tanatra (3462)
GMC Truck & Coach Division

Hardware

Chairman - D. L. Kraatz (5255)
Corn Products Company

Applications

Chairman - Dr. W. Carlson (3009)
Champion Papers, Inc.

1130 Project

Chairman - D. Dunsmore (3428)
Ohio River Valley Commission

1620 Project

Chairman - Richard Ross
University of Mississippi

SESSION M. O.

Monday 8:00-8:45 a.m.

M.O. 1 New Members Session

Chairman: James Stansbury, President

Room: Comstock

Subject: Topics of interest and concern for those attending COMMON for the first time.

M.O. 2 Session and Panel Chairmen Briefing

Chairman: Robert Forstrom

Room: English

Subject: General information and arrangements briefing for session and panel chairmen.

SESSION M. 1

Monday 9:00-10:00 a.m.

M. 1. 1. General Session

Chairman: James Stansbury, President

Room: Rose

Subjects:
a. Introduction of COMMON Officers and IBM Representatives.
b. Reports from Divisional Chairmen
c. Secretary-Treasurer's Report
d. Arrangements Chairman's Announcements
e. Program Chairman's Announcements
f. Announcements by Members
g. Announcement of Chicago Meeting

COFFEE BREAK 10:00 - 10:30

SESSION M.2

Monday 10:30 - 12:00 Noon

M.2.1 360 Project

Chairman: Richard Pratt

Room: Rose

Subject. a. COBOL to PL/1 Conversion: IBM
b. FORTRAN to PL/1 Conversion: IBM

M.2.2 1130 Project

Chairman: Dave Dunsmore

Room: California

Subject: a. Dynamic Model Simulator for the IBM 1130: George Polyzoides
b. Three-and-Four-Bar Linkage System with Plotted Output: Robert Cushman

M.2.3 1620 Project

Chairman: Richard D. Ross

Room: Golden Gate

Subject: Tutorial Sessions Non-IBM Compilers

a. Data SPS: Richard Ross
b. One Dimensional Blast Wave Theory For Explosions: J. Goresh and J. Caslin
c. Discussion of Related Topics

M.2.4 1800 Project

Chairman: Robert Forstrom

Room: Comstock

Subject: Multiprogramming Executive, MPX: D. Schade (IBM)

M.2.5 Numerical Control Project

Chairman: J. Moschetti

Room: Royal Suite (262)

Subject: Discussion of Problems Relating to Numerical Control.

M.2.6 Installation Management and Administration

Chairman: Laurence Baker

Room: Forty-Niner

Subject:
- a. Systems Reference Library SRL: G. Goesch (IBM)
- b. Program Library Discussion: L. Austin
  PREP forms for 1130, 1800 and 360 libraries

SESSION M. 3

Monday 1:30 - 3:00 p.m.

M.3.1 360 Project

Chairman: Richard Pratt

Room: Rose

Subject:
a. OS/DOS Data Management: William Post (IBM)
b. OS/DOS Linkage Editor: William Post (IBM)

M.3.2 1130 Project

Chairman: Dave Dunsmore

Room: California

Subject: Monitor Version II: G. Lester (IBM)

M.3.3 1620 Project

Chairman: Tony Ross

Room: Golden Gate

Subject: Tutorial Session (Continued)

a. Kingston and Forgo FORTRAN: F. Windham and G. Smith
b. A Batch Processing FORTRAN system for a minimal configuration 1620: Gaylord Henry
c. Discussion of Related Topics

M.3.4 1800 Project

Chairman: Robert Forstrom

Room: Comstock

Subject: 1800 MPX: B. Lanek (IBM)

M.3.6 Numerical Control Project

Chairman: J. Moschetti

Room: Royal Suite (262)

Subject: Discussion of Problems Relating to Numerical Control

M.3.7 Installation Management

Chairman: Laurence Baker

Room: Forty-Niner

Subject: 360 Operator Training: H. Cadow (IBM)

M.3.8 Administration

Chairman: L. D. Yarbrough

Room: Bonanza

Panel Discussions: USASI Standards

Panelists: T. B. Steel, Jr., Chairman, USASI X3.4
L. D. Yarbrough, X3.4 and COMMON
(Others to be announced)

Subject:
a. Introduction to USASI: Mr. Steel
b. Current USASI Activities: Mr. Yarbrough
c. Fortran: USASI vs 1620 vs 360

COFFEE BREAK 3:00 - 3:30 p.m.

SESSION M.4

Monday 3:30 - 5:00 p.m.

M.4.1 360 DOS Committee

Chairman: Don McIlvain

Room: Rose

Subject: SOUND-OFF

Members sound off to IBM regarding suggestions, problems and needs for DOS.

M.4.2 360 OS Committee

Chairman: ~~O. B. Anderson~~ WADE NORTON

Room: Royal Suite (264)

Subject: SOUND-OFF

a. Members sound off to IBM regarding suggestions, problems and needs for OS.

b. Organization of OS Committee

M.4.3 360-44 Committee

Chairman: Allen B. Wigdahl

Room: Royal Suite (260)

Subject: SOUND-OFF

Members sound off to IBM regarding suggestions, problems and needs for System 360-44.

M.4.4 1130 Project

Chairman: Dave Dunsmore

Room: California

Subject: Continuation of monitor Ver II: G. Lester (IBM)

SOUND - OFF For System 1130 will be Held 7:30 to 9:00 p.m.

M.4.5 1620 Project

Chairman: Frank Windham

Room: Golden Gate

Subject: Monitor I Papers

a. A Large 1620 DOS, Reasonably 7094 Compatible: Lanny Hoffman
b. Basic Problems of Information Retrieval and a Solution on the 1620: Lanny Hoffman
c. Discussion of Monitor I

M.4.6 1800 Project

Chairman: Robert Forstrom

Room: Comstock

Subject: Teleprocessing Support on 1800/1070/1050/2740/1030: R. Smith (IBM)

M.4.7 Techniques Project

Chairman: W. Pease

Room: Parlor A (214)

Subject:
a. Fast Fourier Transforms with Applications for the IBM 1800: Joe Howard Smith
b. NIMS - The Aerospace Scientific Data Reduction Monitor System: Charles R. Aumann

M.4.8 Installation Management

Chairman: Laurence Baker

Room: Forty-Niner

Subject:
a. Standards in Operation: D. Fulz (IBM)
b. Investigation of Abnormal Operation Conditions: K. Anderson (IBM)

M.4.9 Administration

Chairman: L. D. Yarbrough

Room: Bonanza

Subject: Current Research Concerned with the Standardization and Formal Definitions of PL/1: Dr. J.A.N. Lee

SESSION T. 1

Tuesday 8:30 - 10:00 a.m.

T.1.1 360 Project

Chairman: Richard Pratt

Room: Concert

Subject: a. A Small OS System: Wa[redacted] Norton
b. Panel Discussion: Does OS Belong in COMMON?

T.1.3 800 Project

Chairman: Robert Forstrom

Room: Comstock

Subject: PL-1 Language Development COP: C. Burwick (IBM)

T.1.4 Electric Utility Project - 1130 Working Group

Chairman: S. A. Clark

Room: Golden Gate

Subject: a. 1130 Load Flow
b. 1130 Short Circuit Calculations

T.1.5 Electric Utility Project - 360 Working Group

Chairman: O. B. Anderson, Jr.

Room: Royal Suite (260)

Subject: a. 360 Load Flow
b. 360 Short Circuit Calculations

T.1.6 Electric Utility Project - 1800 Working Group

Chairman: R. W. Page

Room: English

Subject:
a. 1800 Load Flow
b. 1800 Short Circuit Calculations

T.1.7 Techniques Project

Chairman: W. Pease

Room: Parlor A (214)

Subject:
a. Data Collection: G. A. Gallaway
b. Large Matrix Inversion on a Small Computer
T. E. Bridge

T.1.8 Education Project

Chairman: Jack Underwood

Room: Bonanza

Subject:
a. Chico State College Registration/Scheduling Analysis: Neil McIntyre
b. Student Information System of Christian Brothers College Employing the IBM 1130: Brother Jerome Wegener

T.1.9 Installation Management

Chairman: Laurence Baker

Room: Forty-Niner

Subject:
a. Programmer Expectation: D. Mayer (IBM)
b. Programmer Selection and Testing: D. Mayer (IBM)

COFFEE BREAK 10:00 - 10:30

SESSION T.2

Tuesday 10:30 - 12:00 Noon

T.2.1 360 Project

Chairman: Richard Pratt

Room: Concert

Subject: 44PS and its differences from OS: D. Rumney (IBM)

T.2.2 1130 Project

Chairman: Dave Dunsmore

Room: California

Subject: 1130 as a Terminal for S/360: K. Gabbert (IBM)

T.2.3 1800 Project

Chairman: Robert Forstrom

Room: Comstock

Subject: a. Installation Descriptions
Bonneville Power: B. Hoffman
Colorado Public Service Corp: E. McLaughlin
b. ~~2310-2011: R. Conrad~~

from W.1.7 →

T.2.4 Electric Utility Project - 1130 Working Group

Chairman: S. A. Clark

Room: Golden Gate

Subject: a. 1130 Hardware Difficulties
b. Sound Off
c. New business

T.2.5 Electric Utility Project - 360 Working Group

Chairman: O.B. Anderson, Jr.

Room: Royal Suite (260)

Subject: a. 360 Hardware difficulties
b. Sound off
c. New business

T.2.6 Electric Utility Project - 1800 Working Group

Chairman: R. W. Page

Room: English

Subject:
a. 1800 Hardware Difficulties
b. Sound off
c. New business

T.2.7 Techniques Project

Chairman: W. Pease

Room: Parlor A (214)

Subject:
a. Non-linear Regression Analysis with three Independent Variables: T.E. Bridge

b. User Experience with 1130 LP-Moss: S. A. Lynch

T.2.8 Petro Chem Engineering Project

Chairman: Gene Hertel

Room: Forty-Niner

Subject: Application of Simulation in Control, Design and Optimization of Chemical Processes: M.J. Shah (IBM)

T.2.9 Education Project

Chairman: Jack Underwood

Room: Bonanza

Subject: Panel Discussion on Software Requirements in an Instructional Program

DELEGATES LUNCHEON

Tuesday 12:00 Noon

Rose Room

Speaker: Dr. Robert E. Hill
President, Chico State College

Dr. Hill is a recognized authority in the field of international education, finance, investment, and international economics. He has written numerous articles, monographs and short papers and has risen from Assistant Professor of Finance (1957) at the University of Illinois to Professor of Business and President of Chico State College (1966).

Dr. Hill will speak on the subject: "Technology and Administration; A Paradigm."

SESSION T.3

Tuesday 1:30 - 3:00 p.m.

T.3.1 Special Session: Time-Share Computing

Chairman: W. G. Lane

Room: Concert

Subject:
a. Conversational Computing: James Babcock
b. RUSH, Conversational PL-1: Paul DesJardine
c. Computer Assisted Instruction at Stanford University: Max Jerman

T.3.2 360-44 Committee

Chairman: Allen B. Wigdahl

Room: Royal Suite (260)

Subject: Division of Problems related to System 360-44

T.3.3 1800 Project

Chairman: Robert Forstrom

Room: Comstock

Subject: PROSPRO: O. Merklinghouse (IBM)

T.3.4 Electric Utilities Project

Chairman: G. S. Haralampu

Room: California

Subject:
a. Reports of Working Groups
b. Progress reports on fault calculations, transient stability engineering operating systems

T.3.5 Petro Chem Engineering Project

Chairman: Gene Hartel

Room: Forty-Niner

Subject: Laboratory Automation: R. A. Edwards (IBM)

T.3.6. 1130 Project

Chairman: Larry Armbruster

Room:

Subject: LP-MOSS: IBM

COFFEE BREAK 3:00 - 3:30 p.m.

SESSION T.4

Tuesday 3:30 - 5:00 p.m.

T.4.1 Open Board Meeting

Chairman: James Stansbury, President

Room: Rose

Subject:
a. Members Sound Off to Board
b. Discussion of IBM-COMMON Relations
c. COMMON Publications

T.4.2 1800 Project

Chairman: Robert Forstrom

Room: Comstock

Subject:
a. PROSPRO (continued): O. Merklinghouse (IBM)
b. TSX Modification for 6 Disk Drives: E.H. Spencer

T.4.3 Electric Utilities Project

Chairman: G. S. Haralampu

Room: California

Subject:
a. Engineering Data Banks
b. New Business
c. Plans for next meeting

SESSION W.1

Wednesday 8:30 - 10:00 a.m.

W.1.1 360-OS Committee

Chairman: ~~C. B. Anderson, Jr.~~ WADE NORTON

Room: Royal Suite (264)

Subject: a. OS Reread in FORTRAN: Wade Norton

W.1.2 360-DOS Committee

Chairman: Don McIlvain

TO M4.1

Room: Ralston

Subject: a. Discussion of problems related to FORTRAN under DOS

W.1.3 360-Commercial Committee

Chairman: To be announced

Room: Royal Suite (262)

Subject: a. Discussion of problems related to COBOL, RPG, COS

W.1.4 360-44 Committee

Chairman: Allen B. Wigdahl

Room: Royal Suite (260)

Subject: Discussion of problems relating to System 360-44

W.1.5 1130 And Techniques Projects

Chairman: Dave Dunsmore

Room: Bonanza

Subject: a. Overlapped printing for IBM 1130 Commercial Applications Using FORTRAN Write Statement: Brian Swain

b. 1130 Commercial Subroutines

W.1.6 1620 Project

Chairman: Richard Karpinski

Room: Golden Gate

Subject: General information on CAI with special emphasis on 1620 version of COMPUTEST

W.1.7 1800 Project

Chairman: Robert Forstrom

Room: California

Subject: a. A comparison between 2310 and 2311 Disk Storage Systems: Salomon Saroussi
b. Process Control in Natural Gas Transmission: A. A. Douloff

To T2.3 6

W.1.8 Administration Project

Chairman: Laura Austin

Room: English

Subject: a. Discussion of Reference Manual Project
b. Call for help

W.1.9 Installation Management

Chairman: Laurence Baker

Room: Forty-niner

Subject: Systems and Programming Project Management: Ralph Sackman, Jr.

COFFEE BREAK 10:00 - 10:30 a.m.

SESSION W.2

Wednesday 10:30 - 12:00 Noon

W.2.1 360-0S Committee

Chairman: ~~O. B. Anderson, Jr.~~ WADE NORTON

Room: Royal Suite (264)

Subject: RAX: D Madden (IBM)

W.2.2 360-DOS Committee

Chairman: Don McIlvain

Room: Ralston

Subject: Discussion of problems relating to PL-1

W.2.3 360-44 Committee

Chairman: Allen B. Wigdahl

Room: Royal Suite (260)

Subject: Discussion of problems relating to System 360-44

merge

W.2.4 1130 Project

Chairman: Dave Dunsmore

Room: Bonanza

Subject: Discussion of problems relating to 1130

W.2.5 1620 Project 1130 CSMP also.

Chairman: Richard Ross

Room: Golden Gate

Subject: a. Continuation of CAI discussion
b. Suggestions and Wind up
c. Plans for next meeting

W.2.6 1800 Project

Chairman: Robert Forstrom

Room: California

Subject:
a. User Experience with Teletype Terminals on the IBM 1800: W. M. Schonlau
b. Plans for next meeting

W.2.7 Techniques Project

Chairman: W. Pease

Room: English

Subject:
a. Summation and Review
b. Plans for future meetings

W.2.8 Petro Chem Engineering Project

Chairman: Gene Hertel

Room: Royal Suite (262)

Subject:
a. Summation and Review
b. Plans for future meetings

W.2.9 Installation Management

Chairman: Laurence Baker

Room: Forty-Niner

Subject: A Review of the GUIDE Installation Management Project: Arthur Nichols

SPECIAL ATTRACTION

Wednesday Afternoon

IBM San Jose Plant Tour

a. 1500 and 1800 Production line
b. Direct access storage devices
c. Demonstration of 1800 MPX

Complementary Transportation by IBM Charter Bus

Leave	San Francisco	1:15 PM
Rtn	San Francisco Airport	4:45 PM
Rtn	San Francisco (Sheraton Palace Hotel)	5:30 PM

Reservations must be made at the registration desk by 12:00 Noon Tuesday.

Delegates may choose to tour the facilities (a and b) or to attend the demonstration (c). We regret that time does not permit both.

SESSION W.3

Wednesday 1:30 - 3:00 PM

W.3.1 360 Project

Chairman: Richard Pratt

Room: Ralston

Subject:
a. Reports of 360 committees
b. Recommendations to IBM
c. Plans for future meetings

W.3.2 1130 Project

Chairman: Dave Dunsmore

Room: Bonanza

Subject:
a. Suggestions and Improvements
b. Plans for future meetings

Session M.O

M.0.1 New Members Session

At past meetings the orientation session for new members of COMMON has always taken place at the end of the first day. This arrangement has the result of permitting many new members to walk around in a fog for one-third of the meeting! In an attempt to remedy this situation we have scheduled a short session before the meeting proper begins to familiarize new members with the workings of COMMON and to make them feel more at ease at the sessions that follow.

Session M.1

M.1.1 General Session

This meeting will be chaired by the president of COMMON, Mr. James Stansbury and will be in the nature of a formal welcome to the attendees from the Executive Board. There are no concurrent sessions, so that all registrants may be present. In addition to the usual introductions of COMMON Officers and special guests there will be time reserved for last minute changes to be announced by the Arrangements Chairman and the Program Chairman. If possible, procedural questions from the floor will be invited.

Session M.2

DYNAMIC MODEL SIMULATOR FOR THE IBM 1130

The Dynamic Model Simulator is a chain of FORTRAN source programs that permits the IBM 1130 user to investigate in depth the dynamic behavior of physical systems that can be modeled into linear or nonlinear ordinary differential equations.

It can be used for the investigation of a wide range of engineering and mathematical problems from reaction kinetics and reactor responses to the design of electrical networks and structural assemblies.

The system does not require familiarity with analog computers although in a way it forces the IBM 1130 hardware to perform the function of an analog computer while at the same time it offers the digital computer advantages of random memory access and ten digit (extended) precision.

The input consists of English command words (START, WAIT, RESET, etc.) and numeric data which are read in free format.

The system output includes tabulated values of the variables at specified intervals, a table of the variables involved and detailed error and information messages. Plotting of the variables is also available as an option.

System Source Language: FORTRAN IV, Level E

System Hardware Requirements: 1131 CPU-8K, one 2315 disk cartridge
1442 Card Reader
1132 Printer
Benson-Lehner 305 Digital Incremental Plotter (optional)

PROGRAM LIBRARY

Discussion of the proposed PREP Forms for the 1130, 1800 and 360 Library. There will also be a discussion of an extension of the minimal standards for this Library. It is requested that there be representatives from each of the machine system projects in the Systems Division present at this meeting.

THREE-BAR AND FOUR-BAR LINKAGE SYSTEM WITH PLOTTED OUTPUT

A mathematical model of a linkage system is controlled by an array of inputs to plot out the path of the linkage intersection point. In addition, the system simulates turning the entire mechanism about a major axis. The output of the system has yielded some very interesting designs.

In conjunction with this report, a list-oriented free field floating point input program has been developed. This routine enables the control program to modify only specified items of the system input. Under break character control, an automatic interactive procedure can be set up.

Session M.4

'A LARGE 1620 DISK OPERATING SYSTEM, REASONABLY 7094 COMPATIBLE.'

A new monitor for a 40K or 60K 1620 has been developed at Princeton for use as a debugging aid for the 7094 and some 360/OS programs. FN II I/O and FN IV I/O are included as well as private storage of programs. Many control card options exist. Speed can be 2 to 4 times faster than MON-I (if one has hardware FLT. PT.) both in compilation and execution. A users manual does exist. Many students are presently using the system on a completely open-shop basis.

'BASIC PROBLEMS OF INFORMATION RETRIEVAL AND A SOLUTION ON THE 1620.'

Some of the basic problems of information retrieval and implementation are discussed. The techniques of file-organization are discussed. These techniques are discussed for the most common file devices (disk and tape). A solution is shown for the 1620 with disk to demonstrate disk utility.

FAST FOURIER TRANSFORM WITH APPLICATIONS FOR THE IBM 1800

The Fast Fourier Transform technique as developed by Cooley and Tukey has had a widespread effect in the field of time series analysis. However, some difficulty has been encountered by potential users in determining exactly how the technique works. An effort to explain the derivation in detail will be made

Also, a description will be given of an analysis package program in which the Fast Fourier Transform technique is used to yield amplitude/phase spectra, power spectra, cross power spectra, and auto correlations.

NIMS- THE AEROSPACE SCIENTIFIC DATA REDUCTION MONITOR SYSTEM

The IBM-TSX System was designed for an IBM - 1800 operating in a real time environment as a process controller. The Aerospace IBM-1800 is used primarily as a post flight data reduction system interacting with telemetry ground station equipment. This demands the full interrupt facilities of the system but in a batch processing mode of operation such as offered by the TASK stand alone type system. The resulting NIMS Monitor System combines the features of the process and non-process modes of operation into a general mode of operation in which the full interrupt and peripheral facilites of the machine are available to the active application program under execution.

The system basically is a TASK OFF-LINE system with the system components modified to allow Process Subroutines and Interrupt Service Subroutines to be included in the core load generation. Other features include a one card cold start, ability to assemble or compile from magnetic tape input, and a magnetic tape backup system. In addition, the TRACE and CORE DUMP routines of TASK have been modified to provide and extended mnemonic output listing.

CURRENT RESEARCH CONCERNED WITH THE STANDARDIZATION AND FORMAL DEFINITION OF PL/I

In April 1966, a subcommittee of USASI Task Group X3.4.2 was established to investigate the standardization of PL/I. Since that time, two working committees have established with explicit charters:

Group I: The resolution of the language PL/I

and

Group II: Formal Definition of PL/I

This paper deals specifically with the work of the committee on Formal Definition and its relations with the definitional task forces within IBM. This paper will discuss the techniques of definition as proposed by the committee, the uses to which the definition is to be put and the implications of the work of this committee on the standardization of other computer languages.

A familiarity with PL/I is not presumed.

Session T. 1

THE 1620 AS A DATA COLLECTOR OR SOFTWARE, THE CRUTCH HARDWARE BOYS LEAN ON

The paper will cover the date collection methods used at the Sacramento Peak Observatory. The principle reasons for designing a computerized data collection system were :

1) Improve existing methods which used summary punches.
2) Gain experience and insight into problems associated with data collection in preparation for third generation equipment.

The paper will also cover the roles played by the programming staff in this type of operation. Namely, software as a diagnostic tool for the hardware boys in developing data reduction/collection instruments, and software as a useful and flexible tool for the research scientist.

Three methods for data collection and reduction by computer have been tried.

1) On line, one point at a time, testing each point for parity and structural errors and storing it on the disk before the next point is generated.

2) On line, a record at a time, locking the source device out until the data has been tested and stored.

3) Off line, using 7 track incremental tape recorders, with testing and reduction being done during slack times.

CHICO STATE COLLEGE REGISTRATION AND SCHEDULING ANALYSIS

"Salient features of Chico State College's computer approach to student scheduling and registration are discussed; applicability to other colleges and other computer configurations is emphasized."

STUDENT INFORMATION SYSTEM OF CHRISTIAN BROTHERS COLLEGE EMPLOYING THE IBM 1130

We received our 1130 around July 4 and immediately began programming it for our Student Information System. This is a great improvement over the 1620 system since that was chiefly card oriented and we can now keep all the information on the disk. All of the programs are written in FORTRAN with the exception of two or three adapted with the Commercial Subroutine Package. After the students are registered, we are able to produce class lists, student schedules, mailing labels, all types of statistics, statements, report cards, and permanent record labels. The system works very well and will become more efficient when we add an additional two disks and receive our 1403 printer.

USER EXPERIENCE WITH 1130 LP-MOSS

U. S. Reduction Co. has used the 1130 LP-MOSS application package since its first release. Various problems have been encountered during its use; however, useful answers have been obtained. Due to the size of the aluminum alloy blending problem being studied, a second 1130 had to be leased for full dedication to this problem. Early results indicate a reasonable payback may be obtained from this hardware and software combination.

An Outline

1. Company background
2. Early uses of linear programming
3. Problems encountered in implementing 1130 LP–MOSS
4. Results to date and future planning
5. Specific application modifications desired

APPLICATION OF SIMULATION IN CONTROL, DESIGN AND OPTIMIZATION OF CHEMICAL PROCESSES

The problem of supervisory control and optimization of a chemical process requires that the control as well as the optimization programs be provided with a mathematical relationship between the dependent and independent variables in the process.

In this presentation we will discuss methods of arriving at this relationship based on the fundamental laws of physics and chemistry. We will make the proper approximations valid for control purposes to obtain the solution of the differential equations, describing this relationship. Use of simulation languages helps in reducing the programming time as well as the number of trials required to attain successful solutions.

An example of methanol plant will be discussed in some detail to illustrate the mathematical and programming techniques to achieve the desired relationships for control and optimization. It will be shown how these differential equations with modification of the objective function when used in the optimization program can lead to optimum design of the plant.

LABORATORY AUTOMATION

The session on Laboratory Automation will include a general description of programming and equipment aspects of the application of the 1800 and 1130 to this newly emerging and fast growing field. Following this discussion, an application will be discussed in detail. This will include a description of the type of research to be accomplished, the incentives of the on-line computer, the programming system to run the instrument in a closed loop fashion, and other experiences to date with the system.

OVERLAPPED PRINTING FOR IBM 1130 COMMERCIAL APPLICATIONS USING FORTRAN WRITE STATEMENT.

A method is described for incorporating overlapped printing into IBM 1130 Commercial FORTRAN programs. Communication to the sub-subprogram which performs a printing operation is achieved through the FORTRAN WRITE statement rather than through the CALL statement. The advantage of this method is that limited use can be made of the formatting ability of the FORTRAN language. Headings can readily be incorporated, and the layout of the printed page specified by use of FORMAT statements.

A BATCH PROCESSING FORTRAN SYSTEM FOR A MINIMAL CONFIGURATION IBM 1620

This paper describes a software package designed to increase throughput on a 20K card system IBM 1620 computer in an environment where a large number of fairly simple FORTRAN programs must be processed. The increase in throughput is accomplished in two ways; (1) programs are batch executed under control of a loader-monitor routine, and (2) a powerful precompiler reduces the number of execution errors and decreases the requirement for on-line debugging.

The compiler is a version of the PDQ FORTRAN compiler which has been modified to handle the batch processing features of the system. Batch execution is made possible by keeping the entire subroutine library resident in core rather than reloading it with each object deck. Object programs, separated by control cards, are stacked for input. The reading of a control card by the FORTRAN card read subroutine or the execution of a CALL EXIT statement will cause the next object deck to be automatically loaded and executed. The monitor will also terminate a job if the output line count exceeds a control card specification.

The precompiler detects more than seventy distinct errors based on PDQ FORTRAN specifications. Many of these errors, such as undefined symbols, are undetected by the compiler and cause execution checkstops. Recognition of multiple errors in a single statement is possible, thus eliminating multiple debugging passes.

A COMPARISON BETWEEN 2310 AND 2311 DISK STORAGE SYSTEMS

A series of programs has been written to allow 1800 users to fully utilize the capabilities of the IBM 2841-2311 disk storage system within the frame work of the Time-Sharing Executive -- TSX, Version 3, Mod 1.

A comparison is made between the 2311 and the 2310 disks in programming techniques, timing and cost effectiveness.

PROCESS CONTROL IN NATURAL GAS TRANSMISSION WITH AN IBM. 1800

This paper will present the progress made by Trans-Canada Pipe Lines Limited, Toronto in the field of process control computers. A brief description is given of the feasibility study prior to ordering the computer, the organization of the implementation team and the methods of implementation.

The purpose of installing the process control computer at Trans-Canada is to save fuel by more efficient operation of compressor stations, and to guide the dispatcher into better control of the line, thus allowing more throughput at the same or better operating cost. The computer is not closed loop, but accepts telemetered data from all compressor stations on a priority interrupt basis, optimizes the line by means of an on-line simulation program and then informs the dispatcher by typewriter output what change, if any, to make to achieve optimal operation.

A description is given of the telemetering system that feeds the computer and of the simulation/optimization program that is used to control the line. The computer, the I.B.M. 1800, was installed in June 1967.

SYSTEMS AND PROGRAMMING PROJECT MANAGEMENT

The allocation of limited systems and programming resources to the highest payoff areas is becoming more important as data processing installation costs continue to rise. Long and short-range plans properly approved by top management and formalized systems project management are vital tools to assure that systems capability is focused on the major areas of the enterprise and adequately controlled to attain the stated objectives. To work effectively in this environment, additional demands are placed onto the systems group to develop systems plans in terms easily understood by top management and onto data processing management to bring about fulfillment of the approved plans on time and as economically as possible.

USER EXPERIENCE WITH TELETYPE TERMINALS ON THE IBM 1800

The following describes and addition to the programming system for the IBM 1800 computer. The expanded system will support up to 16 remote teletype terminals being used in a time-sharing environment primarily to solve realtime information processing problems.

1800 Hardware

The 1800 CPU makes extensive use of hardware interrupt levels and data channels to operate its standard data processing I/O equipment. In addition the 1800 can have analog and digital I/O capable of communicating directly with almost any kind of equipment. The terminal system uses 2 words of digital input and 1 word of digital output (16 bits/word) to control all 16 terminals simultaneously. No data channels or additional hardware are employed.

TSX Programming System

The 1800 programming system provides many conveniences. Programs are of two types, process and non-process.

Process programs can be initiated by externally generated interrupts or can be queued by any program for execution. They can be a part of the system skeleton which is in core at all times or they can be kept on disk in core-image form. Process programs have highest priority and generally use some analog or digital I/O.

Non-process programs ordinarily do not use the process (analog and digital) I/O. They are usually stacked jobs of a conventional type to be run under the non-process monitor.

When time-sharing, the system does the following:

(1). Runs non-process programs for background (job shop).
(2). Periodically checks the queue for process programs.
(3). Permits externally or internally generated interrupts to load and execute process programs at any time.

Both (2) and (3) require that the non-process job be saved and restored when the process work is finished.

The Terminal System

The terminals extend the time-sharing capabilities of the system by allowing up to 16 users to communicate with the system simultaneously.

Terminal activities include:

(1). Queing process programs for execution.
(2). Communicating with programs during execution.
(3). Programming in NUtran (conversational fortran).

Many other activities are planned.

Most of the terminal system capabilities are achieved by simply making the standard TSX functions more accessible. The only programming efforts unique to the terminal system are the communications controller (simulated by an in-skeleton program), and the time-slicing of terminal service programs.

COMMON MEETING SAN FRANCISCO (F)

ATTENDEES

AMS	D	AMES RESEARCH CENTER		MOFFETT FIELD CALIF	1800
ALEXANDER	C	CIDEAL TOY CORP		HOLLIS N.Y.	360
ALLBRITTON	E	JDUNDEE CEMENT CO	1800	CLARKSVILLE MO	1800
ALLEN	R	FIBM		CHICAGO ILL	360
ALVES	F	MIBM		SAN JOSE CALIF	
ANDERSON	I	IBM		SAN FRANCISCO CALIF	360
ATKINS	D	FEASTERN ILL UNIV		CHARLESTON ILL	360
AUMANN	C	RAEROSPACE CORP		REDLANDS CALIF	1800
AUSTIN	L	BGENERAL MOTORS INST	3070	FLINT MICH	1620
AYERS	R	DHOWMET CORP		CHICAGO ILL	360
BADEN	J	B		COSTA MESA CALIF	
BADEN	S	FMARSHALL COMMINICTNS		COSTA MESA CALIF	1130
BAILEY	D	CIBM		SARATOGA CALIF	1130
BAKER	L	HPIONEER HIBRED CORN		DES MOINES IOWA	360
BAKKEN	J	EMIDWEST OIL CORP		DENVER COLO	360
BALL	M	JUNIV OF KENTUCKY	1553	LEXINGTON KY	1800
BARABACHEFF	R	CIBM CORP		NEW YORK N.Y.	1130
BARNES	W	RPACIFIC GAS + ELECT		SAN FRANCISCO CALIF	1800
BARR	S	WESTERN ELECTRIC		NEW YORK CITY N.Y.	1800
BATE	M	WAMERICAN SCIENCE-ENG	1776	CAMBRIDGE MASS	360
BEARD	T	EIBM		LOS GATOS	1800
BECK	D	MIBM		RIVERSIDE CALIF	1130
BEITEL	B	JIBM		SAN JOSE CALIF	1800
BELANGER	C	BEAUCHEMIN-BEATON		MONTREAL CANADA	1130
BELL	G	PAFFORD + ASSOCIATES		LOS ANGELES CALIF	1130
BENIGNUS	V	AU OF TEXAS-MEDICAL BR		GALVESTON TEXAS	1800
BERG	R	PIBM		DENVER COLO	1800
BERGER	D	EBROWN ENGINEERING CO		HUNTSVILLE ALABAMA	1130
BERGLUND	A	BIBM CORP		SAN JOSE CALIF	1130
BETHENOD	R	LIBM		OAKLAND CALIF	1130
BIBLER	D	SOAK RIDGE ASSOCD UNIV		OAK RIDGE TENN	1800
BICKFORD	P	ADEPAUW UNIVERSITY	3025	GREENCASTLE IND	1620
BIMA	A	JRYAN AERONAUTICAL		SAN DIEGO CALIF	1130
BLACKNEY	W	CDOW CHEMICAL CO		MIDLAND MICH	1800
BOBAY	J	PCUMMINS ENGINE CO	2400	COLUMBUS INDIANA	1130
BOND	W	FST REGIS PAPER		BROOKHAVEN MISS	1800
BONSON	V	LIBM		CAMPBELL CALIF	360
BOUCHER	R	MTHE UNITED ILLIM CO		NEW HAVEN CONN	1130
BOWEN	R	WUS GEOLOGICAL SURVEY		MENLO PARK CALIF	360
BOYER	V	TIBM WTC		SAN JOSE CALIF	1800
BRAQUET	M	XCENTRAL LA ELECTRIC		NEW IBERIA LA	360
BRIDGE	T	ECATALYTIC CONST CO		SWARTHMORE PA	360
BRIDGEMAN	K	AGENERAL DYNAMICS ELE	1347	ROCHESTER NY	360
BRIGGS	D	RMOBIL OIL CORP		MIDLAND TEXAS	1800
BRODIE	M	PIBM		OAKLAND CALIF	
BROOKS	R	FMONSANTO CO	3438	ST LOUIS MO	1130
BROWNELL	D	NAMERICAN CYNAMID		BOUND BROOK N.J.	1800
BRUCE	J	WCOUNTY OF SAN DIEGO	5162	SAN DIEGO CALIF	1130
BRUMMER	E	ST EDWARDS UNIVERSITY	5228	AUSTIN TEXAS	1620
BUCKLEY	R	EMONSANTO CO		DECATUR ALABAMA	1130

BULTMAN	W CIBM FE TECH OPS		SAN JOSE CALIF	1800
BURGGRABE	W FNOOTER CORP	3418	ST LOUIS MO	1130
CADOW	H WIBM ED DEV		POUGHKEEPSIE N.Y.	
CAFFERTY	J IBM CORP		BINGHAMTON N.Y.	360
CAIN	J EIBM		SANTA BARBARA CALIF	360
CALLIN	W JDUQUESNE LIGHT CO		PITTSBURGH PA	360
CAPLAN	F LIBM		WHITE PALINS N.Y.	
CAPREZ	A TACOMA CITY LIGHT	5292	TACOMA WASH	1130
CARLSON	J WIBM-PTL		ROCHESTER MINN	
CARPENTER	K MIBM		DENVER COLO	1800
CARR	R AIBM		ST CHARLES ILL	360
CARROLL	W JRUTGERS-STATE UNIV		NEWARK N.J.	1130
CASLIN	J CAEROSPACE RESEARCH		DAYTON OHIO	1620
CASTELLAN	A PSUN OIL CO		PHILADELPHIA PA	360
CHAMBERS	F WIBM RESEARCH CENTER		YORKTOWN HTS NY	1130
CHESSIN	P LWORLD TRADE CORP		SAN JOSE CALIF	1130
CHOW	W MUNION CARBIDE	1101	BOUND BROOK N.J.	1130
CLARK	S AP.S.CO.N.H.	1251	MANCHESTER NH	1130
CLAUVE	S MIBM		SARATOGA CALIF	1130
CLOSMAN	S IBM		WHITE PLAINS N.Y.	
CONNELL	J RIBM		REGINA SASK	1130
COX	N WGENERAL VALVE CO		LONG BEACH CALIF	1130
CRESPY	J ABEAUCHEMIN-BEATON-LAP		MONTREAL CANADA	1130
CUMMINGS	W FIBM		ENDICOTT NY	360
DAVIS	R FORDNANCE RESEARCH		STATE COLLEGE PA	1130
DE HAAN	S FORD MOTOR CO		DETROIT MICH	1130
DECK	J CINLAND STEEL RESEARCH	3237	EAST CHICAGO IND	1800
DENISON	G BFAIRBANKS MORSE	3346	BELOJT WISC	360
DENNY	D LCLARK + GROFF ENGRS	5414	SALEM OREGON	1130
DEVINE	J PNASA AMES RES. CENT.		SAN JOSE CALIF	1130
DEWEY	M RDREXEL HARRIMAN RIPLY		PHILADELPHIA PA	1130
DICKERSON	G SSTEEL SERVICE CO INC	1592	NASHVILLE TENN	1130
DICOSTANZO	J AIBM		POUGHKEEPSIE N.Y.	360
DIKUM	N IBM		SAN JOSE CALIF	1130
DONCHIN	E NASA-AMES RES CENTER		MENLO PARK CALIF	1800
DOTY	H RUNIV OF MISSOURI		COLUMBIA MO	1130
DOWELL	R EAGRICO CHEMICAL CO		MEMPHIS TENN	360
DOWELL	D S		MEMPHIS TENN	
DRAVES	R HN.W. STATES PORTLAND		MASON CITY IOWA	1800
DRYBURGH	J RIBM		VICTORIA BC CANADA	1130
DUNSMORE	D AORSANCO	3428	CINCINNATI OHIO	1130
DYE	D RIBMHAWTHORNE N.Y.			
ELENBAAS	J ADOW CHEMICAL	3612	MIDLAND MICH	1130
ELMER	C MIBM		WHITE PLAINS N.Y.	
EMERSON	J TFRESNO STATE COLLEGE		FRESNO CALIF	1620
ERICSON	J OCENTURY ELECTRIC CO		ST LOUIS MO	1130
FAROUGHI	S CHAMPION PAPERS INC		PASADENA TEXAS	1800
FARWELL	R AUNIV OF ALBERTA		EDMONTON ALTA CANADA	1800
FELDMAN	M N.Y. MEDICAL COLLEGE		NEW YORK N.Y.	360
FERRAL	R LUSWB SACRAMENTO RFC		SACRAMENTO CALIF	1620

F[illegible]ER	D	GUNIV OF ALBERTA		EDMONTON ALBERTA CANADA	1800
FITZPATRICK	E	DILLINOIS STATE UNIV	3295	NORMAL ILL	1130
FOERSTER	C	SIBM PC + SSS		SAN JOSE CALIF	1800
FORSTROM	R	WNORTH AMER ROCKWELL		YORBA LINDA CALIF	1800
FORSYTH	D	WCONTINENTAL CAN CO		CHICAGO ILLINOIS	1620
FOUTS	C	AUS ARMY CORPS OF ENG		PORTLAND OREG	1800
FOX	D	RGMC TRUCK + COACH		PONTIAC MICH	1800
FRANNEA	J	AMOBIL OIL CORP		MIDLAND TEXAS	1130
FRICKE	E	CLINFIELD COLLEGE	5407	MC MINNVILLE OREG	1620
FRIEDMAN	J	LKAISER ENGINEERS		OAKLAND CALIF	1130
FRIEDLAND	M	JUS PUBLIC HEALTH SERV		LAS VEGAS NEVADA	1130
FULLAN	D	JIBM CORP		WHITE PLAINS N.Y.	360
GANATRA	B			DETROIT MICH	
GANATRA	J	KGENERAL MOTORS CORP		PONTIAC MICH	1800
GARD	D	AIBM		HAVERSTRAW N.Y.	1800
GARDNER	D	SGENERAL FOODS CORP		TARRYTOWN N.Y.	1130
GHISELLI	J	FAIRCHILD		MT VIEW CALIF	360
GILBERT	D	MPUBLIC SERVICE CO		DENVER COLO	
GLADFELTER	G	WSD SCHL MINES TECH	5120	RAPID CITY SO DAKOTA	1130
GOESCH	G	WIDM		LOS GATOS CALIF	
GOLDMAN	N	BOSTON UNIVERSITY	1089	BOSTON MASS	360
GOLDSTEIN	J	HEMORY UNIVERSITY	1410	ATLANTA GA	1620
GOODE	J	MHALLIBURTON COMPANY		DUNCAN OKLA	1800
GOSSETT	E	CSINCLAIR OIL CORP		ALLENDALE NJ	1800
GOUGH	R	JIBM		LOS ANGELES CALIF	360
G[illegible]Y	W	CWAGNER ELECTRIC CORP	3165	BALLWIN MO	1130
GREEN	D	WWARWICK ELECTRONICS		SEPULVEDA CALIF	360
GREENE	R	AIBM		ENDICOTT NY	360
GRISELL	J	LLAFAYETTE CLINIC		DETROIT MICH	1800
GROSS	P	FUNIV OF SASKATCHEWAN		REGINA SASKATCHWN CANADA	1130
GROVE	M	CIBM		SAN JOSE CALIF	1800
GUDOBBA	R	LAFAYETTE CLINIC	3277	DETROIT MICH	1800
HACKETT	J	EDRESSER-CLARK	1216	OLEAN N.Y.	360
HAHN	D	ACORN PRODUCTS CO	3385	PEKIN ILLINOIS	1800
HALE	M	RHUMAN FACTORS RESCH	5016	LOS ANGELES CALIF	1130
HALLAM	A	FFIRESTONE TIRE		AKRON OHIO	360
HALLING	B	ROLLS-ROYCE LTD		DERBY ENGLAND	1800
HAMIL	W	RIBM-SDD		ROCHESTER MINN	1130
HARALAMPU	G	SNEW ENGLAND ELEC SYS	1041	BOSTON MASS	1130
HATFIELD	F	AMCGRAW EDISON CO	3008	ZANESVILLE OHIO	360
HAYES	M	EIBM DEPT 206		SAN JOSE CALIF	360
HAYWARD	A	PDUQUESNE LIGHT CO	1712	PITTSBURGH PA	360
HEALEY	P	DIBM		SAN JOSE CALIF	1800
HEATH	W	DIBM		MARTINEZ CALIF	1800
HENKE, JR	J	WIBM		OAK PARK MICH	1800
HENRY	G	GIBM		SAN JOSE CALIF	1800
HENSON	R	CBC GOV'T		VICTORIA BC CANADA	360
HERRICK	H	LIBM ERO		NEW YORK CITY N.Y.	
HERTEL	E	SUNIROYAL CHEMICAL		NAUGATUCK CONN	360
HETTINGER	V	EIBM-DP DIVISION		WHITE PLAINS N.Y.	1130

HILL	W	HBENDIX CORP	1613	TETERBORO NJ	360
HIPSKIND	J	G C BESTOR + ASSOC		CARMEL CALIF	1130
HO	W	IBM		SAN JOSE CALIF	1800
HOFFMAN	L	LPRINCETON UNIVERSITY	1170	PRINCETON N.J.	1620
HOWELL	W	EHERCULES INCORP		WILMINGTON DEL	360
HUGHES	R	OMINN POWER + LIGHT		DULUTH MINN	360
HWANG	C	BERKELEY SCIENTIFIC		BERKELEY CALIF	1130
INDIVERI	R	LSINCLAIR OIL CORP	1525	CHERRY HILL N.J.	360
IWASAKI	S	PALO ALTO UNFD SCH D	5210	PALO ALTO CALIF	1620
JOHNSON	R	RD M WEATHERLY CO		ATALANTA GA	360
JOLLEY	S	MIBM		SAN JOSE CALIF	1800
JONAS	C	RDOW CHEMICAL CO	3417	MIDLAND MICH	1800
JONES	C	ETENN A+I STATE UNIV		NASHVILLE TENN	1620
JULIAN	F	BCOUNT OF SAN DIEGO		SAN DIEGO CALIF	
KATZ	E	LA WATER + POWER		DOWNEY CALIF	360
KENDRICK	N	BNECHES BUTANE PROD		PORT NECHES TEXAS	1800
KENNY	A	DIBM		NEW YORK CITY N.Y.	1800
KIEL	D	FMICHIGAN STATE UNIV	3625	EAST LANSING MICH	1130
KING	J	LARK. POWER + LIGHT	1457	PINE BLUFF ARK	1130
KIRKHAM	G	KIBM		SAN JOSE CALIF	1800
KLEES	G	NAUTONETICS		FULLERTON CALIF	1800
KNOELL	D	LBASIC VEGETABLE PROD	5406	VACAVILLE CALIF	1130
KOCH	W	JIBM		SAN FRANCISCO CALIF	1130
KOSTER	D	EMARATHON OIL CO	5317	TEXAS CITY TEXAS	1130
KRIEGER	J	MIBM		POUGHKEEPSIE N.Y.	360
KRIEGE	K	BCAL POLY COLLEGE		POMONA CALIF	1130
KUREK	N	CANOGA ELECTRONICS		CANOGA PARK CALIF	1130
LACOURSIERE	J	EWESTERN COWTR CORP		SIOUX CITY IOWA	360
LAHNERS	F	LVA HOSPITAL	3055	OMAHA NEBRASKA	1620
LAMAR	J	RIBM		CHICAGO ILL	1800
LANDECK	B	WIBM		LOS ALTOS CALIF	1800
LANDWEHR	M	FIBM		SAN JOSE CALIF	1800
LANE	S	EOKLAHOMA UNIVERSITY		NORMAN OKLA	360
LANE	W	GCHICO STATE COLLEGE		CHICO CALIF	1620
LARUE	R	HUNIV OF SO DAKOTA		VERMILLION SO DAKOTA	360
LAWRENCE, JR	W	WIBM		POUGHKEEPSIE NY	1800
LAYNG	W	LSUNDSTRAND AVIATION		ROCKFORD ILL	1130
LEE	E	SUNIV OF TORONTO	701	TORONTO ONT CANADA	1620
LEE	J	AUNIV OF MASSACHUSETTS	1107	AMHERST MASS	1620
LEHR	R	VPANHANDLE EASTERN		KANSAS CITY MO	
LENT	R	SIBM		ELMIRA N.Y.	1800
LESTER	G	IBM		SAN JOSE CALIF	1130
LEVY	L	NMOBIL OIL CO		BELLINGHAM WASH	1800
LEWIS	N	JIBM		POUGHKEEPSIE N.Y.	
LEWIS	E	SYLVANIA		BUFFALO NY	360
LINICK	E	FSOFTWARE RESOURCES		LOS ANGELES CALIF	360
LIPSON	A	LVIRGINIA ELECT + PWR	1044	RICHMOND VA	360
LITTMAN	I	BOISE CASCADE CORP		BOISE IDAHO	1130
LOMAS	W	AIBM		SAN JOSE CALIF	
LONERGAN	P	IBM		WHITE PLAINS N.Y.	

LOUIS	J YLONG ISLAND LTG CO	1120	HICKSVILLE NY	360
LOVE	M AIBM		CLEVELAND OHIO	1130
LUBY	W KCORN PRODUCTS CO ARGO		DOWNERS GROVE ILL	1800
LUKINS	J CUNIV OF KENTUCKY	1553	LEXINGTON KY	1800
LYCHE	D WSOCAL		EL SEGUNDO CALIF	1800
LYNCH	S AUS REDUCTION CO		EAST CHICAGO IND	1130
MACGREGOR	G TLINFIELD COLLEGE	5407	MC MINNVILLE ORE	1620
MALLOY	M CHAMBERLAYNE JR CLGE		BOSTON MASS	360
MANION	R EIBM		DETROIT MICHIGAN	1130
MARKS	M IBM CORP		WHITE PLAINS N.Y.	
MARMIE	F DBENDIX CORP	5422	NORTH HOLLYWOOD CALIF	1800
MASKIELL	F MMCGRAW-EDISON	3081	CANONSBURG PA	360
MASTER	J ESINCLAIR PETROCHEM		LAPORTE TEXAS	1800
MATTHEISS	P KSUN OIL CO		MARCUS HOOK PA	1800
MAY	R WIBM		SAN JOSE CALIF	1130
MAY	R SSUNDSTRAND AVIATION		DENVER COLO	360
MAYER	D IBM		YORKTOWN HTS N.Y.	
MC GUIRE	S WKEN R WHITE CO		DENVER COLO	1130
MC ILVAIN	D RAIR PRODUCTS + CHEM		ALLENTOWN PA	360
MC KINNON	G JIBM		ENDICOTT N.Y.	360
MC LAUGHLIN	E EPUBLIC SERVICE CO		DENVER COLO	1800
MC PAHIL	R BGENERAL DYNAMICS		GROTON CONN	360
MC QUARRIE	C AB C GOVT		VICTORIA BC CANADA	360
MCCALL	E HIBM CO	1389	ST PAUL MINN	360
MEACHERN	N VDELEUW CATHER/CANADA	7068	DON MILLS ONT CANADA	1130
MCINTYRE, JR	N CCHICO STATE COLLEGE		CHICO CALIF	360
MEHL	J WIBM		LOS GATOS CALIF	1130
MEIDL	R AJOS SCHLITZ BREWING	3020	MILWAUKEE WISC	1130
MICHALOWSKI	E WSYLVANIA		TONAWANDA N.Y.	1620
MISFELDT	D DUNITED OF OMAHA		OMAHA NEBRASKA	360
MOORE	W TUSAAVLABS		FORT EUSTIS VA	1620
NECESSARY	J RAVCO CORP ORDNANCE		RICHMOND IND	360
NELSON	J LIBM		PALO ALTO CALIF	360
NEMETH	L PHILADELPHIA WATER		PHOENIXVILLE PA	1130
NORTON	W ASOUTHERN SERVICES	1125	BIRMINGHAM ALABAMA	360
NOWIKOWSKI	R CHECK-UP INC		SOUTHFIELD MICH	1800
O'DESKY	R IIBM		LEXINGTON KY	1800
OLDE	G LUNIV OF KENTUCKY	1553	LEXINGTON KY	1130
OLDEN	L RGEN DYN CONVAIR	5276	SAN DIEGO CALIF	1800
OLSON	R WIBM		CHICAGO ILL	360
ORLOFF	M JLEAR SIEGLER INC		GRAND RAPIDS MICH	360
ORTBALS	M HWESTERN FARMERS ELEC		ANADARKO OKLA	1130
OSHEL	W WNAVAL WEAPONS CENTER		CHINA LAKE CALIF	1800
PAGE	H LIBM		ENDICOTT N.Y.	
PAGE	R WN.Y. STATE ELECTRIC		BINGHAMTON N.Y.	1800
PARSONS	R IBM		LOS GATOS CALIF	1800
PAULIN	R HUNIVERSITY OF ALBERTA		EDMONTON ALTA CANADA	1800
PEASE	W AWEST VIRGINIA P + P		CHARLESTON HTS SC	1130
PEREZ	V US COAST-GEODETIC SUR		SAN FRANCISCO CALIF	360
PENAAR	L VFISH RES BD CANADA	7072	NANAIMO BC CANADA	1130

PINFIELD	E	RUNIV OF COLO-MED CENT		WESTMINSTER COLO	1800
PODOLSKY	J	LFAIRCHILD SEMICONDCT		MT VIEW CALIF	
POLLE	A	IBM FRANCE		PARIS FRANCE	1800
POLYZOIDES	G	DHOOKER CHEMICAL CORP		NIAGARA FALLS N.Y.	1130
PORZAK	J	PLAFAYETTE CLINIC		DETROIT MICH.	1800
POWELL	J	EUNIV OF SO DAKOTA	3318	VERMILLION SO DAKOTA	360
PRATT	R	LDATA CORP	3121	DAYTON OHIO	360
RAFAELIAN	L	ACONTINENTAL AVI ENG		DETROIT MICH	360
RAGSDALE	A	WGENERAL FOODS LTD	7065	TORONTO ONT CANADA	360
REAMS	H	BWESTERN CONTR CORP		SIOUX CITY IOWA	360
REESE	C	HCHEVRON RESEARCH CO		RICHMOND CALIF	1800
RESENSTEIN	R	DCRYSTALLOGRAPHY LAB	1117	PITTSBURGH PA	1130
ROBERTS	B	JPANHANDLE EASTERN		KANSAS CITY MO	1800
RODANTE	F	CTRAVELERS RESEARCH	0184	HARTFORD CONN	360
ROSS	R	DUNIV OF MISSISSIPPI		UNIVERSITY MISSISSIPPI	1620
ROSS	T	AUNIV OF MISSISSIPPI		UNIVERSITY MISSISSIPPI	360
RYAN	W	JCHEVRON RESEARCH CO		RICHMOND CALIF	1800
SAADAT	M	HPIONEER SERV + ENG CO		CHICAGO ILL	360
SAMUELS	L	BIBM		NEW YORK CITY N.Y.	
SANDEFUR	G	GSCIENCE ENGNR ASSOC		SAN MARINO CALIF	1130
SAUNDERS	A	FTRAVELERS RESEARCH	0184	HARTFORD CONN	360
SAUTER	J	LIBM		SAN JOSE CALIF	360
SCHADE	D	EIBM		SAN JOSE CALIF	1800
SCHIFTNER	S	KCLARKSON COLLEGE		POTSDAM NY	360
SCHODITSCH	G	EMONSANTO	3432	ST LOUIS MO	1130
SCHOROW	M	CEN FOR RES MED U.I.		EVANSTON ILL	1130
SCHRADER	H	LCSCH-HAYWARD		HAYWARD CALIF	1800
SCUDDER	J	EASTMAN KODAK	1035	ROCHESTER NY	1130
SCULLY	P	EARTHUR G MC KEE		SAN FRANCISCO CALIF	1130
SEARS	F	DAERO COMMANDER		NORMAN OKLA	1800
SEROUSSI	S	FIBM-RESEARCH CTR		YORKTOWN HTS N.Y.	1800
SHAW	J	FIBM		SAN JOSE CALIF	1800
SHOEMAKER	U	S NAVAL CIV ENG LAB		PORT HUENEME CALIF	1620
SIMS	B	GG C BESTOR + ASSOC		CARMEL CALIF	1130
SLOBODA	J	FLINT JUNIOR COLLEGE		FLINT MICH	1620
SMART	R	NHARZA ENG CO	3396	CHICAGO ILL.	1130
SMITH	R	EGULF STATES UTILITY	1601	BEAUMONT TEXAS	360
SMITH	J	HTRINITY UNIVERSITY		SAN ANTONIO TEXAS	360
SMITH	J	AIBM		POUGHKEEPSIE N.Y.	1130
SMITH	R	LIBM		MIDLAND MICH	1800
SMITH	G	DUNIF OF MISSISSIPPI		WATER VALLEY MISS	360
SNAVELY	C	JOPTICAL COATING LAB	5049	SANTA ROSA CALIF	1130
SOUDERS	R	CIBM CORP		CENTURY CITY CALIF	360
SPENCER	E	HESSO RESEARCH LABS	1132	BATON ROUGE LA	1800
STAFFORD	H	CCOMMONWEALTH ASSOC		JACKSON MICHIGAN	1130
STANSBURY	J	CHALCON INTERNATIONAL	1178	NEW YORK NY	360
STARR	A	TSOUTHERN PACIFIC CO		HOUSTON TEXAS	1800
STAUTNER	J	FIBM		WHITE PLAINS N.Y.	360
STEEL, JR.	T	BSDC		SANTA MONICA CALIF	360
STEELE	L	LIBM CORP		POUGHKEEPSIE N.Y.	360

STEINBERG	D	SDOW CHEMICAL	PITTSBURG CALIF	1130
STEIN	T	WHALCON INTERNATIONAL	NEW YORK CITY N.Y.	360
STEINHART	R	FIBM	WHITE PLAINS N.Y.	360
STEPHENSON	J	WWESTERN FARMERS ELECT	ANADARKO OKLA	1130
STEWART	W	BUNIV OF KENTUCKY	LEXINGTON KY	1800
STOUT	J	EDOW CHEMICAL CO	MIDLAND MICH	1130
STOUT	F	BAGRICO CHEMICAL CO	MEMPHIS TENN	360
STREETER	T	DROCK ISLAND ARSENAL	ROCK ISLAND ILL	1620
SWANSON	T	JU S REDUCTION	EAST CHICAGO IND	1130
SWARD	R	WPALO ALTO UNFD SCH D 5210	PALO ALTO CALIF	1620
TAGHON	M	EIBM	MOUNTAIN VIEW CALIF	1130
TALKINGTON	J	EILLINOIS STATE UNIV	NORMAL ILL	1130
TAPP	R	EBUTCHER + SHERBERD	PHILADELPHIA PA	360
TARBUTTON	R	RIBM	SAN JOSE CALIF	1800
TENNISON	R	DIBM CORP	WHITE PLAINS N.Y.	3644
THOMAS, JR	A	TRINITY UNIVERSITY	SAN ANTONIO TEXAS	360
THOMSON	J	AFISH RES BD CANADA 7072	NANAIMO BC CANADA	1130
TOBIAS	R	LDOW CHEMICAL	PITTSBURG CALIF	360
TURNER	N	PCAMERON IRON WORKS	HOUSTON TEXAS	360
TUTZER	A	CIBM	CHICAGO ILL	360
UCZEN	E	JANACONDA WIRE + CABLE	SYCAMORE ILL	1130
VAN NESS	V	VIBM CORP 1392	ARMONK N.Y.	1130
VAUGHAN	N	CIBM	LOS ANGELES CALIF	
WAGNER	L	FPHYSICS INTERNATIONAL	SAN LEANDRO CALIF	360
WALKER	R	PWHIRLPOOL CORP 3381	ST JOSEPH MICH	1800
WALKUP	G	WIBM	WHITE PLAINS N.Y.	
WALL	H	MIBM CORP	SILVER SPRING MD	1130
WALTZ	A	JSINCLAIR-KOPPERS	PASADENA TEXAS	1800
WANTA	J	AIBM	DEARBORN MICH	1130
WARREN	H	AIBM	ENDICOTT N.Y.	
WATSON	T	STANFORD ELEC LABS 5123	STANFORD CALIF	1620
WAY	C	EPRATT + WHITNEY AIRC	WEST PALM BEACH FLA	1130
WEATHERLY	B	CDM WEATHERLY CO	ATLANTA GA	360
WEAVER	S	EIBM	SAN JOSE CALIF	1800
WEGENER	J	DCHRISTIAN BROS COLL 3097	MEMPHIS TENN	1130
WEISS	R	IBM	ENDICOTT N.Y.	
WERNICKE	G	FELS RESEARCH INST	YELLOW SPRING OHIO	360
WHEELER	F	MBELOIT COLLEGE	BELOIT WISC	1800
WHEELER	J	WWESTERN ELECTRIC CO	GREENSBORO NC	1800
WHETZEL	B	WLOX EQUIPMENT CO	LIVERMORE CALIF	1130
WHITE	C	RIBM	VESTAL N.Y.	360
WHITE	G	WUNIV OF CALGARY	CALGARY ALTA CANADA	1130
WHITLEY	G	LUNIV OF OKALAHOMA	NORMAN OKLA	1130
WIGDAHL	A	BALLEN BRADLEY CO 3134	MILWAUKEE WISC	360
WILD	G	LHAWAIIAN ELECTRIC CO	KAILUA HAWAII	1800
WILEY	J	KFLORIDA POWER CORP	ST PETERSBURG FLA	360
WILSON	G	WCOMMONWEALTH ASSOC	JACKSON MICH	1130
WILSON	J	EROCK ISLAND ARSENAL	ROCK ISLAND ILL	1620
WINDHAM	C	FUNIF OF MISSISSIPPI	OXFORD MISSISSIPPI	360
WISE	L	EIBM	SAN JOSE CALIF	1800

WOLF	G	LIBM	SAN JOSE CALIF	1[illegible]0
WOOD	G	KLTV ELECTROSYSTEMS	DALLAS TEXAS	1800
WOOD	P	CLOWELL TECH INST	LOWELL MASS	1620
WOOLDRIDGE	C	AG C BESTOR + ASSOC	CARMEL CALIF	1130
WRIGHT, JR.	O	GPIONEER NATURAL GAS 5086	AMARILLO TEXAS	1620
WRIGHT	J	RTRANE CO 3320	LA CROSSE WISC	1130
YARBROUGH	L	DARCON CORP	WAKEFIELD MASS	1130
YARBROUGH	J	ACRANE CO 3406	CHICAGO ILL	1130

PL/1 AND FORTRAN: A COMPARISON

PART I STATEMENT SIMILARITY

PART II A SAMPLE PROGRAM

PART III CAPABILITIES OF PL/1 BEYOND FORTRAN

PL/1 AND FORTRAN: A COMPARISON

PART 1 - STATEMENT SIMILARITY

DATA DEFINITION

FORTRAN

```
DIMENSION A(50,50), B(25,100), C(2)
COMMON A
EQUIVALENCE (A, B)
DATA C/2*1.0/
INTEGER*2     A
REAL*8        B
COMPLEX       D
LOGICAL       E
```

PL/1

```
DECLARE A(50,50) BINARY (15,0) EXTERNAL,
        B(25,100) FLOAT BINARY (53) DEFINED A,
        C(2) INITIAL (1.0),
        D FLOAT BINARY COMPLEX,
        E BIT(1) PACKED;
```

ASSIGNMENT

FORTRAN

```
A = B+C*SQRT(E)
```

PL/1

```
A = B+C*SQRT(E);
```

CONTROL STATEMENTS

FORTRAN

```
      GO TO 25
      GO TO (1,3,5), N
      ASSIGN 3 TO N
      IF (X .EQ. Y .AND. Z .GT. Y)  A=B+13
      DO 100    I=1,15,3
100 CONTINUE
      PAUSE 'END PHASE 1'
```

PL/1

```
      GO TO NEXT;
      GO TO L (I);
      I=3;
      IF X=Y & Z>Y   THEN A=B+13;
D100:DO        I=1 TO 15 BY 3;
      END D100;
      DISPLAY ('END PHASE 1');
```

INPUT OUTPUT

FORTRAN

```
READ (5,1)  X,Y,Z
FORMAT  (F8 .2,2F4 1)
WRITE (6,2)
2 FORMAT (1HI,'HEADINGS')
WRITE (8)  X,Y,Z
ENDFILE 8
REWIND 8
```

PL/1

```
GET EDIT X,Y,Z (F 8,2), F  5.1));
PUT LIST ('HEADING')PAGE;
WRITE FILE (SCRATCH) FROM (WORK)
     DCL  1  WORK  2X,2Y,2Z;
CLOSE FILE (SCRATCH);
```

SUBPROGRAM CONSTRUCTION

FORTRAN

```
CALL MATMPY (A, B, C)
FUNCTION SPEC (A, B)
SUBROUTINE XTR(4,Z)
ENTRY   XTRA(Q, R)
EXTERNAL   S1, S2, S3
RETURN (A+B+C)
```

PL/1

```
CALL MATMPY (A, B, C);
SPEC:  PROCEDURE (A, B);
XTR:  PROCEDURE (Y, Z);
XTRA:  ENTRY (Q, R);
DECLARE   (S1, S2, S3) ENTRY;
RETURN (A+B+C);
```

SAMPLE PROBLEM

QUADRATIC MODEL $A_0 + A_1 T + 1/2\ A_2 T^2$

EXPONENTIALLY SMOOTHED COEFFICIENTS

READ IN DATA

UPDATE MODEL AND MAKE NEW FORCAST

PRINT NEW FORCAST

PUNCH UPDATED MODEL

PART II - COMPARISON OF SAMPLE PROGRAMS

FORTRAN MAIN PROGRAM

```
      DIMENSION ID(5)
      INTEGER*2  ID
      COMMON  OBS, PRJ, A0, A1, A2
      WRITE (6,100)
100   FORMAT (1H1)
      1 READ (5,101) OBS, PRJ, A0, A1, A2, (ID(I), I=1,15)
101   FORMAT (F8.4, 4F10.4,15A2)
      CALL FORCAST
      WRITE (6,102) PRJ, (ID(I), I=1,15)
102   FORMAT (1H  , F8.4, X1, 15A2)
      WRITE (7,103) PRJ, A0, A1, A2, (ID(I), I=1,15)
103   FORMAT (X8, 4 F10.4, 15A2)
      GO TO 1
      END
```

PL/1 MAIN PROGRAM

```
SAMPL: PROCEDURE OPTIONS(MAIN);
DECLARE   (OBS,PRJ,A0,A1,A2,ALPHA,BETA)
   FLOAT BINARY, ID CHARACTER(30) ) STATIC
   EXTERNAL;
ALPHA=.1;  BETA=.9;
PUT EDIT PAGE;
START: GET EDIT (OBS,PRJ,A0,A1,A2,ID)
      (F(8,4), 4(X(2), F(8.4)), X(2), A(30));
CALL FORCAST;
PUT EDIT(PRJ,ID) (F(8,4),X(2), A(30)) SKIP(1);
PUT FILE (PUNCH) EDIT
   (PRJ,A0,A1,A2,ID)
   (X(8), 4 ( X(2),F(8,4)),X(2),A(30));
GO TO START;
END;
```

FORTRAN SUBPROGRAM

```
SUBROUTINE FORCAST
COMMON OBS,PRJ,A0,A1,A2
REAL*4 ALPHA/.1/,BETA/.9/
ERR=PRJ-OBS
TEMP=A2-ALPHA**3*ERR
A1=A1+A2-1.5*ALPHA**2*(2.0-ALPHA)*ERR
A0=OBS+BETA**3*ERR
A2=TEMP
PRJ=A0+A1+.5*A2
RETURN
END
```

PL/1 SUBPROGRAM

```
FORCAST: PROCEDURE;
ERR = PRJ-OBS;
TEMP= A2- ALPHA**3*ERR;
A1 = A1+A2-1.5*ALPHA**2*(2- ALPHA)* ERR;
A0 = OBS+BETA**3*ERR;
A2 = TEMP;
PRJ = A0+A1+.5*A2;
RETURN;
END;
```

PL/1 HAS MORE CONCISE EXPRESSION

PROBLEM INITIALIZE ARRAY A AND COMPUTE

A_{IJ2} B_{IJ} $+M_{IMS_6J}$ FOR ALL I.J

FORTRAN

```
      DIMENSION  A (100,100,100)
      DO 5       I=1,100
      DO 5       J=1,100
      DO 5       K=1,100
5     A(I, J,K)=0.0
      DO 10  I=1 100
      DO 10  J=1 100
10    A (I,J,2)=B (I,J)+ M (I,MS(6),J)
```

PL/1

```
DCL A (100,100,100); A=0;
A(*,*,2)= B+M(*,MS(6),*);
```

PL/1 HAS BETTER INTERRUPT CONTROL

PROBLEM ALLOW FLOATING POINT UNDERFLOW FIRST 100 TIMES THEN KILL JOB.

FORTRAN REQUIRES ASSEMBLY LANGUAGE ROUTINE.

PL/1

```
ON UNDERFLOW BEGIN;
DCL COUNT FIXED(3) INIT(0);
IF COUNT = 99 THEN DO;
PUT LIST('100 UNDERFLOWS') SKIP(1);
SIGNAL FINISH; END;
COUNT=COUNT+1;
RETURN; END;
```

PL/1 HAS MORE EXTENSIVE DATA EDITING

SOURCE	TARGET
00100	**100
10203	1 2 3
1234.56	1.234.56
12	1 2
001.23	$1.23
-123	$1.23CR
123	123+
-123	123-

PL/1 HAS SUPERIOR ARRAY HANDLING

EXAMPLES

```
DECLARE A(-5:-25,17:18);
DO I=-5 TO -25 BY -5,
      -25 TO -30 BY -1,
      WHILE ( X=19.6);
A=B+C/E;
      WHERE A,B,C,AND E ARE
      ALL N-DIMENSIONAL ARRAYS
```

PL/1 HAS MORE BUILT IN FUNCTIONS

EXAMPLES

DATE	RETURNS CHARACTER STRING YYMMDD
TIME	RETURNS CHARACTER STRING HHMMSSTTT
SUM(X)	RETURNS SUM OF ALL ELEMENTS OF X
PROD(X)	RETURNS PRODUCT OF ALL ELEMENTS OF X

PL/1 HAS CHARACTER STRING AND BIT STRING PROCESSING

EXAMPLES

```
DCL SYMPTOMS BIT(64) PACKED,
HEADACHE BIT(1) DEFINED SYMPTOMS
POSITION(25),
FEVER BIT(1) DEFINED SYMPTOMS
POSITION(35);

IF HEADACHE¬FEVER THEN GO TO ASPIRIN;
ELSE GO TO PENECILLIN;
```

BIT(64) REQUIRES 8 BYTES OF STORAGE

```
X='XYCOMABCMON';
Y= INDEX(X,'COM');
Z= INDEX(X,'MON');
A= SUBSTR(X,Y,3)|| SUBSTR(X,Z,3);
```

A WILL BE SET TO COMMON

IN ADDITION PL/1 OFFERS

COMPILE TIME CAPABILITY (MACROS)

LIST PROCESSING

MULTI-TASKING

DYNAMIC ALLOCATION AND RELEASE OF STORAGE

EXTENSIVE DEBUGGING

EXTENSIVE I/O CAPABILITY

hooker

M.2.2 (a)

DYNAMIC MODEL SIMULATOR FOR THE IBM 1130

by

George D. Polyzoides
Hooker Chemical Corporation, Niagara Falls, New York

The Dynamic Model Simulator is a chain of FORTRAN source programs that permits the IBM 1130 user to investigate in depth the dynamic behavior of physical systems that can be modeled into linear or nonlinear ordinary differential equations.

It can be used for the investigation of a wide range of engineering and mathematical problems from reaction kinetics and reactor responses to the design of electrical networks and structural assemblies.

The system does not require familiarity with analog computers although in a way it forces the IBM 1130 hardware to perform the function of an analog computer while at the same time it offers the digital computer advantages of random memory access and ten digit (extended) precision.

The input consists of English command words (START, WAIT, RESET, etc.) and numeric data which are read in free format.

The system output includes tabulated values of the variables at specified intervals, a table of the variables involved and detailed error and information messages. Plotting of the variables is also available as an option.

System Source Language: FORTRAN IV, Level E

System Hardware Requirements:
1131 CPU-8K, one 2315 disk cartridge
1442 Card Reader
1132 Printer
Benson-Lehner 305 Digital Incremental Plotter (optional)

DYNAMIC MODEL SIMULATOR

FOR

THE IBM 1130 SYSTEM

George D. Polyzoides
Hooker Chemical Corporation
December 1967

hooker

NIAGARA FALLS, NEW YORK 14302, PHONE (716) 285-6655

ABSTRACT

The Dynamic Model Simulator (DYNAMO) consists of a series of FORTRAN IV programs that allow the IBM 1130 user to investigate with detail and accuracy the behavior of physical systems that can be modeled into linear or nonlinear ordinary differential equations.

The DYNAMO Simulator can be of great help to engineers and scientists who consider such time consuming problems as the investigation of the transient behavior of chemical reaction systems, electrical networks, process variables and control systems.

The input to the DYNAMO Simulator is in free format and in the form of distinct key words and numbers. The digital computer setup prohibits real time operation but it offers the additional advantages of accuracy, data storage and detailed plotting. Besides the input-output and calculation mainlines DYNAMO utilizes the 1130 Plotting System, a plotting software package created by Hooker's Systems Engineering Group.

The minimum equipment requirements for program execution are:

1131-CPU-8K-2B

1442 Card Reader and 1132 Printer

The Plots (optional) can be obtained through an on line digital plotter (.005"/step).

hooker

TABLE OF CONTENTS

	Page
INTRODUCTION	3-4
PART ONE GENERAL INFORMATION	
A) PROGRAM CHARACTERISTICS	5-7
B) PROGRAMMING TECHNIQUES	8-13
PART TWO EXAMPLES OF APPLICATIONS	
CASE ONE A CONTINUOUS STIRRED REACTOR BATTERY	14 a-e
CASE TWO DESIGN OF AN RLC CIRCUIT	15 a-p
APPENDIX A DYNAMO INPUT LANGUAGE	17
APPENDIX B PLOTTING INSTRUCTIONS	21 a-b
APPENDIX C FORTRAN-ALGEBRAIC EQUIVALENCE OF THE MODEL	22-24
APPENDIX D CONTENTS OF THE STANDARDS FILE	24 -

INTRODUCTION

The expansion of the range of the applications of digital computers into fields that were the traditional domain of analog computers is a rather recent trend that has produced such interesting results as the GPSS, the CONSIM, the CSMP, the PACTOLUS and other simulation systems. The DYNAMO Simulator does not claim "a place in the sun" among these systems. Its mode of operations is more restricted and less sophisticated. The object of DYNAMO is the solution of ordinary differential equations and the emphasis is on the mathematics rather than the block diagrams. Since differential equations and block diagrams are frequently equivalent in describing a system, it follows that in many instances DYNAMO can be used in obtaining results similar to the ones obtained from larger and more complicated systems.

The point of decision when programming for analog to digital equivalences is whether analog procedures and nomenclatures should be carried over to the digital system's specifications. For example, should gain be labeled as such, or should it be implied by a multiplication? One can argue about the advantages and the nuisances of both types of approach.

The DYNAMO Simulator approaches the problem from the point of view of the person who has more experience in digital than in analog computation. This implies that analog computer nomenclature, wiring

diagrams, and scaling are not necessary to describe and set up the problem. The analog concepts, although still present and helpful, tend to fall in the background while familiarity with digital concepts becomes essential.

PART ONE

PROGRAM CHARACTERISTICS

A) INPUT REQUIREMENTS

The input to the DYNAMO Simulator consists of some key words and numeric data. The programs have been written in such a way as to permit experimentation with the differential equations that describe the model. The execution of the programs can be interrupted at five distinct points and restarted at some later time without loss of continuity.

1. Input Language

The key input words or commands constitute a very elementary input language. The term "language" is applicable only as far as it is understood as the substitution of a numeric code with English words. The input commands can be divided into six categories:

i) Integration Information commands: They define the step length and interval of the integration, value of the equation coefficients, etc.) CONTROL, COEF

ii) Identification commands: TITLE, TABLE

iii) Initial conditions commands: START, RESET, REPEAT, RESTART

iv) Utility commands: SETUP, LOG, NOLOG

v) End Indicator commands: WAIT, END, HALT

vi) Output commands: PLOT (see Appendix B), LIST, FILE

For those interested in more details the DYNAMO input language is presented in more length in Appendix A.

2. Input Procedures

The DYNAMO commands and the required numeric data can be entered through punched cards or the 1131 console. The two input modes can be alternated during program execution. All commands are listed on the 1131 console, but listing can be stopped upon use of the NOLOG command. The SETUP command can be used to reset the system standard files (see Appendix D).

All commands and data are entered on a free format basis. The special plotting instructions are only sequence dependent. More will be said about free format in subsequent sections.

Upon detection of F- type input errors in the numeric data or an erroneous command the user has the option of causing a "PAUSE" and correcting the error. More serious logic errors (i.e. no initial conditions specification) cause exits to the monitor. The files containing the standards for the system are not closed and most of the times the error can be corrected and the run can be restarted without loss of continuity.

3. The Mathematical Model

All the first order ordinary differential equations describing the model must be previously stored as a function subprogram. The present dimensioning allocations allow a maximum of twenty variables (equations).

These ordinary differential equations may be linear or nonlinear. Second or higher order equations can be expressed as two or more equivalent first order equations. The coefficients of the equation terms can be

variable and their values must be present during the input step.

The DYNAMO Simulator will accept any first order differential which can be arranged in the form:*

$$\frac{d}{dx}(Y_i) = f(y_i, y_{i-1}, \cdots y_1, x) \qquad \text{-1-}$$

where Yi is the i^{th} dependent variable and x is the independent variable.

EXAMPLE:

Algebraic expression: $\frac{dY_2}{dx} + .03Y2 = SINX + .5Y1$

DYNAMO FORTRAN: `F = SIN(X) - .03*Y(2,L) + .5*Y(1,L)`

The value of L is determined in the calling mainline. The function subprogram must contain a computed GO TO statement so that the appropriate derivative value for each variable is selected. Appendix C contains sample subprograms in both algebraic and FORTRAN notations.

* Other equation forms can be expressed in the form of equation -1- after some mathematical manipulation.

B) PROGRAMMING TECHNIQUES

The present version of the program consists of two input mainlines, an error checking mainline, the mainline that solves the differential equations and the mainline that generates the plotting specifications. The actual plotting is accomplished through a separate software package. Due to the number of CALL LINK's that is involved the majority of the programs have been stored in core image to allow for fast loading.

1. System Standards

A seventy (70) digit integer vector is initialized and then continuously updated and maintained through each execution of the DYNAMO programs. Appendix D contains a list of the information contained in the vector. The standards can be reset by using a special "standards generation" program or by using the SETUP command during the program execution.

2. The Predictor-Corrector method for numerical integration.

The solution of the differential equations describing the model is the most vital part of DYNAMO. The Predictor-Corrector method of Hamming(1) has been found to fulfill the important considerations of stability, accuracy and relative calculation speed.

No attempt will be made here to discuss the method in detail. Any such effort would be a reproduction of the excellent article by Ralston(2) describing the details. The limited information to be presented

in this paper is here just for clarifying the overall approach.

Given the values of M dependent variables (Y) and an independent variable (X) at four equispaced points in the X domain (0, 1, 2, 3) the predictor corrector method (fifth order accuracy) is a means for obtaining the M values of (Y) at point (4). This procedure can be generalized by setting the four known points of the domain as n-3, n-2, n-1, and n and by considering the points to be obtained as the x domain value becomes Xntl. A set of four equations is used for each i^{th} variable (i=1,M) and this procedure will repeated until x_{n+1} reaches a present value:

i) The predictor equation gives a rough estimate of Y(i,n+1)

$$P_{n+1} = Y(i,n-3) + (4h/3) * (2y'(i,n) - y'(i,n-1) + 2y'(i,n-2)) \quad -2-$$

ii) The modifier and corrector equations eliminate the need for iterative convergence:

$$m_{n+1} = P_{n+1} + (112/121) * (p_n - c_n) \quad \text{and} \quad m'_{n+1} = f(x_{n+1}, m_{n+1}) \quad -3-$$

$$c_{n+1} = \frac{1}{8}(9Y(i,n) - Y(i,n-2) - 3h(m'_{n+1} + 2y'(i,n) + y'(i,n-1))) \quad -4-$$

where $(p_n - c_n)$ is the truncation error from the last step, y' is the derivative of y and h is the interval between $x_n, x_{n-1}, x_{n-2}, x_{n-3}$ and x_{n+1}

III) The final equation takes the form:

$$Y(i,n+1) = c_{n+1} + \frac{9}{121}(p_{n+1} - c_{n+1})$$

where $(p_{n+1} - c_{n+1})$ is the present truncation step.

From the point of view of calculations the only additional requirement is that the values for Y(i,K), where (K=2,4) must be calculated from the initial conditions Y(i,1) before equations (2-5) can be executed. This initial step, commonly called the STARTER consists of a first order Newtonian extrapolation and an iterative convergence technique described in (2).

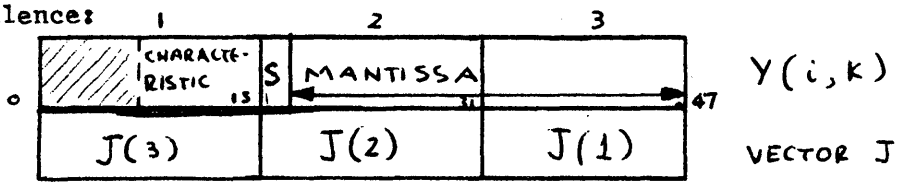
Stability checks have been incorporated in the integration routines. The user can check the stability of the calculations by using DATA SWITCH 4 during the execution of the program.

3. Data conversion from extended to standard precision.

In order to minimize the accumulation of the inherent roundoff errors, the DYNAMO mainlines (with the exception of the plot generation mainline) have been compiled in extended precision. The 1130 Plotting System is composed of standard precision mainlines. The DYNAMO data must be converted into standard precision before they are stored in the files from where the plotting mainlines are to pick them up.

Conversion is accomplished by equivalencing the extended precision floating point values of X and Y(i,k) to a three digit vector J and by creating the new integers IY(20,2) and IX(20). The technique can be outlined as follows:

i) Equivalence:

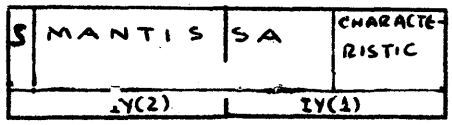


ii) IY(i,2) = J(2)

iii) IY(i,1) = (J(1)/256)*256

iv) IY(i,1) = IY(i,1) + J(3)

v) store IY(i,1) and IY(i,2) ⟶



The 32 bits comprising the last figure will be treated as a floating point variable Y(i,k) when read from the permanent file by a standard precision program.

4. The free format routines

The free format input is accomplished by a group of subroutines that perform the following operations, after a data card has been read in (80A1 Format) and a starting and ending column number has been indicated:

i) Detection of the first nonblank character.

ii) Detection of a predetermined (EBCDIC) character code.

iii) Retrieval of an integer number.

iv) Retrieval of a floating point number.

v) Retrieval of an alphameric text contained between two apostrophi.

As an example of the utilization of these subroutines consider the data card shown below:

```
Column
        10      18.6        T=24
1 2 3 4 5 6 7 8 9 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 . . . . . .
                  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
```

i) set starting column (1) and detect the first nonblank (5).

ii) Find first blank column after (5) (column 7).

iii) Use integer retrieval routine (columns 5-6).

iv) Starting with column 7 find nonblank character (11).

v) Find next blank (15) and retrieve floating point number (11-14).

vi) Search for T and if search is successful (as it will be) retrieve the number following the (=) sign.

5. The Digital Plotting

All plotting must be done on a .005"/step digital plotter. By interpretting the special instructions (Appendix B), the program generates the plotting specifications. These specifications are then stored in a file, the plotter programs are loaded in core, the plotting specifications are read in from

the file and the plots are produced. The DYNAMO user may exercise a variety of options in generating the plot specifications:

i) The right-hand Y axis can be used independently of the left hand Y-axis or it may be omitted completely.

ii) Titles may be specified for the plots and the plot axes.

iii) Two of the following three quantities may be present or all three may be determined by the plotter scaling routines.

a) axis minima

b) axis maxima

c) axis "delta" increment (units/inch)

iv) The data can be plotted as points, least squares lines, or points with fill-in lines.

v) The X-Y variables can be transformed (Y2→Y2/Y1*Y3)

vi) The size of any of the axes to be used may be either 7." or 10." without the frame or 8.5" or 11". with the frame.

vii) An additional axis, same as the parallel to the X axis may be specified so that the axes enclose the graph in a rectangular box.

viii) Undesirable Y variables may be omitted and the x- axis variables for the plot may be any of the X-Y variables.

ix) The X (or Y) axis may be shifted to the zero point of the Y (or X) axis.

x) The data for the plots may be generated from values stored between any two records of the file (max. 50 records/plot).

Plotting of the results from the simulation can be postponed for some other time by INTERRUPTING the program after the data are calculated,

printed and stored. An examination of the listed output may suggest patterns for a more meaningful graphical presentation of the data. The punched output is in a format directly acceptable to the stepwise regression program of the Hooker library. In this way the results from the DYNAMO Simulator can be used to form algebraic equations describing the variables. This can be a great time saving feature when the model under investigation is part of a bigger model.

6. Future Expansions

The present form of DYNAMO can become a more versatile computational tool with the addition of a function generator and a way to specify disturbances directly. (Distrubances can be investigated now in an indirect manner).

From the viewpoint of calculation speed two more steps are to be considered for future improvements:

a) Introduction of a variable step length.

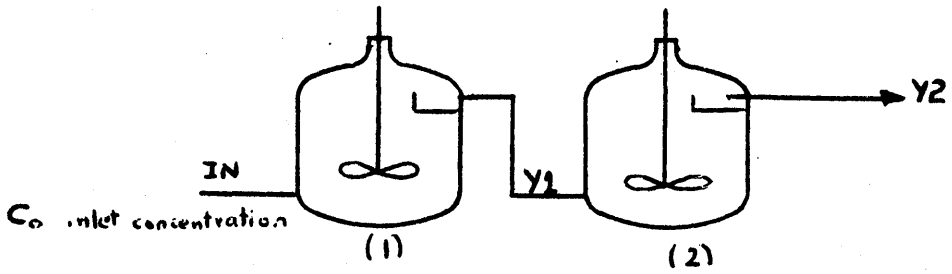
b) Use of an assembly language "patch" to speed up the numerical integration.

PART TWO

EXAMPLES OF APPLICATIONS

CASE No. 1 A CSTR Battery (4)

We will investigate the changes of concentration of a component A when a second order reaction $A \xrightarrow{k} B$ takes place in a two stage CSTR battery. The diagram below shows the flow pattern and the equipment layout:



The mass balance equations around the tanks are:

i) first tank: $\frac{d}{dt}(Y_1) + \frac{Y_1}{\theta_1} + K_1 Y_1^2 = C_o$

ii) second tank: $\frac{d}{dt}(Y_2) + \frac{Y_2}{\theta_2} + K_1 Y_2^2 = Y_1$

where: θ is the retention time

K_1 = the reaction rate constant

Y_1, Y_2 = the concentrations of A in tanks 1 & 2 respectively.

The following pages illustrate a) the use of the DYNAMO language in the input phase b) the standard printed output and c) a sample of plotter output.

```
TITLE
TWO CSTR BATTERY  -SECOND ORDER REACTION
CONTROL  6
1  2
2   2.4
3  .01
4  1
5  10
6  1
START
0.
1.
1.
LIST  10
PLOT  1
NUMBER = 1 TITLE'CSTR BATTERY//TANK 1 SOLID LINE//TANK 2 DOTTED LINE'
X AXIS TYPE = 12 TITLE'ELAPSED TIME IN MINUTES'
VARIABLES
FILE NO. = 1
*
YLEFT AXIS  TYPE = 12  TITLE 'MOLE FRACTION OF COMPONENT A IN TANK'
VARIABLES
FILE NO. 2=  LINE=1000   GRAPH=4222
FILE NO.3=  LINE=2510  GRAPH=4322
*
**
GENERAL OPTIONS
FRAME
STOP
TABLE
1       THE FOLLOWING RESULTS INDICATE THE CHANGES IN THE
2       CONCENTRATION OF A COMPONENT (A) UNDERGOING A SECOND
3       ORDER REACTION (K=.5      ) IN A TWO
4       CSTR BATTERY.
5       FIRST VARIABLE    TIME IN MINUTES
6       SECOND VARIABLE    CONCENTRATION OF A IN TANK 1
7       THIRD VARIABLE     CONCENTRATION OF A IN TANK 2
8       THE INITIAL CONDITIONS ARE
9       TIME 0.  A IN TANK 1 AND 2 100 PERCENT
10      THE RETENTION TIME (THETA) IS ONE MINUTE
11      CONCENTRATION OF INPUT STREAM IS 100 PERCENT

        END OF RUN
```

TWO CSTR BATTERY -SECOND ORDER REACTION

~~DYNAMO~~ INFORMATION TABLE

THE FOLLOWING RESULTS INDICATE THE CHANGES IN THE CONCENTRATION OF A COMPONENT (A) UNDERGOING A SECOND ORDER REACTION (K=.5) IN A TWO CSTR BATTERY.

FIRST VARIABLE TIME IN MINUTES

SECOND VARIABLE CONCENTRATION OF A IN TANK 1

THIRD VARIABLE CONCENTRATION OF A IN TANK 2

THE INITIAL CONDITIONS ARE

TIME 0. A IN TANK 1 AND 2 100 PERCENT

THE RETENTION TIME (THETA) IS ONE MINUTE

CONCENTRATION OF INPUT STREAM IS 100 PERCENT

TWO CSTR BATTERY -SECOND ORDER REACTION

2
14 C

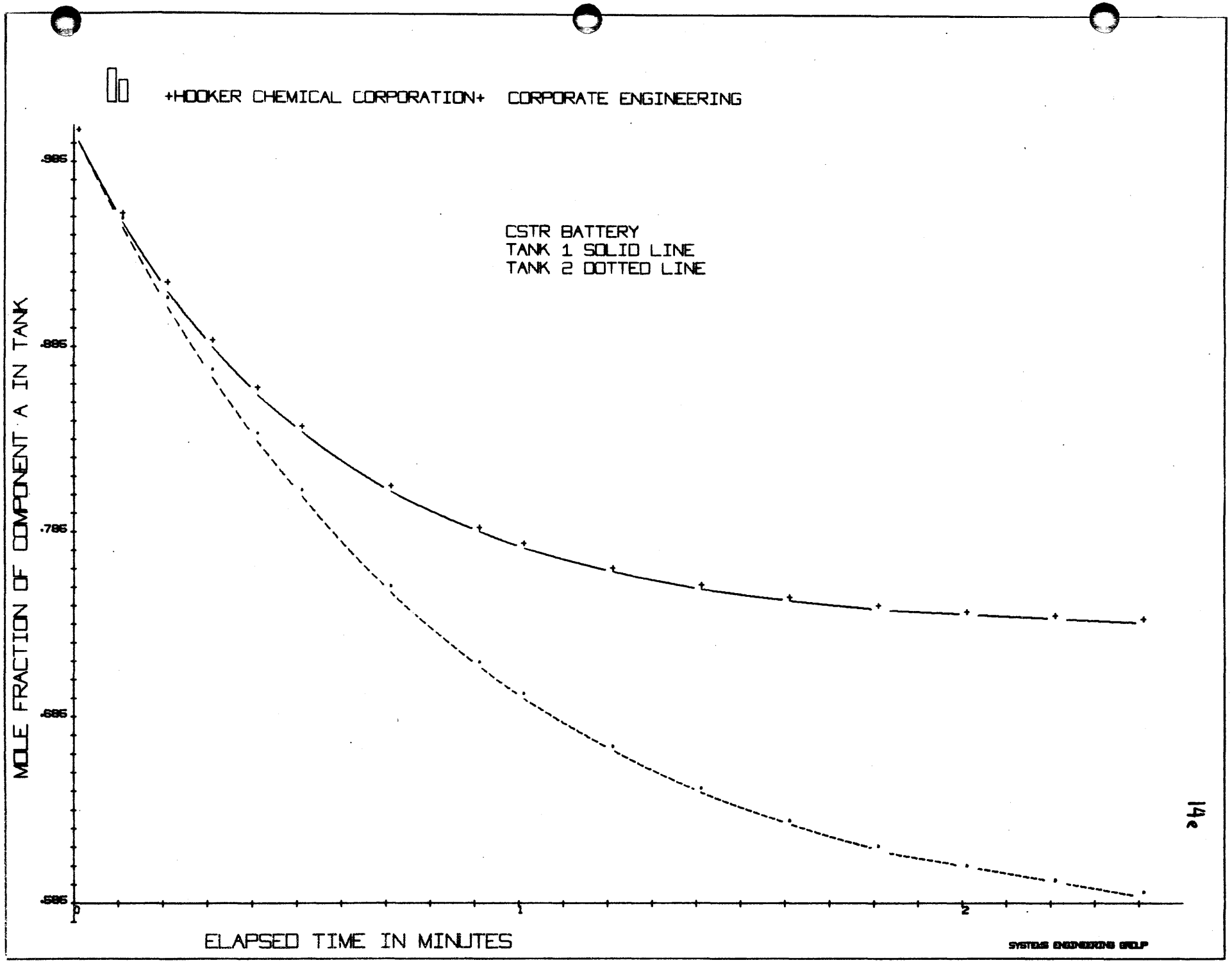
FIRST COLUMN- INDEPENDENT VARIABLE(X). DEPENDENT VARIABLES(Y) FOLLOW SIX PER ROW

0	0.00000E 00	0.10000E 01	0.10000E 01
10	0.10000E 00	0.95464E 00	0.95245E 00
20	0.20000E 00	0.91735E 00	0.90961E 00
30	0.30000E 00	0.88656E 00	0.87114E 00
40	0.40000E 00	0.86108E 00	0.83669E 00
50	0.50000E 00	0.83992E 00	0.80590E 00
60	0.59999E 00	0.82231E 00	0.77843E 00
70	0.69999E 00	0.80765E 00	0.75397E 00
80	0.79999E 00	0.79540E 00	0.73222E 00
90	0.89999E 00	0.78517E 00	0.71291E 00
100	0.99999E 00	0.77661E 00	0.69577E 00
110	0.10999E 01	0.76945E 00	0.68059E 00
120	0.11999E 01	0.76345E 00	0.66716E 00
130	0.12999E 01	0.75842E 00	0.65528E 00
140	0.13999E 01	0.75419E 00	0.64479E 00
150	0.14999E 01	0.75065E 00	0.63554E 00
160	0.15999E 01	0.74768E 00	0.62738E 00

TWO CSTR BATTERY -SECOND ORDER REACTION

FIRST COLUMN- INDEPENDENT VARIABLE(X). DEPENDENT VARIABLES(Y) FOLLOW SIX PER ROW

170	0.16999E 01	0.74518E 00	0.62020E 00
180	0.17999E 01	0.74309E 00	0.61388E 00
190	0.18999E 01	0.74133E 00	0.60832E 00
200	0.19999E 01	0.73985E 00	0.60344E 00
210	0.20999E 01	0.73861E 00	0.59916E 00
220	0.21999E 01	0.73756E 00	0.59541E 00
230	0.22999E 01	0.73668E 00	0.59212E 00
240	0.23999E 01	0.73594E 00	0.58924E 00


+HOOKER CHEMICAL CORPORATION+ CORPORATE ENGINEERING
CSTR BATTERY
TANK 1 SOLID LINE
TANK 2 DOTTED LINE
MOLE FRACTION OF COMPONENT A IN TANK
.985
.885
.785
.685
.585
0
1
2
ELAPSED TIME IN MINUTES
SYSTEMS ENGINEERING GROUP

14e

87

CASE No. 2 Design of an RLC circuit (5)

The investigation of the transient behavior of electrical networks is another potential area of applications. The signle loop RLC circuit pictured below is a textbook example of differential equation solving. The diff. equations describing the transient characteristics of the circuit are:

Current: $$L\frac{d^2(i)}{dt^2} + R\frac{di}{dt} + \frac{1}{C}i = 0$$

Voltage (across the capacitor): $$dV/dt = \frac{1}{C}i$$

where: i = current amp

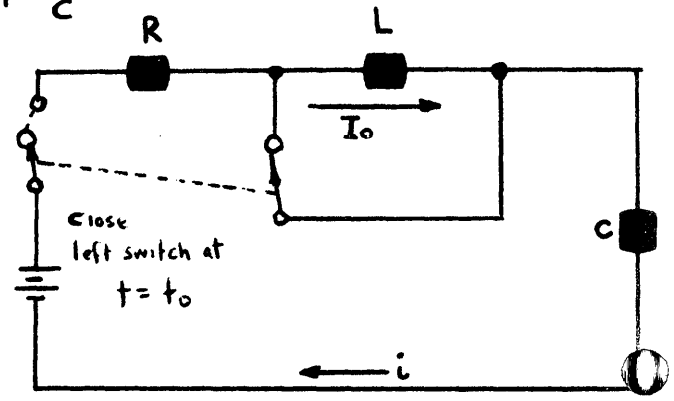
L = inductance henry

C = capacitance farad

R = resistance ohm

E = potential volts

v = voltage "



The behavior of such a circuit will be investigated for three cases: a) overdamping b) critical damping c) underdamping. This will be accomplished by using the COEF command to set appropriate values for R, L, and C. The REPEAT command will cause continuous execution with the original initial conditions. The next pages show the input requirements and the resulting output (the plotting instructions have been omitted for the sake of brevity).

```
          TITLE
          RLC CIRCUIT CALCULATIONS-OVERDAMPED CASE
          TABLE
1         THE INITIAL CONDITIONS FOR THE RUN ARE
1           A)  I(0)=0
1           B)  V(0)=0
1           C) (D(I)/DT).0 = (E-VCO)/L = 4
1           E) THE CIRCUIT POTENTIAL IS 4 VOLTS
1         VARIABLE COEFFICIENTS
1           NUMBER 1      RESISTANCE
1           NUMBER 2      CAPACITANCE
1           NUMBER 3      INDUCTANCE
1
1         OUTPUT
1         TIME IN SECONDS - FIRST COLUMN
1         CURRENT IN CIRCUIT - THIRD COLUMN
1         VOLTAGE ACROSS CAPACITOR - FOURTH COLUMN

          COEF
          1      3.0
          2      1.0
          3      1.0

    CONTROL    6
          1      3
  2  10.
   3    .001
          4      1
  5  0
   6 0
          START
          0.
          4.
          0.
          0.
  LIST 1000
          WAIT
          TITLE
          RLC CIRCUIT CALCULATIONS-CRITICALLY DAMPED CASE
          REPEAT  1
          TABLE
1         SAME INITIAL CONDITIONS AS PREVIOUS RUN

       CONTROL  1
  2    8.0
          COEF
          1      2.0

       LIST  1000
          WAIT
          TITLE
          RLC CIRCUIT CALCULATIONS-UNDERDAMPED CASE
          REPEAT   1
    CONTROL     1
```

```
1          TABLE
           SAME INITIAL CONDITIONS AS BEFORE

           COEF
           1      1.0

           LIST    100
           END
```

15b

RLC CIRCUIT CALCULATIONS-OVERDAMPED CASE

1

15 C

DYNAMO INFORMATION TABLE

THE INITIAL CONDITIONS FOR THE RUN ARE

A) I(0)=0

B) V(0)=0

C) (D(I)/DT).0 = (E-VCO)/L = 4

E) THE CIRCUIT POTENTIAL IS 4 VOLTS

VARIABLE COEFFICIENTS

NUMBER 1 RESISTANCE

NUMBER 2 CAPACITANCE

NUMBER 3 INDUCTANCE

OUTPUT

TIME IN SECONDS - FIRST COLUMN

CURRENT IN CIRCUIT - THIRD COLUMN

VOLTAGE ACROSS CAPACITOR - FOURTH COLUMN

LIST OF ENTERED COEFFICIENTS

COEF. NO. 1 THE VALUE IS 0.30000E 01

COEF. NO. 2 THE VALUE IS 0.10000E 01

COEF. NO. 3 THE VALUE IS 0.10000E 01

RLC CIRCUIT CALCULATIONS-OVERDAMPED CASE

FIRST COLUMN- INDEPENDENT VARIABLE(X). DEPENDENT VARIABLES(Y) FOLLOW SIX PER ROW

0	0.00000E 00	0.40000E 01	0.00000E 00	0.00000E 00
1000	0.99999E 00	-0.12472E 00	0.10904E 01	0.85341E 00
2000	0.19999E 01	-0.29337E 00	0.82378E 00	0.18220E 01
3000	0.29999E 01	-0.21542E 00	0.56805E 00	0.25112E 01
4000	0.39999E 01	-0.14813E 00	0.38813E 00	0.29837E 01
5000	0.49999E 01	-0.10118E 00	0.26493E 00	0.33063E 01
6000	0.59999E 01	-0.69069E-01	0.18082E 00	0.35266E 01
7000	0.69999E 01	-0.47141E-01	0.12341E 00	0.36769E 01
8000	0.79999E 01	-0.32174E-01	0.84235E-01	0.37794E 01
9000	0.89999E 01	-0.21960E-01	0.57492E-01	0.38495E 01

RLC CIRCUIT CALCULATIONS-CRITICALLY DAMPED CASE

DYNAMO INFORMATION TABLE

SAME INITIAL CONDITIONS AS PREVIOUS RUN

RLC CIRCUIT CALCULATIONS-CRITICALLY DAMPED CASE

LIST OF ENTERED COEFFICIENTS

COEF. NO. 1 THE VALUE IS 0.20000E 01

COEF. NO. 2 THE VALUE IS 0.10000E 01

COEF. NO. 3 THE VALUE IS 0.10000E 01

RLC CIRCUIT CALCULATIONS-CRITICALLY DAMPED CASE

FIRST COLUMN- INDEPENDENT VARIABLE(X). DEPENDENT VARIABLES(Y) FOLLOW SIX PER R

0	0.00000E 00	0.40000E 01	0.00000E 00	0.00000E 00
1000	0.99999E 00	0.23819E-06	0.14715E 01	0.10569E 01
2000	0.19999E 01	-0.54134E 00	0.10826E 01	0.23759E 01
3000	0.29999E 01	-0.39829E 00	0.59744E 00	0.32034E 01
4000	0.39999E 01	-0.21978E 00	0.29305E 00	0.36336E 01
5000	0.49999E 01	-0.10780E 00	0.13475E 00	0.38383E 01
6000	0.59999E 01	-0.49575E-01	0.59490E-01	0.39306E 01
7000	0.69999E 01	-0.21885E-01	0.25532E-01	0.39708E 01
8000	0.79999E 01	-0.93930E-02	0.10734E-01	0.39879E 01

RLC CIRCUIT CALCULATIONS-UNDERDAMPED CASE

15 J

DYNAMO INFORMATION TABLE

SAME INITIAL CONDITIONS AS BEFORE

RLC CIRCUIT CALCULATIONS-UNDERDAMPED CASE

15 J

LIST OF ENTERED COEFFICIENTS

COEF. NO.	1	THE VALUE IS	0.10000E 01
COEF. NO.	2	THE VALUE IS	0.10000E 01
COEF. NO.	3	THE VALUE IS	0.10000E 01

RLC CIRCUIT CALCULATIONS-UNDERDAMPED CASE

10
15 K

FIRST COLUMN- INDEPENDENT VARIABLE(X). DEPENDENT VARIABLES(Y) FOLLOW SIX PER ROW

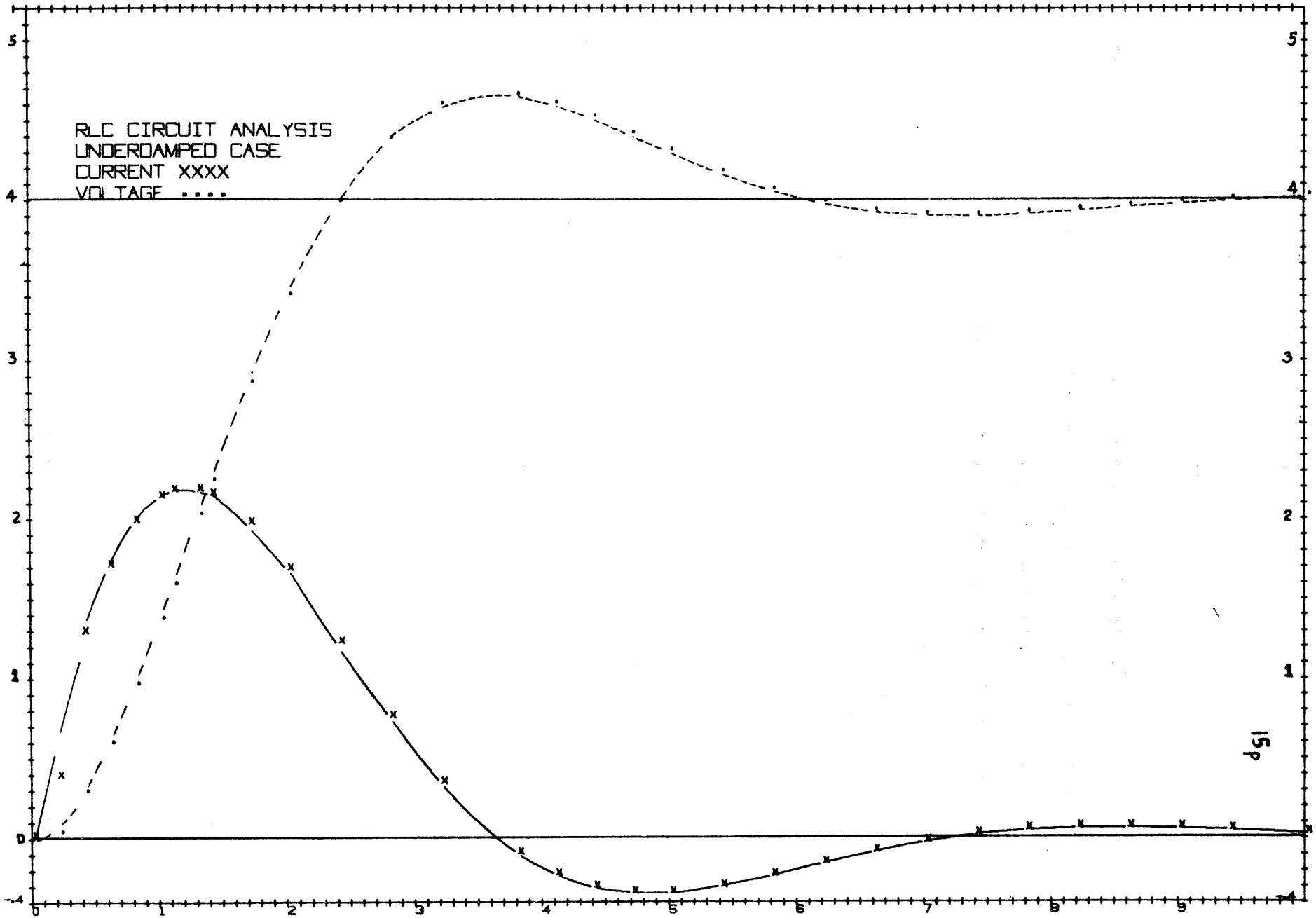
0	0.00000E 00	0.40000E 01	0.00000E 00	0.00000E 00
100	0.99999E-01	0.36006E 01	0.38001E 00	0.19333E-01
200	0.19999E 00	0.32050E 01	0.72025E 00	0.74676E-01
300	0.29999E 00	0.28166E 01	0.10212E 01	0.16207E 00
400	0.39999E 00	0.24384E 01	0.12839E 01	0.27765E 00
500	0.49999E 00	0.20729E 01	0.15093E 01	0.41762E 00
600	0.59999E 00	0.17226E 01	0.16990E 01	0.57833E 00
700	0.69999E 00	0.13892E 01	0.18544E 01	0.75628E 00
800	0.79999E 00	0.10743E 01	0.19774E 01	0.94814E 00
900	0.89999E 00	0.77922E 00	0.20700E 01	0.11507E 01
1000	0.99999E 00	0.50477E 00	0.21340E 01	0.13612E 01
1100	0.10999E 01	0.25163E 00	0.21716E 01	0.15766E 01
1200	0.11999E 01	0.20195E-01	0.21850E 01	0.17947E 01
1300	0.12999E 01	-0.18941E 00	0.21764E 01	0.20129E 01
1400	0.13999E 01	-0.37727E 00	0.21479E 01	0.22293E 01
1500	0.14999E 01	-0.54366E 00	0.21017E 01	0.24419E 01
1600	0.15999E 01	-0.68905E 00	0.20398E 01	0.26491E 01

RLC CIRCUIT CALCULATIONS-UNDERDAMPED CASE

FIRST COLUMN- INDEPENDENT VARIABLE(X). DEPENDENT VARIABLES(Y) FOLLOW SIX PER ROW

10200	0.10199E 02	-0.28095E-01	0.15698E-01	0.40124E 01
10300	0.10299E 02	-0.26781E-01	0.12952E-01	0.40138E 01
10400	0.10399E 02	-0.25338E-01	0.10345E-01	0.40150E 01
10500	0.10499E 02	-0.23791E-01	0.78887E-02	0.40159E 01
10600	0.10599E 02	-0.22165E-01	0.55902E-02	0.40166E 01
10700	0.10699E 02	-0.20484E-01	0.34573E-02	0.40170E 01
10800	0.10799E 02	-0.18767E-01	0.14946E-02	0.40173E 01
10900	0.10899E 02	-0.17035E-01	-0.29559E-03	0.40173E 01

+HOOKER CHEMICAL CORPORATION+ CORPORATE ENGINEERING



ACKNOWLEDGMENTS

I would like to thank my supervisor Marliss O. Bird for his understanding and encouragement throughout the preparation of this paper. The suggestions of G. A. R. Trollope have resulted in a considerable improvement of the DYNAMO capabilities.

APPENDIX A

DYNAMO INPUT LANGUAGE

#

The following is a list of the DYNAMO input commands and the functions they initiate during the program execution:

A) INTEGRATION and STORAGE control:

CONTROL M

This command causes the program to accept M data cards. These cards may be used to define: a) The number of equations describing the system, b) The integration increment (delta), c) The total integration domain, d) The initial conditions files to be used, e) The frequency for storing the integration data on the disk, and f) The disk record where the first X-Y data are to be stored.

COEF

This command permits the entering of the values for the variable coefficients of the differential equations.

B) INITIAL CONDITIONS control:

START

This command permits the entering of the initial conditions for the first step of the calculations.

REPEAT M

The program will return M-1 steps back and it will begin calculations with the initial conditions of this step.

RESTART M

Depending on the value of number M an interrupted run can be restarted at different points.

RESET

The last X-Y record of the previous run is used as the initial conditions of this run. Other integration variables can be changed through the CONTROL command.

C) IDENTIFICATION control:

TITLE

The contents of the card following the TITLE card will be printed as a heading to all output pages.

TABLE

A maximum of twenty cards with information pertinent to the input data or the model can be read and stored with this command. This information is later printed on the 1132 Printer.

D) OUTPUT control:

LIST M

The calculated results from every M^{th} integration step will appear on the 1132 Printer.

PLOT M

This command specifies that M sets of plotting instructions follow.

FILE OP = n START = m END = k

This command indicates some operation on the data files, depending on the value of the OP code. (storing, listing, punching). M and K indicate the starting and ending records for listing or punching only. When OP is negative no calculations need preceed the file Operation. The number of variables to be dumped or listed is controlled by the CONTROL command.

E) UTILITY commands:

SETUP:

This command resets the standards files during program execution.

LOG, NOLOG

These commands control the listing of the input data on the 1131 console.

F) END INDICATORS

WAIT

Stop input and proceed with execution. Return for more data.

HALT

Stop input and pause. When the START button is pressed resume execution of the input.

END

Stop input and proceed with execution. Return to monitor.

APPENDIX B

PLOTTING INSTRUCTIONS

THE CARDS FOLLOWING THE '' PLOT M '' COMMAND CONSTITUTE A SIMPLIFIED APPROACH TO DIGITAL PLOTTING. A SET OF PLOTTING SPECIFICATIONS, SIMILAR TO THE ONES LISTED BELOW MUST BE PRESENT FOR EVERY ONE OF THE M PLOTS SPECIFIED.

```
      PLOT    2    ***  DYNAMO COMMAND****
   NUMBER = 1  TITLE'PLOT NO. 1'
   X AXIS   TYPE = 12 TITLE' INCHES OF WATER' MIN = .001  MAX =  .2          C
   SHIFT = 0
   VARIABLES
   FILE NO. =1
*
   Y LEFT AXIS  TYPE=12  TITLE'VOLUME OF WATER IN LITERS'
   VARIABLES
   FILE NO. = 2  LINE = 1000  GRAPH = 4122 CODE = 4251
   TAG ' DIAMETER 2 FEET'
   FILE NO. =3  LINE = 2511  GRAPH = 4122 CODE = 4281
   TAG' DIAMETER 3 FEET'
*
**
   GENERAL        ** INSTRUCTION TO  PERMIT ENTERING  GENERAL SPECS**
   PARALLEL X
   FRAME
   STOP
   NUMBER=2 TITLE ' SECOND PLOT'
   . . . .          . .
   . . . .          . .
   . . . .          . .
   . . . .          . .
   . . . .          . .
   . . . .          . .
   . . . .          . .
   STOP
   . . . .          . .
   . . . .          . .
   ** OTHER DYNAMO COMMANDS  **
   . . . .          . .
   . . . .          . .
```

THE PLOTTING INSTUCTIONS ARE USED TO DEFINE THE PLOT TITLE, THE TYPE OF THE AXES(LINEAR, LOGARITHMIC, POLAR), THE SIZE OF THE AXES(8.5 OR

11 INCHES) AND THEIR MAXIMUM OR MINIMUM LIMITS IF IT IS SO DESIRED. INPUT FOR DIFFERENT AXES IS SEPARATED BY (*) AND THE END OF ALL INPUT FOR THE AXIS CHARACTERISTICS IS INDICATED BY (**) .

THE FILE NO. IS THE SEQUENCE NUMBER OF THE VARIABLE IN THE FILE OBSERVATIONS. THE FIRST VARIABLE IS TO BE THE INDEPENDENT VARIABLE X AND VARIABLES 2 AND 3 ARE TO BE PLOTTED ON THE LEFT HAND Y AXIS. THE** LINE** ENTRY IDENTIFIES THE TYPE OF LINE(CONTINUOUS OR DOTTED) AND THE **GRAPH** ENTRY IDENTIFIES THE TYPE OF PLOT(POINTS,LEAST SQUARES,FILL IN LINES, E.T.C.). THE **CODE** ENTRY IDENTIFIES THE CODE FOR THE TAG THAT FOLLOWS IN THE NEXT CARD.

THE GENERAL SPECIFICATIONS FOLLOW WITH THE **STOP** CARD INDICATING THE END OF DATA INPUT FOR THE FIRST PLOT.

APPENDIX C

FORTRAN-ALGEBRAIC EQUIVALENCE

OF MATHEMATICAL MODEL

The two systems of differential equations that were presented in PART TWO have to be converted into function subprograms for use with the DYNAMO Simulator. The FORTRAN format of the differential equation model is shown below for both of the cases illustated on pp. 14- 16.

CASE NO. 1 The CSTR battery

This is an example of straight forward application. The derivatives of of the concentrations of the reacting component in each of the tanks with respect to time are identified through subscript J and a computed GO TO statement:

```
// DUP
*DELETE              F
// FOR
*EXTENDED PRECISION
*ONE WORD INTEGERS
* LIST ALL
      FUNCTION F(J,L)
      COMMON JCARD(80),LC(70),AM,DELX,COEF(40),INDIC(40)
      COMMON DIST(6,10),X(5),Y(20,5),IX(2),IY(20,2)
      GO TO(1,2),J
    1 F= 1.   - Y(1,L) *(1.+ .5*Y(1,L))
      GO TO 3
    2 F= Y(1,L)- Y(2,L)*(1. + .5*Y(2,L))
    3 RETURN
      END
// DUP
*STORE      WS  UA  F
```

CASE NO. 2 The RLC circuit

This is a case where the higher order derivative must be reduced to a first order derivative. The reduction method can be outlined as follows:

1. Set Y1 equal to the first order derivative
2. The derivative of Y1 is the second derivative and it can be expressed in terms of the differential equation.
3. The second equation, describing variable Y2 is the first order derivative, that is, d/dt(Y2) = Y1

The same type of approach will reduce higher oder equations.

```
// DUP
*DELETE              F
// FOR
*ONE WORD INTEGERS
* LIST ALL
*EXTENDED PRECISION
      FUNCTION F(J,L)
      COMMON JCARD(80),LC(70),AM,DELX,COEF(40),INDIC(40)
      COMMON DIST(6,10),X(5),Y(20,5),IX(2),IY(20,2)
      GO TO (1,2,3),J
C.........FIRST EQUATION IS THE SECOND DERIVATIVE
    1 A=(1./COEF(3))* ((1./COEF(2))*Y(2,L)+COEF(1)*Y(1,L)   )
      F= -A
      GO TO 4
    2 F= Y(1,L)
      GO TO 4
    3 F=(1./COEF(2))*Y(2,L)
    4 RETURN
      END
// DUP
*STORE       WS  UA  F
```

APPENDIX D

LIST OF SYSTEM'S OPERATING STANDARDS

DYNAMO STANDARDS FILES GENERATOR

1	NUMBER OF EQUATIONS	0
2	PROGRAM RETURN CODE	2
3	FILE INITIALIZATION INDICATOR	0
4	FILE NUMBER FOR THE FIRST INITIAL COND	0
5	INTERRUPTION POINT INDICATOR	0
6	RUN TITLE INDICATOR SWITCH	0
7	PLOTTING INDICATOR SWITCH	0
8	INITIAL CONDITION INDICATOR	0
9	LISTING OPTION SWITCH	0
10	LISTING AND/OR PUNCHING OF THE RESULTS	0
11	LISTING FREQUENCY FOR 1132 PRINTER	100
12	NUMBER OF PLOTS REQUESTED	0
13	CURRENT INITIAL CONDIT. FILE RECORD	0
14	CONTROL COMMAND INDICATOR	0
15	LISTING OF THE TABLE OF VARIABLES	0
16	UNEVEN FILE ALLOCATION INDICATOR	0
17	NEXT INITIAL CONDITIONS FILE RECORD	0
18	INTERRUPTED RUN INDICATOR	0
19	NO OF INCREMENTS OF THE NUMERICAL INTE	0
20	LAST CASE REGARDLESS OF ALL INDICATORS	0
21	DISK USAGE INDICATOR	0
22	STORE FREQENCY(RESULTS TO FILE)	0
23	RESERVED FOR SYSTEM USE	0
24	RESERVED FOR SYSTEM USE	0
25	RESERVED FOR SYSTEM USE	0
26	LOW BOUND/INITIAL CONDITIONS FILES	0
27	MAXIMUM RETURN FOR REPEAT COMMAND	0
28	START PUNCHING FROM FILES AT REC.	0
29	STOP PUNCHING FROM FILES AT REC.	0
30	INDICATOR SWITCH PLOT NO. 1	0
31	PLOT NO. 2	0
32	PLOT NO. 3	0
33	PLOT NO. 4	0
34	PLOT NO. 5	0
35	PLOT NO. 6	0
36	PLOT NO. 7	0
37	PLOT NO. 8	0
38	PLOT NO. 9	0
39	PLOT NO. 10	0
40	PLOT NO. 11	0
41		0
42	1132 PRINTER LINES/LINE OF OUTPUT	0
43	PRINTER SKIP CONTROL	0
44	FILES STORAGE FREQ. CONTROL	0
45	INTEGRATION INTERVAL COUNTER	0
46	PRINTER LIST CONTROL	0
47	NO. OF NEXT ENTRY REC. IN DATA FILE	0
48		0
49		0
50		0
51	OUTPUT PAGE NUMBER	0
52	TYPEWRITER LOG CONTROL	1
53	LISTING FROM FILES STARTS AT REC.	0
54	LISTING FROM FILES STOPS AT REC.	0
55	RETURN FROM PLOTTING INDICATOR	0

IBM Publications

Manual No.	
H20-0282-0	1130 Continuous System Modeling Program-Program reference manual
C26-3750-	IBM 1130 Disk Monitor System-Reference Manual
C26-5933-3	IBM 1130 FORTRAN Language
B20-0001-0	General Purpose Systems Simulator III

BIBLIOGRAPHY

(1) Hamming R. W. 'Stable Predictor-Corrector Methods for ordinary Differential Equations'
J. Assoc. Comp. Mach volume 6, 1959, pp. 37-47

(2) Ralston Anthony 'Numerical integration methods for the solution of ordinary differential equations'
Mathem. methods for Digital Computers, chapt. 8. pp 95-109
John Wiley & Sons, Inc.

(3) Milne W. E. and Reynolds R.R. 'Fifth-Order methods for the numerical solution of ordinary Differential equations
J. Assoc. Comp. mach. volume 9, 1962, pp. 64-70

(4) Walas S. M. "Reaction Kinetics for Chemical Engineers" pp. 93-97
McGraw-Hill New York 1959

(5) Miller K.S and Walsch J.B "Introductory electical circiuts"
John Wiley and Sons, Inc. New York 1960

(6) Franks R.G.E. "Mathematical modeling in Chemical Engineeeing"
John Wiley and Sons, Inc. New York 1967

M.2.2[6]

Abstract for December Common Meeting

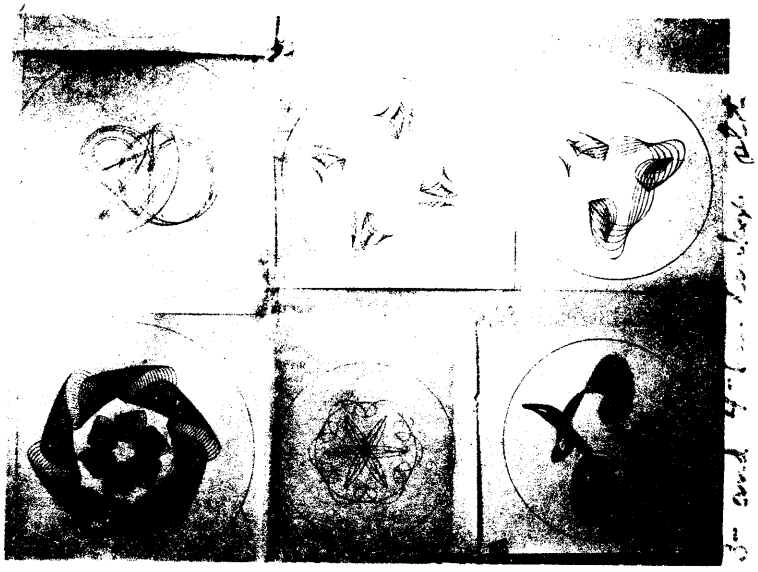
Robert L. Cushman
ARCON Corporation
Wakefield, Massachusetts

THREE-BAR AND FOUR-BAR LINKAGE SYSTEM WITH PLOTTED OUTPUT

A mathematical model of a linkage system is controlled by an array of inputs to plot out the path of the linkage intersection point. In addition, the system simulates turning the entire mechanism about a major axis. The output of the system has yielded some very interesting designs.

In conjunction with this report, a list-oriented free field floating point input program has been developed. This routine enables the control program to modify only specified items of the system input. Under break character control, an automatic iterative procedure can be set up.

Some sample outputs are shown in the enclosed photo.



COMMON
San Francisco, California

PROJECT: Management Installation Division, Operation Project

SUBJECT: The Systems Reference Library

SPEAKER: Mr. G. W. Goesch, Manager, Product Publications
IBM Corporation, San Jose, California
Telephone (408) 227-7100

FOR
PRESENTATION: Monday, December 11, 10:30 AM, Session M.2.6
8 Pages Text

The System Reference Library

I'm Gordon Goesch, Product Publications Manager, IBM, San Jose. There is a Product Publications group at each of the Development Laboratories, as well as a Programming Publications group at most of them. And, of course, there is a department in White Plains that prepares Application descriptions. These groups all provide input to the System Reference Library, and that, rather than my own department, is the subject of my remarks.

A manual looks quite simple, and sometimes it's hard for our own field people and for customer people to understand why it seems so difficult to get a few manuals written, and then to get them delivered while they are still fresh. There is a great number of manuals that must be kept current and kept in stock, and record-keeping alone is a sizeable task. The System Reference Library is the current method of indexing and classifying system manuals, and is the latest of a series of different methods, each designed to solve the problems that the previous methods couldn't handle. Problems created by new and very complex systems. This method isn't perfect either, as you are well aware, but we are certainly constantly trying to improve it. (And I have one improvement to announce today, a little later.)

That is the reason I am always happy to talk and listen to groups such as yours about our publications. It gives us an opportunity to discuss with you our Publication Library; its organization, its purpose, and its distribution system - because even with a good system you must understand how to use it if it is to be effective for you.

I think that this type of a meeting can be a two-way street for information: We explain the principles; you provide feedback. We do get feedback from you via Reader's Comment Forms - we want more of your comments and we certainly appreciate them.

I don't know how knowledgeable you are about our publications; therefore, for the benefit of the new members and also for the updating of the veteran members, I'll run through a few slides which I think will tell you the publication story.

By the way, you have probably seen the publications display in the main convention lobby. The display, I am sure, contains publications of interest to you. Many of the publications on display have been published since your last COMMON meeting. Feel free to examine them in detail but please do not take them away, as there is only one copy of each and we want everyone to benefit from the display.

Slide #1 BOL	To begin, we have what we call the IBM Branch Office Library. BOL contains a great amount of information, the major portion of which are publications for users of our equipment.
Slide #2 FYI	In order to call your attention to those publications in the BOL that are <u>For Your Information</u>
Slide #3 Swamped	and to help you avoid being swamped by ordering blindly as the man shown in the slide
Slide #4 SRL	we have developed the Systems Reference Library (SRL). As you probably know, the SRL is a rather extensive library system.
Slide #5 Library Shelves	To give you an understanding of how to approach this volume of material
Slide #6 Mr. SRL Helps Select	let Mr. SRL help you make the correct selection.
Slide #7 What, Where, How	He will acquaint you with the SRL and tell you: a. <u>What</u> is available b. <u>Where</u> you can get the information c. and, <u>how</u> you can go about getting it.
Slide #8 System	You are probably interested in establishing a library for only one system type,
Slide #9 System X	and for only a particular configuration of that system. Considerable thought and effort was spent in the design of the Systems Reference Library, to allow IBM people and customer people to select that material that is of particular interest to them.
Slide #10	Each SRL is an encyclopedia for a <u>particular</u> system - with separate publications for the <u>major</u> <u>subject</u> areas. It consolidates all the basic reference literature necessary for you to: Plan, Program, Install, and Operate that system.
Slide #11 SRL Key	The <u>key</u> you need for opening any of the System Reference Libraries

Slide #12

is the SRL Bibliography for your system.

Currently there are 13 System Reference Libraries, ranging from the 1130 to the System 360 - and of course each system has its own separate Bibliography.

Each Bibliography is actually an Index of all the current publications about a specific system. In it, publications are listed both by subject code and by machine number - and it contains abstracts describing each available publication. (We will discuss subject code a little later.)

Slide #13
Bibliography & SRL Newsletter

Now, how do we update the Bibliography?

Each Bibliography has its own Newsletter (green for easy identification) and it is issued monthly (when there are changes). The Bibliography Newsletter updates the Bibliography by:

1. providing titles and abstracts of new publications
2. providing form numbers of Technical Newsletters
3. Listing type 1 programs and their latest modifications, and
4. Listing obsolete form numbers.

Actually, the Bibliography newsletter is a current "Accumulative Index of Publications and Programs" available for a given system.

To keep the Newsletter from getting too large, Bibliographies are periodically scheduled for revision - when that occurs, the information from the current newsletter is merged into Bibliography.

Slide #14
DPT

An additional source of information is the "Data Processing Techniques" (DPT) Bibliography (Form F20-8172). It indexes a series of publications of techniques for Study, Analysis, Design, Implementation, Programming, Documentation, Installation, Operation, etc.

Slide #15
Series of DPTs

This slide shows some of the DPT manuals.

Slide #16
KWIC

Another excellent Index for your use is the KWIC Index of Marketing Publications (Form 320-1621).

Slide #17
KWIC &
TNL

It is published quarterly and updated by a monthly TNL.

The KWIC Index is based on an abbreviated 30-position publication title - listing and cross-referencing publications by the important words in the title.

The KWIC Index is listed in five ways:

1. Alphabetically, by all key words in the title
2. Machine or System Type
3. Form Number
4. Type I & II Programs in System Sequence
5. Type III & IV Programs in System Sequence

Slide #18
3 Way List
1-2-3

For example, this slide shows 3 separate word listings for a single publication - "Programs for Petroleum Engineering"

Slide #19
2 More List
4-5

and here, the same publication listed by machine number and by form number.

The latest KWIC Index was prepared from 11,546 publication titles which generated 29,122 listings.

Actually, the KWIC Index lists many publications other than SRL publications, such as Executive Guides and Brochures, tools and techniques manuals, applications manuals and briefs, educational material.

Therefore, to start a System Reference Library, it is necessary that you work with your IBM representative, using the three key publications we have talked about, the:

1. Bibliography and its Newsletter
2. the DPT Bibliography
3. KWIC Index of Marketing Publications

You can select and build your own reference library to support your system.

However, please do not order your publications by writing to Product Publications (the address shown on the manual) or our Distribution Center in Mechanicsburg, as it will only delay the order. You must order through your local IBM representative, and I will review the details of them shortly.

Now lets talk about some of the other interesting and useful features of the library.

Slide #20
SRL
Masthead

Each SRL publication, listed in a System Bibliography, is identified by a file number and a form number, located on the upper right hand corner of the publication cover - as shown in the slide.

The file number performs two functions: the first part specifies the system(s) number; the last two digits the Subject Code. The Subject Code structure is a group of two-digit numbers (00-99) assigned to the various system components, e. g.

00	General System material; Bibliographies, System Summaries, Configurators
01	Machine System (CPU)
03	Input/Output Units
05	Magnetic Tape Units
20-50	Programming Systems
60	Application Manuals

The subject code 13 shown in this slide, indicates that this publication is about Special and Custom features.

Incidentally, these subject codes are all defined by their use in Part I of each Bibliography.

The form number is self explanatory; however, the form number suffix indicates the edition, or editorial level, of the publication.

Slide #21
SRL TNL

Because of the nature of computers and of the computer industry, it is frequently necessary to make changes to manuals. Just as green newsletters update the Bibliography, so do regular newsletters update any publication in the Library.

In most cases TNL packages are made up of an identifying cover page and replacement change pages for the parent publication - when you receive such a TNL you merge it into its parent manual and throw away the old pages.

Incidentally when you order a publication you will automatically receive the latest technical level copy as well as all the outstanding TNLs against that publication.

Slide #22
TNL
Masthead

This slide shows the upper right hand corner of a TNL. It indicates how the TNL identifies itself with its parent publication.

It carries the file number and form number of the parent publication it updates (and it is form-number-suffix sensitive)

Below that is the TNL's own number, publication date of the TNL, and the form numbers of any previous TNLs outstanding against the parent publication.

Incidentally, all page replacement TNL pages carry similar identifying information.

Outstanding TNLs are incorporated into the parent publication when it is being revised. TNLs may also be merged into the parent publication when it is being reprinted.

Information regarding the technical level of a publication, and the TNLs that may have been merged into it, is printed inside the front cover of each SRL manual.

Slide #23
SRL, NL
Parent Pub.
Manual

This slide shows a manual, a corresponding TNL, and a green SRL newsletter. With this combination on hand you have all the publishing reference you need for the parent manual.

Keep in mind that the best SRL publishing information source is the green SRL newsletter, because it not only updates its own parent publication (the bibliography) but also lists all the existing publications that are current for that system as well as their outstanding TNLs.

Slide #24
2 Biblios
2 NLs

and of course that each major system has its own bibliography.

Slide #25
In-Out

Thus, the SRL system keeps you well posted on what is in (current) and what is out (obsolete) for an effective library.

Slide #26
Wrapped Up

Basically, you have at your disposal a "living-doll" of a library system.

How can you get new manuals and TNLs after you have established your library?

a. If you desire, you may continue ordering specific publications through the IBM Branch Office.
b. However, you may prefer to use the SRL Subscription Service that the System Reference Library offers.

Slide #27
SRL, TNL
Revision
Service

The SRL/SS provides automatic shipping of new and revised manuals, and newsletters, directly to you without having to go through the IBM Branch office each time.

All Systems Reference Library manuals listed in SRL Bibliographies under subject codes 00 through 50 are available by profile.

This distribution of publications according to profile is designed to provide you with a basic, master library tailored to your needs.

Application Program manuals, Student Texts, Data Processing Techniques manuals, Course Description Bulletins, and other miscellaneous documents (except padded forms) listed in SRL Bibliographies under subject codes 60 to 99 can be subscribed to by form number.

In addition, and this is new, revision service on multiple copies of SRL's and PLM's under subject codes 00 through 50 may be ordered by form number.

Here's how it works:

Slide #28
Card

To convert existing Revision Service Subscribes, special pre-printed conversion forms are being sent to all the sales offices. The salesman will review your needs with you and establish your "profile", which is basically a description of your system and your applications. You should also identify those publications you want in multiple quantities and order the quantities you need. The salesman sends the form to the Distribution Center and you are in business. You will receive as many copies of the form-numbered publications as you ordered, and any new or revised (or newslettered) form that fits your profile.

Slide #29
Mailman
w/pkg.

For new subscribers the system is the same except that pre-printed forms obviously won't be used.

Slide #30
SRL & Rev
Pkg

Now, you are all set, you know what is available, you know how to select the publications, and you know where to get them.

Slide #31
The Key
Is Yours

The key is yours.

Slide #32
Key to
Knowledge
Slide #33
SRL

Use this key to open up a tremendous amount of timely knowledge about your system through the IBM Systems Reference Library. I hope that the result of all this, will make you as happy as the man in the next slide - careful planning of your library may help!

Most of you have probably noticed the Reader's Comment Form that is appearing on the back of many manuals these days. Some of you may have filled one or more out. The response to these has often been very gratifying and we appreciate it.

I want to encourage you to use them, as it is one of the means, along with meetings like this, by which we get the feedback from our readers that is essential if we are to improve our publications and make them more useful to you.

This form is self-addressed and post-paid and will be directed to the correct publications group. As a reminder, please don't use the form to order manuals as we have to redirect such orders back to the Branch Office with a consequent delay.

On behalf of the publications groups, I want to thank you for your attention and for your comments, and pledge to you our whole-hearted effort in providing first rate publications support for your systems.

ABSTRACT FOR PROGRAM LIBRARY PROJECT

Discussion of the proposed PREP Forms for the 1130, 1800, and 360 Library. There will also be a discussion of an extension of the minimal standards for this Library. It is requested that there be representatives from each of the machine system projects in the Systems Division present at this meeting.

M.3.1 (G)

Speech for COMMON

Linkage Editor	Slide
I. Introduction	(1)
45 minutes is a short period; therefore an overview	
II. Function of Editor	
A. Compiler turns a SOURCE module to an OBJECT MODULE	(2)
B. Linkage Editor takes OBJECT & LOAD Modules as input	(3)
1. Resolves external references (automatic library call)	
2. Replaces control sections, if required	
3. Produces aliases	
4. Creates a "relocatable" module	(4)
C. Overlay structures	
1. 5 control sections without overlay	(5)
2. 5 control sections with overlay	(6)
3. An overlay tree	(7)
a) Use of CALL or branch to cause overlay	(8-9)
b) Automatic loading of segments	
III. Dynamic Overlay Capabilities	
A. LINK	(10)
B. XCTL	(11)
C. LOAD	(12)
	(13)
IV. Discussion of Partitioned Data Sets	Foil
V. JCL and Control Card Example	1
	2

Speech for COMMON

Data Management

I. Introduction - general run-thru

A. Allocation to a volume
B. Allocation of SPACE if disk
C. Retrieval of Data Sets by name alone
D. Specify Data Set specs deferred to program execution

II. Access Methods

A. Sequential
B. Indexed
C. Direct
D. Partitioned
E. EXCP

III. Device Independence (mostly with Sequential Methods)

A. Characteristics specified by control card
B. Characteristics specified by Data Set Label

IV. Data Set Identification

A. Use of Catalog & Indexes
B. Generation Data Groups
C. DS name not in program, DD name is.

V. Qualities of Direct Access Volumes

A. VTOC
B. DSCB's & space accounting (must have label)

VI. Qualities of Tape Volumes

A. Labeled, unlabeled, NSL
B. Density
C. TRTCH

VII. Record Formats

A. F
B. V
C. U

VIII. Interface with OS

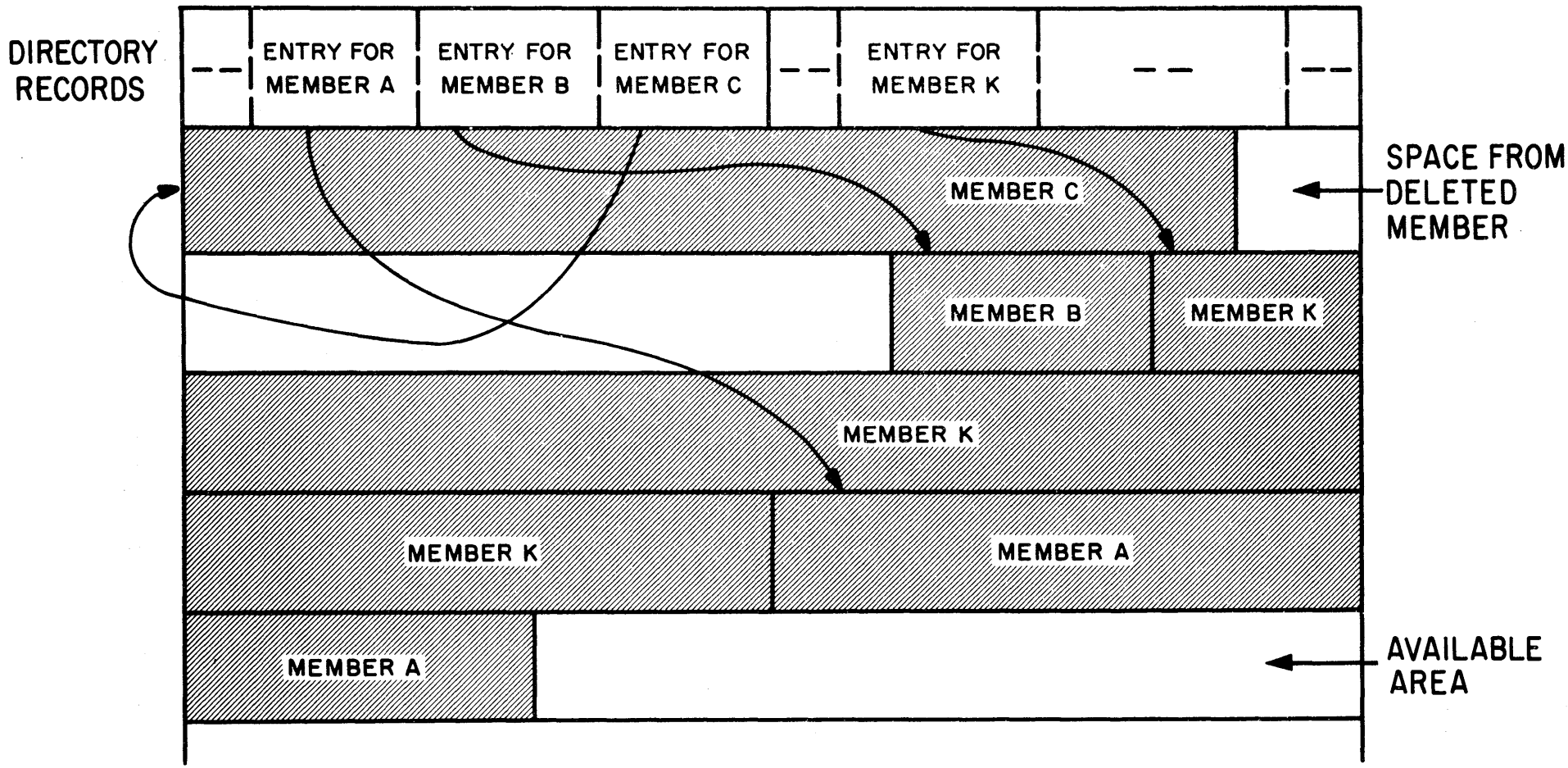
A. DCB
B. DD card
C. DS label
D. DCB exit

IX. Describing a Data Set

X. Processing Program Description

A. MACRF
B. EODAD
C. SYNAD
D. EXLIST

PARTITIONED DATA SET



PARTITIONED DATA SETS

BUILDING PDS AND INSERTING 1ST MEMBER.

```
                    :
//SYSLMOD  DD  DSNAME=NEWLIB(FIRSTPRG),          X
//               UNIT=2311,DISP=(NEW,KEEP),      X
//               SPACE=(TRK,(50,10,8)),          X
//               VOLUME=SER=333333
```

INSERTING 2ND MEMBER

```
//SYSLMOD  DD  DSNAME=NEWLIB(SECOND),            X
//               UNIT=2311,DISP=OLD,             X
//               VOLUME=SER=333333
```

EXECUTING A MEMBER FROM THE LIBRARY

```
//DEPTX    JOB   01,.....
//JOBLIB   DD    DSNAME=NEWLIB,UNIT=2311,        X
//               VOLUME=SER=333333,DISP=(OLD,PASS)
//STP1     EXEC  PGM=FIRSTPRG
           DD
           }
//STP2     EXEC  PGM=SECOND
```

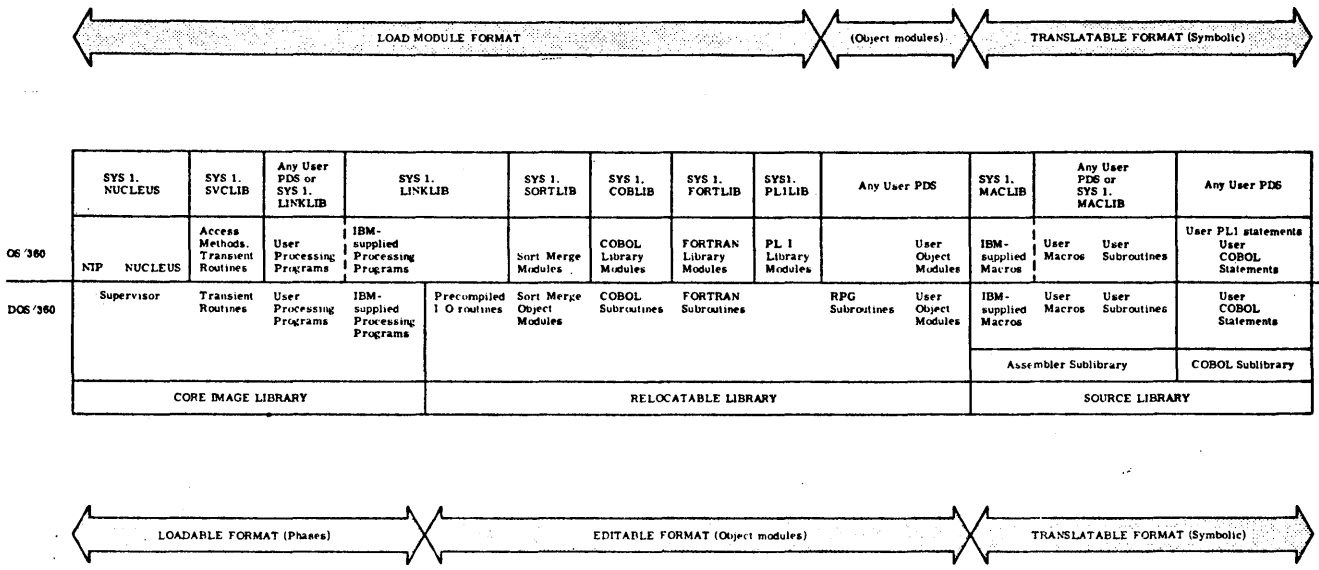


Figure 8. Program library correspondence between DOS/360 and OS/360

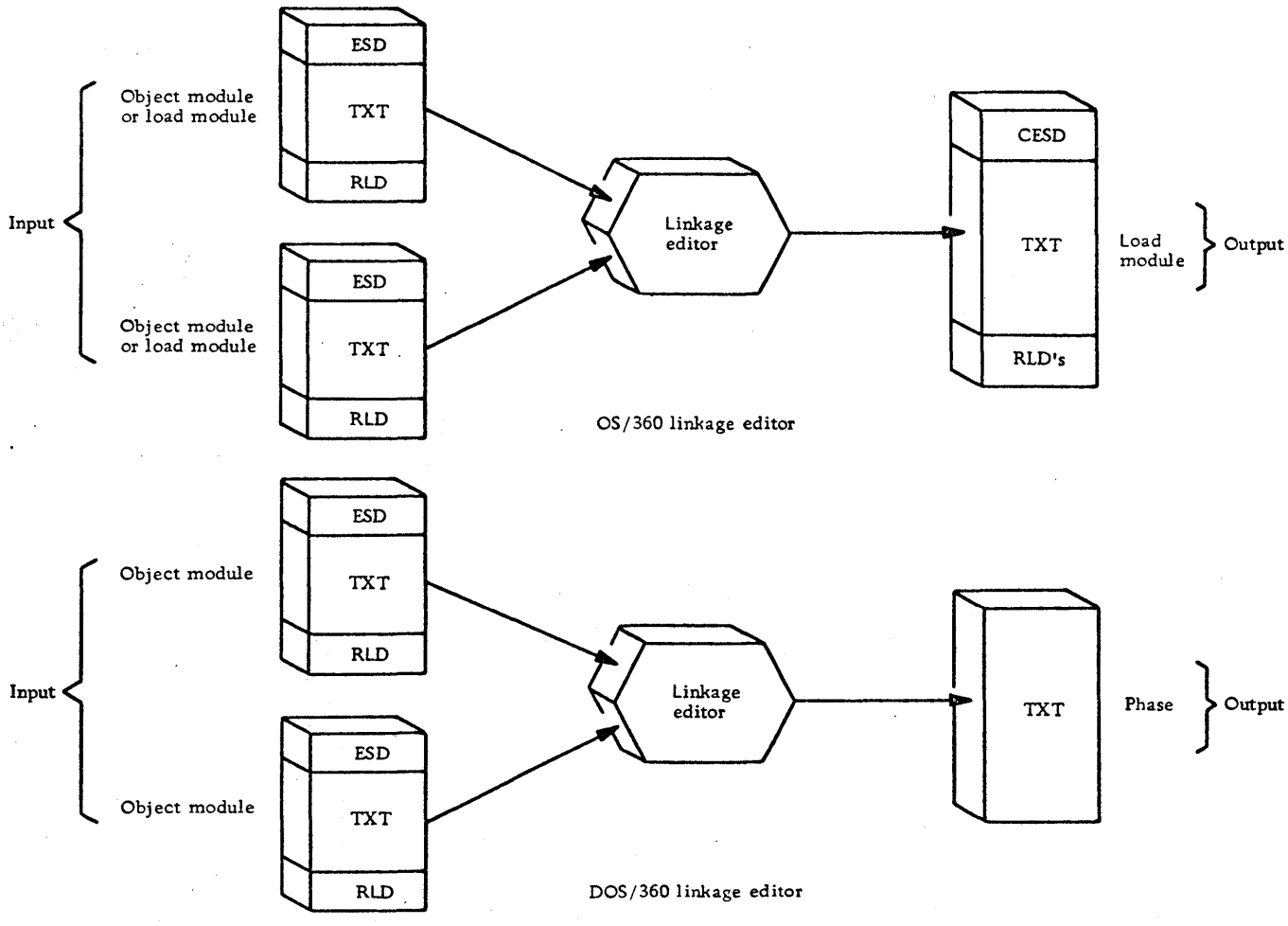


Figure 9. Physical difference in linkage editor output of the operating systems

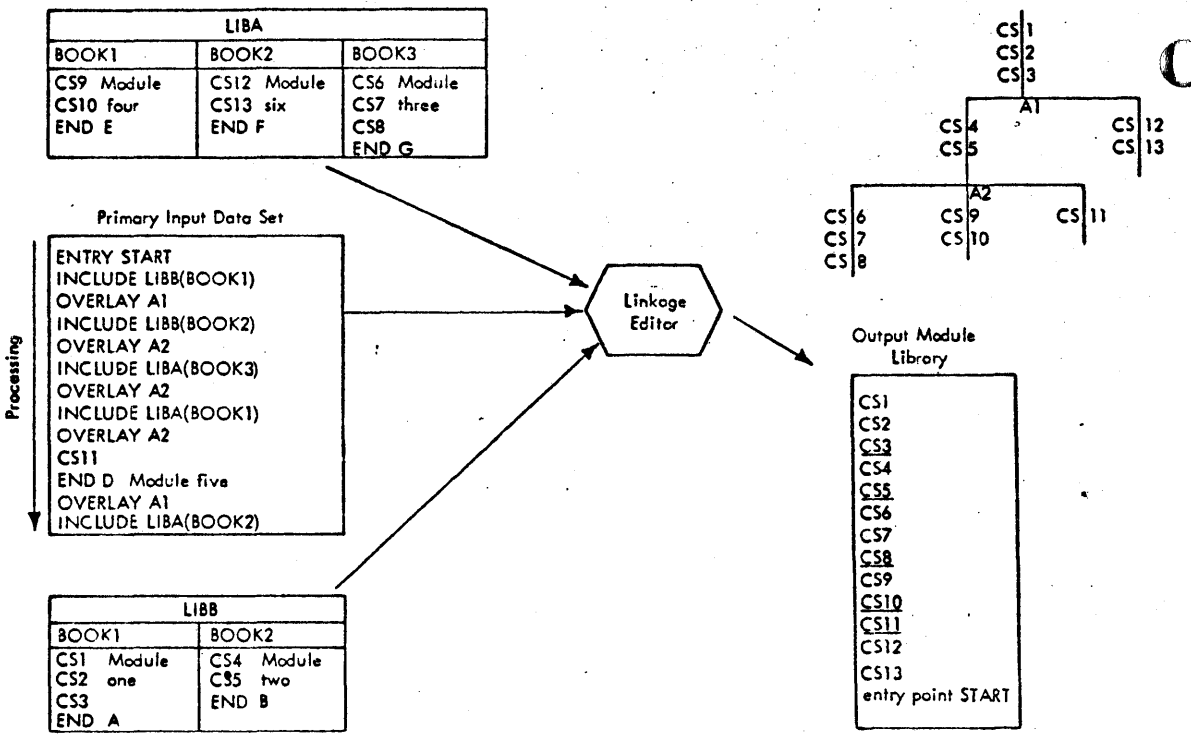


Figure 35. Processing an Overlay Program From Libraries

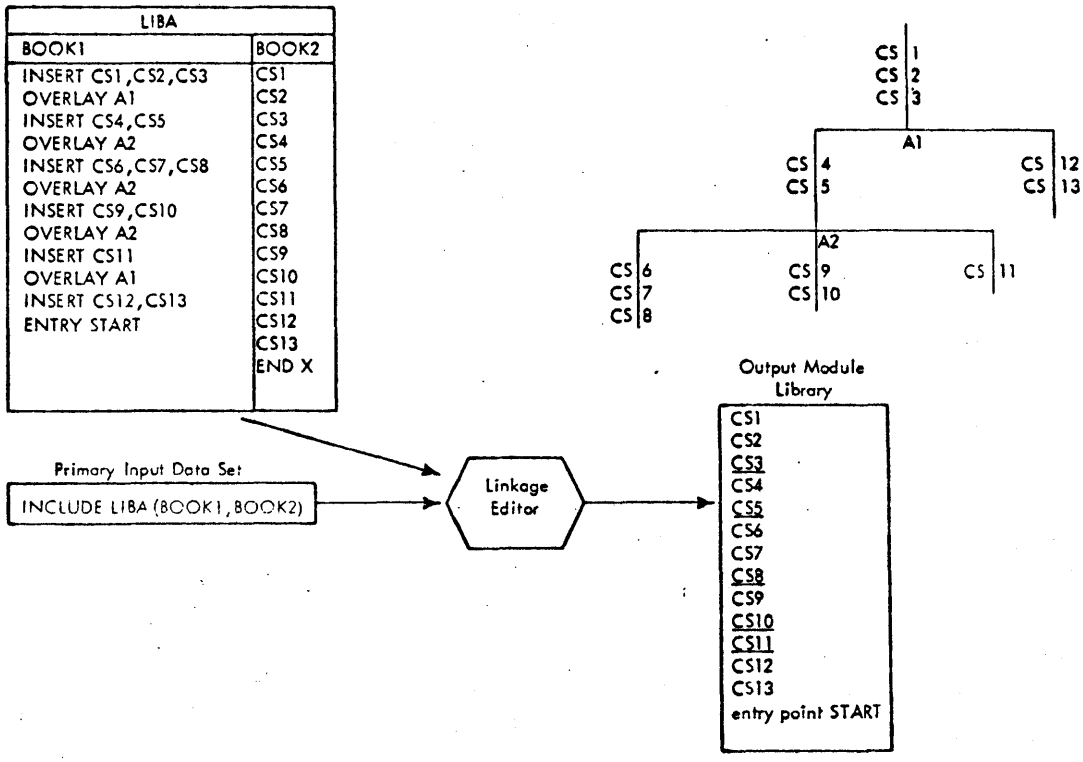


Figure 36. Processing an Overlay Program With Insert Statement

... primary input data set contains

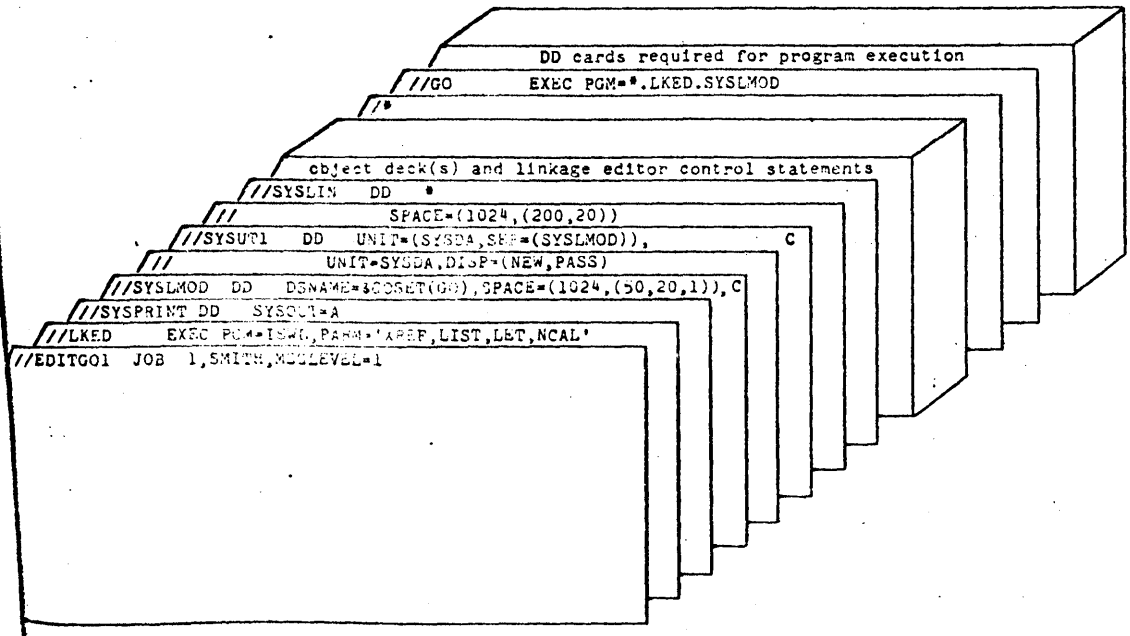


Figure 26. The Linkage Editor Step of an Edit and Execute Procedure

ALLOCATION OF SPACE

BLOCKS

SPACE = (300,(5000,100))

- 300 → PHYS. RECORD SIZE
- 5000 → PRIMARY QUANTITY
- 100 → SECONDARY QTY.

TRACKS OR CYLINDERS

SPACE = (TRK,(100,20))

SPACE = (CYL,(5,1)) — (1) → SECONDARY

- 5 → NUMBER OF CYLINDERS OR TRACKS
- 1 → PRIMARY

Note: If allocate by blocks, device independency is achieved.

ROLE OF DSNAME WHEN CREATING A DATASET

GENERATED DSCB NAME

```
//JOBNAME JOB
```

```
//ddname   DD   DSNAME=WORK,UNIT=2311,                    *   X
                SPACE=(TRK,(10)),                            X
```

44 CHARS

WORK

```
//ddname   DD   DSNAME=&WORK,UNIT=2311,                   *   X
                SPACE=(TRK,(10,5))
```

44 CHARS

WORK2.JOBNAME

```
//ddname   DD   UNIT=2311,SPACE=(TRK,(15))                *
```

44 CHARS

AAA~A.AA~A.A~A.A~A.00~01

* IMPLIED DISPOSITION = (NEW, DELETE)

CATALOGUED DATA SETS

TO CATALOG A DATA SET :

```
//MASTER   DD   DISP=(,CATLG),DSNAME=ACCTS.MASTER, X
//              UNIT=2400,VOLUME=SER=(T00410,T00200)
```

TO LOCATE A CATALOGUED DATA SET

```
//ddname   DD   DSNAME=ACCTS.MASTER,DISP=OLD
```

SYSCTLG

NAME	VOLUMES	UNIT
ACCTS.MASTER	T00410	2400
	T00200	
ABC.G0099V00	XXXXXX	2311
ABC.G0100V00	444444	2311
ABC.G0101V00	555555	2311

GENERATION DATA GROUPS

```
//OLDMAST  DD   DSNAME=ABC(0),DISP=OLD
```

```
//NEWMAST  DD   DSNAME=ABC(+1),DISP=(,CATLG),    X
//              UNIT=2311,VOLUME=SER=555555,SPACE=(CYL,50)
```

OVERRIDING DD PARAMETERS IN CATALOGUED PROCEDURES

ASMFCLG IN SYS1.PROCLIB

```
//STP1    EXEC   PGM=IEUASM
//SYSLIB  DD     DSNAME=SYS1.MACLIB,DISP=OLD
//SYSPUNCH DD    DSNAME=SYS1.UT4,DISP=OLD
//SYSUT1  DD     ~~~
          :
//SYSPRINT DD    SYSOUT=A
//STP2    EXEC   PGM=LINKEDIT,PARM='LET,LIST'
          :
//SYSPRINT DD    SYSOUT=A
//SYSLIN  DD     DSNAME=SYS1.UT4,DISP=OLD
//SYSLMOD DD     ~~~
//STP3    EXEC   PGM=*.STP2.SYSLMOD
```

** EXECUTING THE PROCEDURE **

```
//TEST    JOB    01,TESTING-PROC,MSGLEVEL=1
//A       EXEC   ASMFCLG,PARM.STP1=LOAD
//STP1.SYSPUNCH  DD  UNIT=2400,LABEL=(,NL),DISP=(,PASS)
//STP1.SYSPRINT DD  DUMMY,SYSOUT=,DSNAME=&NOTHING
//STP1.SYSGO DD  UNIT=2540-2
//STP1.SYSIN DD  *
                 } SOURCE PROGRAM
/*
//STP2.SYSLIN    DD  DSNAME=*.STP1.SYSPUNCH
//STP3.DD1       DD  ~~~
//STP3.DDN       DD  *
                   } DATA
/*
```

USE OF DCB PARAMETER

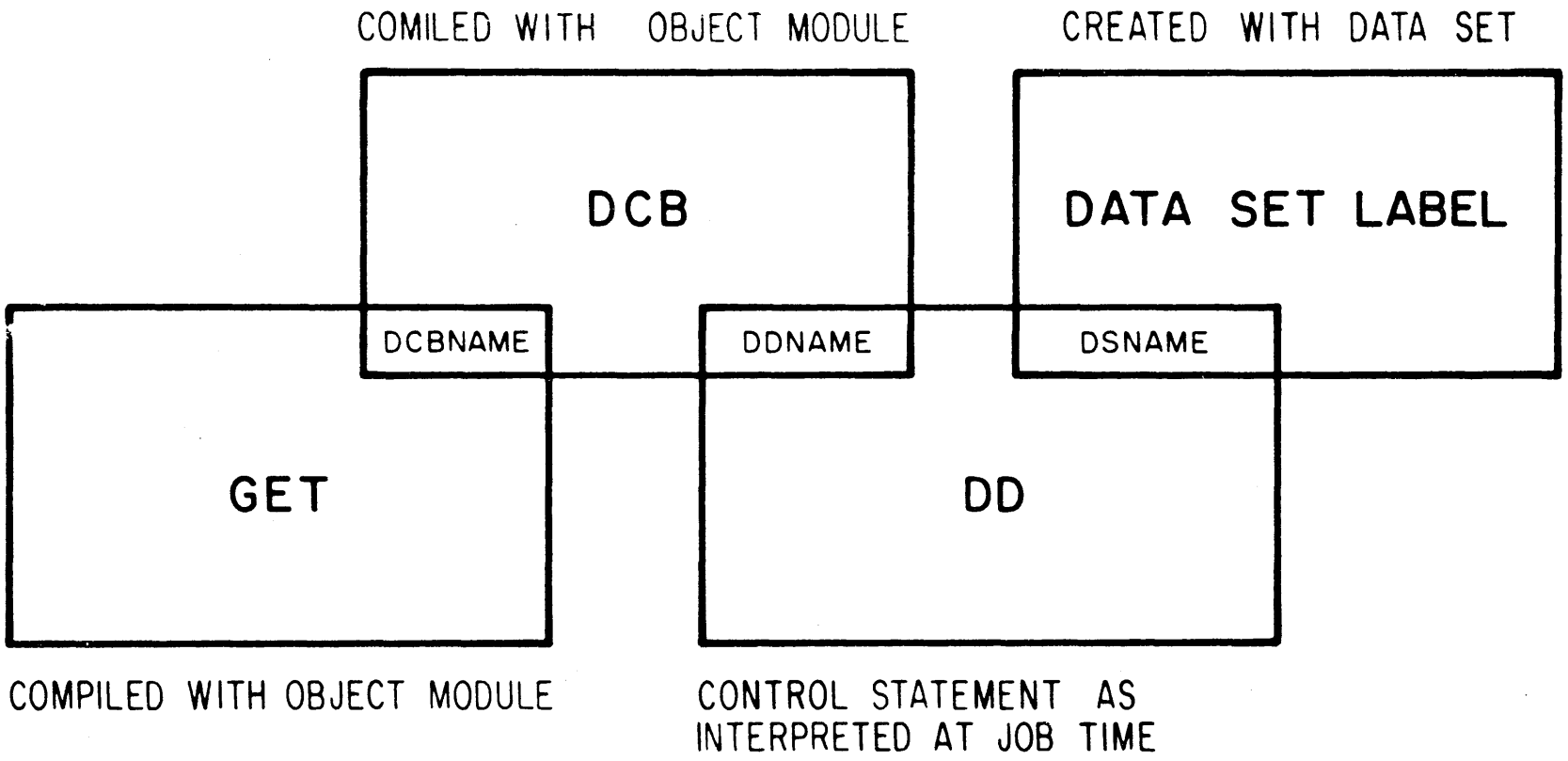
```
//     EXEC   SORT
         :
//SORTIN   DD   UNIT=2400-1,LABEL=(,NL),                    X
//              VOLUME=SER=006042,DISP=OLD,                 X
//              DCB=(DEN=1,TRTCH=ET,                        X
//              BLKSIZE=300,LRECL=75,RECFM=FB)

//SORTOUT  DD   UNIT=2400,LABEL=(,NL),                      X
//              DCB=(BLKSIZE=450,LRECL=75,                  X
//              RECFM=FB),DISP=(,KEEP),                     X
//              DSNAME=NEWFILE
```

INPUT RECORDS ARE ON 9-TRACK, 75 BYTES, BLOCKED 4; DENSITY IS 556 BYTES/INCH.

OUTPUT RECORDS ARE ON 9-TRK TAPE, BLOCKED 6.

CHAIN OF SYMBOLIC REFERENCES



SEQUENTIAL ORGANIZATION*

	CREATE	ADD	RETRIEVE	UPDATE
QSAM	X	X	X	DA ONLY
BSAM	X	X	X	DA ONLY

* APPLICABLE TO ALL DEVICES

1482

PARTITIONED ORGANIZATION*

	CREATE	ADD	RETRIEVE SEQ.
BPAM-BSAM	X	X REWRITE MEMBER	X
BPAM-QSAM (or BSAM) (ONE MEMBER)	X	X REWRITE MEMBER	X

* DIRECT ACCESS DEVICES ONLY NO UPDATE

143

DIRECT ORGANIZATION*

	CREATE	ADD	RETRIEVE	UPDATE
BSAM	X (WRITE-LOAD MODE)			
BDAM		X	X	X

* DIRECT ACCESS DEVICES ONLY

INDEX SEQUENTIAL ORGANIZATION*

	CREATE	ADD	RETRIEVE SEQ.	UPDATE SEQ.	RETRIEVE DIRECT	UPDATE DIRECT
QISAM	X (LOAD MODE)		X (SCAN MODE)	X (SCAN MODE)		
BISAM		X			X	X

* DIRECT ACCESS DEVICES ONLY

ACCESS METHOD SUMMARY

ORGANIZATION	SEQUENTIAL		INDEXED SEQUENTIAL			DIRECT	PARTITIONED
			QISAM				
ACCESS METHOD	QSAM	BSAM	LOAD	SCAN	BISAM	BDAM	BPAM
PRIMARY MACRO-INSTRUCTIONS	GET,PUT PUTX	READ WRITE	PUT	SETL,GET, PUTX	READ WRITE	READ WRITE	READ,WRITE FIND,STOW
SYNCHRONIZATION OF PROGRAM WITH INPUT/OUTPUT DEVICE	AUTOMATIC	CHECK	AUTOMATIC	AUTOMATIC	WAIT	WAIT CHECK	CHECK
RECORD TYPE TRANSMITTED	LOGICAL,F,V BLOCK U	BLOCK F,V,U	LOGICAL F,V	LOGICAL F,V	LOGICAL F,V	BLOCK F,V,U	BLOCK (PART OF A MEMBER) F,V,U
BUFFER CREATION AND CONSTRUCTION	BUILD GETPOOL AUTOMATIC	BUILD GETPOOL AUTOMATIC	BUILD GETPOOL AUTOMATIC	BUILD GETPOOL AUTOMATIC	BUILD GETPOOL AUTOMATIC	BUILD GETPOOL AUTOMATIC	BUILD GETPOOL AUTOMATIC
BUFFER TECHNIQUE	AUTOMATIC: SIMPLE EXCHANGE	GETBUF FREEBUF	AUTOMATIC: SIMPLE	AUTOMATIC: SIMPLE	GETBUF FREEBUF DYNAMIC FREEDBUF	GETBUF FREEBUF DYNAMIC FREEDBUF	GETBUF FREEBUF
TRANSMITTAL MODES (WORK AREA/ BUFFER)	MOVE LOCATE - SUBSTITUTE	LOCATE	MOVE LOCATE	MOVE -	-	-	

SYSTEM CONTROL FUNCTIONS

	DOS/360	OS/360
Device independence	No	Yes
DASD extent information	XTENT cards with absolute addressing	DD cards, DSCB with only space request
Space allocation-primary	XTENT cards at job time	DADSM
Space allocation-secondary	XTENT cards at step time	DADSM, dynamically
Device allocation	Fixed at assembly time, but actual device addresses may be changed by job control cards at execution time	Dynamic allocation at execution time
Cataloging data sets	No	Yes
Cataloged procedures	No	Yes
Systems residence	2311 only	2311, 2314, 2301, 2303
Split systems residence	No	Yes
Libraries	1 source 1 relocatable 1 core image (all on SYSRES)	n source n relocatable n load (on any DASD)
Private libraries	No	Yes
Job control language	Basic, easy to use	Extensive and therefore complicated, but use of cataloged procedures helps
Link editor	Overlay, map	Overlay, map, XREF, mixed load, and object module input
Relocatable programs	Limited to MPS macro utilities and LIOCS	Yes
Overlay	Fetch	CALL SEGLD SEGWT

SYSTEM CONTROL FUNCTIONS (Cont.)

	DOS/360	OS/360
Dynamic program loading	Fetch LOAD	LINK XCTL LOAD ATTACH (option 4 only)
End of program	EOJ	Return
Timing	Time of day in all partitions; interval timer in any one partition at a time	Time of day and interval timer in any partition
Multiple wait	TP only	Yes

DATA MANAGEMENT

	DOS/360	OS/360
Sequential files	Yes	Yes
Partitioned files	No	Yes
Index-sequential files	Fixed records only	Fixed and variable records, automatic record deletions, dynamic buffers
Direct access	Fixed records, undefined records; absolute track only. Extended search (to cyl end only), multivolumes	Fixed, variable, and undefined records; absolute and relative track. Automatic insertions, multivolumes, extended search (to cyl end), dynamic buffers
BTAM	No burst mode devices on multiplex channel with TP devices. Official support for 7770/72 and 2780. Support for 2260 local and remote	MFT and MVT only. No support for 7770/72, 2780, or 2260 local
QTAM	With multiprogramming option only	MFT and MVT only
Graphics	No	Yes, 2260 and 2250

DATA MANAGEMENT (Cont.)

	DOS/360	OS/360
Data set specification	DTF only	DCB, DD, label information
User standard labels	Sequential: header and trailer Direct: header index-sequential: none	All organizations supported mid '67
Nonstandard labels	n program routines	1 system routine
Level of support	Queued (update, move, locate) Basic (index-sequential, direct, and work files) EXCP	Queued (update, locate) Basic (all access methods) EXCP
Buffering	1 or 2 static (GETBUF, FREEBUF in BTAM only)	n dynamic (OPEN, GETPOOL, BUILD, GETBUF, FREEBUF macros)

LANGUAGES AND SYSTEM COMPONENT SIZES

	DOS/360	Equivalent OS/360	High Performance OS/360
Assembler	10K	18K	44K
COBOL	14K	17K	80K
FORTRAN	10K	16K	128K (G) 200K (H)
PL/1	10K		44K
SORT	10K		17K
RPG	10K	15K	
Link-edit	10K	15K	18K 44K 88K
Scheduler	10K	18K	44K 100K
Supervisor + other resident routines for workable system	6K-10K	PCP 14-20K MFT: 26-36K	MVT: 100K and up
Minimum partition sizes	10K background 2K foreground 1 2K foreground 2	PCP: 18K MFT: 18K per partition	MVT: 44K per partition

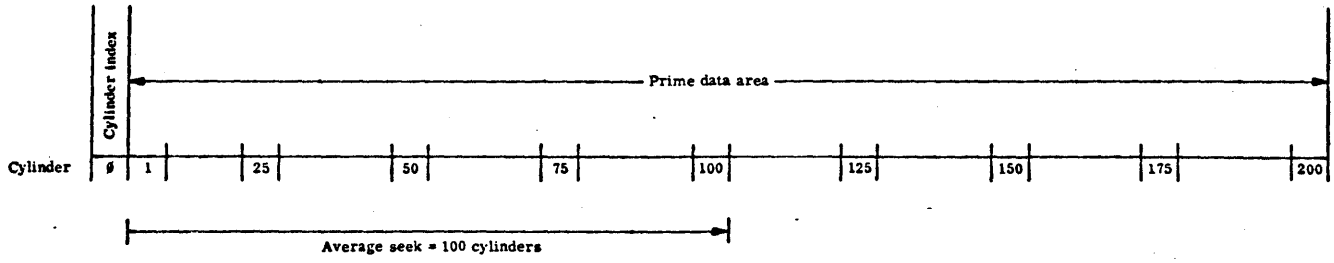
File Organization	DOS/360	OS/360
Index Seq:		
Location master index	Contiguous with cylinder index	Any place on any DASD device
Highest-level index with random processing	Called in from DASD device	May reside in core
Prime data area	Must be single, contiguous extent within volume	Can be multiple extents within volume
Record deletion	User responsibility	Automatic deletion of tagged records
Direct:		
Multiple track search	Limited to cylinder boundary	Can search complete data set
Relative track and block addressing	No	Yes
Track overflow	No	Yes
Partitioned Data Sets	No	Yes

	DOS/360	OS/360-MFT
Maximum I/O Areas:		
Seq. file	2 + work	No limit
Index seq. direct	1 + work	No limit
Buffering Techniques:		
Simple	Yes	Yes
Exchange	No	Seq. files only with restrictions
Buffer Generation	Compile time	Compile, or dynamically by OPEN
File I/O Characteristics:	DTF	DCB
Assigned at -	Compile time	Compile or execution time
Device independence	No	Yes - within devices using same file organization
Data Set Concatenation	No	Yes

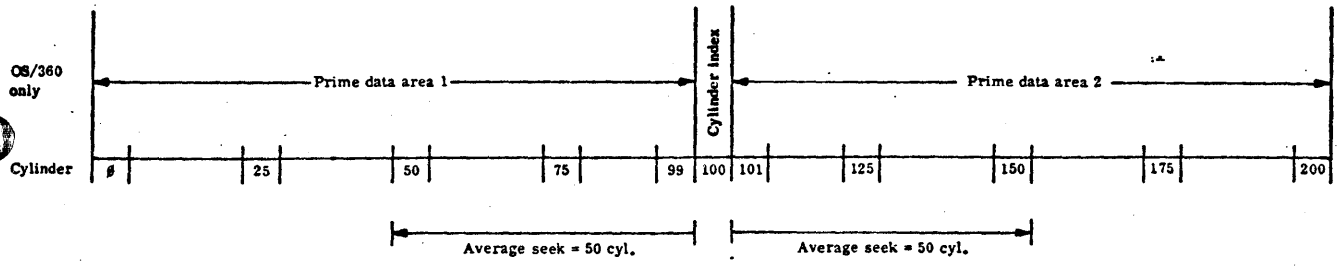
	Sequential			Indexed-Sequential		Direct	Telecommunications		Graphics
	Read/write	Get/Put	Read/write (partitioned)	Read/write	Get/Put	Read/write	Read/write	Get/Put	Basic
DOS/360	SAM	SAM	none	ISAM	ISAM	DAM	BTAM	QTAM	none
OS/360	BSAM	QSAM	BPAM	BISAM	QISAM	BDAM	BTAM	QTAM	GAM

	BSAM*		QSAM		BPAM*		BISAM		QISAM		BDAM*	
Format F - unblocked	DOS	OS	DOS	OS		OS	DOS	OS	DOS	OS	DOS	OS
Format F - blocked	DOS	OS	DOS	OS		OS	DOS	OS	DOS	OS	DOS	OS
Format V - unblocked	DOS	OS	DOS	OS		OS		OS		OS		OS
Format V - blocked	DOS	OS	DOS	OS		OS		OS		OS		OS
Format U	DOS	OS	DOS	OS		OS					DOS	OS

* Deblocking and blocking of logical records must be done by problem program.



Example 2: OS/360 split data area



A Batch Processing FORTRAN System for a Minimal Configuration IBM 1620

This paper describes a software package designed to increase throughput on a 20K card system IBM 1620 computer in an environment where a large number of fairly simple FORTRAN programs must be processed. The increase in throughput is accomplished in two ways; (1) programs are batch executed under control of a loader-monitor routine, and (2) a powerful pre-compiler reduces the number of execution errors and decreases the requirement for on-line debugging.

The compiler is a version of the PDQ FORTRAN compiler which has been modified to handle the batch processing features of the system. Batch execution is made possible by keeping the entire subroutine library resident in core rather than reloading it with each object deck. Object programs, separated by control cards, are stacked for input. The reading of a control card by the FORTRAN card read subroutine or the execution of a CALL EXIT statement will cause the next object deck to be automatically loaded and executed. The monitor will also terminate a job if the output line count exceeds a control card specification.

The precompiler detects more than seventy distinct errors based on PDQ FORTRAN specifications. Many of these errors, such as undefined symbols, are undetected by the compiler and cause execution check-stops. Recognition of multiple errors in a single statement is possible, thus eliminating multiple debugging passes.

1620 answer M.4.5 a+b

WILLIAM G. LANE
CHICO STATE COLLEGE
CHICO, CALIFORNIA

DEAR MR. LANE,

PLEASE EXCUSE THE USE OF THIS TYPING FOR A LETTER, BUT I CAN PUNCH CARDS FASTER AND MORE ACCURATELY THAN I CAN TYPE. ALSO, I DO NOT HAVE A SECRETARY AT THE PRESENT TIME.

MY MAIN PURPOSE IN WRITING TO YOU IS TO ASK IF THERE IS A SLOT IN THE 'COMMON' MEETING FOR TWO PAPERS, ONE OF WHICH I MENTIONED TO YOU IN SEPT. THESE ARE THE SAME TWO PAPERS I GAVE IN SEPT., BUT I THINK THERE WILL BE ENOUGH PEOPLE WHO COULD NOT COME TO THE MEETING IN SEPT. TO BE WORTHWHILE FOR ANOTHER PRESENTATION.

I HAVE TWO PAPERS, ONE SPECIFICALLY FOR THE 1620, AND ANOTHER OF A MORE GENERAL NATURE. THE PAPERS SHOULD NOT NEED MORE THAN ABOUT 10 - 15 MINUTES. THE ABSTRACTS ARE -

1) 'A LARGE 1620 DISK OPERATING SYSTEM, REASONABLY 7094 COMPATIBLE.'

A NEW MONITOR FOR A 40K OR 60K 1620 HAS BEEN DEVELOPED AT PRINCETON FOR USE AS A DEBUGGING AID FOR THE 7094 AND SOME 360/OS PROGRAMS. FN II I/O AND FN IV I/O ARE INCLUDED AS WELL AS PRIVATE STORAGE OF PROGRAMS. MANY CONTROL CARD OPTIONS EXIST. SPEED CAN BE 2 TO 4 TIMES FASTER THAN MON-I (IF ONE HAS HARDWARE FLT. PT.) BOTH IN COMPILATION AND EXECUTION. A USERS MANUAL DOES EXIST. MANY STUDENTS ARE PRESENTLY USING THE SYSTEM ON A COMPLETELY OPEN-SHOP BASIS.

2) 'BASIC PROBLEMS OF INFORMATION RETRIEVAL AND A SOLUTION ON THE 1620.'

SOME OF THE BASIC PROBLEMS OF INFORMATION RETRIEVAL AND IMPLEMENTATION ARE DISCUSSED. THE TECHNIQUES OF FILE-ORGANIZATION ARE DISCUSSED. THESE TECHNIQUES ARE DISCUSSED FOR THE MOST COMMON FILE DEVICES (DISK AND TAPE). A SOLUTION IS SHOWN FOR THE 1620 WITH DISK TO DEMONSTRATE DISK UTILITY.

I AM SORRY ABOUT THE LATENESS OF THIS, BUT I HAVE BEEN OUT OF TOWN FOR SOME TIME AND CAN ONLY GET BACK FOR WEEK-ENDS AT THIS TIME. I HOPE TO SEE YOU IN S.F.

VERY TRULY YOURS,

Lanny Hoffman

LANNY HOFFMAN
COMPUTING GROUP
GUGGENHEIM LABS
PRINCETON, N.J.
08540

RECEIVED
NOV 1 1967
C.S.C. COMPUTER CENTER

1800/1896 COMMUNICATIONS ADAPTER
PROGRAMMING SUPPORT

Presented at COMMON, December 11, 1967
San Francisco, California

by Robert L. Smith
Systems Engineer
IBM Corporation
1602 West Third Avenue
Flint, Michigan 48504

Phone:313-235-6631

TABLE OF CONTENTS

Section	Title	Page
I	Introduction	1
II	Terminal Descriptions	3
III	Hardware Interface	6
IV	General Description of System	7
V	Line Operation Requests	10
VI	Data Table Layout	20
VII	Skeleton Core Requirements	21

DIAGRAMS

Diagram	Title
1.	1070 System
2.	1070 Data Flow
3.	1050 System and Data Flow
4.	2740 System and Data Flow
5.	Data Table Layout
6.	CA Hardware Interface
7.	Programming System Data Flow

I. INTRODUCTION

This paper describes the program written to provide programming support for the 1896 Communication Adapter used to attach start/stop terminals such as the IBM 1070,1050, 1030, and System/360 via 2701, 2702, or 2703 Communication Adapters to the 1800 control system. The initial 1896 support was written as a joint effort between the Flint, Michigan branch office and the Chicago North branch office to support 1800/1070 systems. The project was started in February of 1966 and we were able to run 1070's, though somewhat erratically, by November of that year. An 1800/1070 Type III package running under TSX II was submitted a few weeks ago to PID. The authors are Norm Rawson and Chuck Reiling of the chicago office, and myself of the Flint office. Implementation has taken roughly five man-years of time, and five years off each of our lives. You are recommended to the paper being presented by Charles Jonas of The Dow Chemical Company to hear the other side of the story, the user's.

The original commitment to the customer specified that the 1896 Support would be accessed via FORTRAN and run under the 1800 Time Shared Exceutive (TSX), with little or no loss of the capabilities of either the 1800 or TSX.

We also required ourselves to consider the following three points:

1. The 1800 Control System is at least three magnitudes of speed faster than the terminals, implying that we not hold the 1800 to wait for a communication function to be performed. This requirement was reinforced by point 2 below.

2. An 1800/1896/1070 System may be controlling more than one process, including processes interfaced through the 1800 Process I/O features.
3. The resulting system must not use more than 3000 words of memory (Skeleton resident).

To fulfill these requirements, the system was designed to meet the following specifications:

A. All user programming may be done in FORTRAN.
B. All requests for communication functions will be placed in a priority queue.
C. Upon completion of a requested function, a user specified flag will be set and, if the option is included, a mainline coreload may be placed on the TSX coreload queue (priority and coreload specified for each function requested).
D. Printer output may be done via FORTRAN WRITE statements.
E. The support should take less than 3,000 words of memory (i,e,, run on a 16K 1800).
F. Multiple terminals per communication line, and multiple lines per 1800 will be supported.
G. TSX will be altered the minimum amount possible so that TSX maintenance by FED will not be sacrificed more than necessary.

The above specifications have been met. The use of the FORTRAN WRITE statement for 1053 output required the modification of either FORTRAN I/O or Skeleton I/O; we chose what we felt was the lesser of two evils, and modified TYPEN in Skeleton I/O.

II. TERMINAL DESCRIPTIONS

The following brief descriptions of the various Start/Stop Terminals available are meant to present the operating differences, and the capability of each pertinent to the discussion that follows. For more detailed information, please see the appropriate reference manual.

IBM 1070 Process Communication System (Form No. A26-5989)

The IBM 1070 system is a tele-processing system designed to monitor, supervise, and control widely spread process areas. Two-way transmission between a remote 1070 System and a central station is over standard voice or sub-voice grade communication lines. A 1070 System consists of a 1071 Terminal Control Unit, 1 or more 1072 Terminal Multiplexors, plus options to allow analog input, digital input, digital output, pulse counters, pulse output, 1053 Printers, manual entry stations, and display units. Transmission speed is selectable from 134.5, 600, or 1200 baud. See Diagram 1 for a schematic. Diagram 2 gives some examples of character exchanges necessary to perform certain functions. Analog Input is received as a series of three digits for each point scanned, representing a count between 0 and 7999 proportional to the signal attached.

The 1070 Process Communication System will recognize 5 function codes:

0 - Conditional take from address 0

5 - Unconditional take from address 0

6 - Conditional take from present address

7 - Unconditional take from present address

9 - Send

In addition, the 1070 can be set to random or sequential operating modes, and, on send, the printer output channel may be selected.

Terminal Addressing is a sequence of 3 characters transmitted from the 1800 that specifies the terminal and function wanted. The three characters and their use are:

1. EOT - © - Calls to attention all terminals on that line.
2. A - The address (one alpha character) of the terminal that is wanted.
3. N - The function (one BCD digit) that is requested.

The 1070 Character Coding consists of bits BA8421. The 1071 Terminal Control Unit contains a 3-Digit Multiplexor Address Register. Random or sequential operating mode is selected by the presence or absence of the B-Bit in the hundreds digit. Printer Output Channel is selected by a B-Bit present in the unit's digit of this register. This register may be set by

a short SEND, i.e., ⓒA 9..ⒹA $M_1M_2M_3$Ⓑ, where ".." represents a positive response from the 1070. M_1 is the hundreds digit, M_2 the tens digit, and M_3 the units digit that are placed in the 1071 Multiplexor Address Register. The B-Bits are placed in the correct digit by the program before this message is transmitted.

1050 Data Communication System (Form No. A24-3474)

The 1050 system is a tele-processing system designed to send and receive data in the form of cards, paper tape or manually. A system will contain a 1051 Control unit, plus any or all of the following units: Card Reader, Card Punch, Paper Tape Reader, Paper Tape Punch, Programmed Numerical Keyboard, Selectric Printers, and an Alphanumeric Keyboard. Diagram 3 shows a typical configuration and data exchanges with a central computer.

The 1050 receives and transmits on standard voice or sub-voice grade communication lines at a rate of 134.5 baud. Addressing the terminal is done by a two character sequence consisting of a terminal address character and a device selection code. The device selection code also specifies whether data is to be sent or received. Multiple blocks of data may be sent or received without relinquishing control of the communication line.

2740 Communications Terminal (Form No. A24-3403)

The IBM 2740 Terminal is a tele-processing Selectric typewriter. As such it has the capability of sending data inserted via the standard typewriter keyboard, or receiving data as a printer. The 2740 communicates with a central processor via standard voice or sub-voice grade communication lines. Diagram 4 illustrates the character exchanges necessary to communicate with a 2740. It is highly recommended that 2740 model 2's, with the buffer attachment be utilized for most efficient operation. 2740's normally operate at 134.5 baud, but with the buffer attachment, 600 baud rates can be used.

Note that all three terminals use a slightly different adressing scheme and that all three utilize fill characters (Ⓣ) to allow time for mechanical movements on carriage returns, tabulates, and warm up periods. On the 1070, filler characters are used on the 600 baud model to present output characters to the printers at a rate less than 14.8 characters per scond.

III. HARDWARE INTERFACE

The 1896 Communications Adapter interfaces to the 1800 via Process Interrupt - Voltage, and Electronic Contact Operate. Each separate communication line attached requires a full group of each. See Diagram 5 for a schematic of the interface. The programming system assumes interrupt assignments to level 0. The reasoning here is that with the exception of the 1816 Keyboard, this is the only device that has the possibility of data overrun without hardware failure.

Each group of process interrupt is wired into one bit of the level 0 Interrupt Level Status Word, beginning with bit 0 and proceeding through bit 15. In this manner an inherent priority is established for each line. However, once the servicing of a line has begun, it will proceed through to the BOSC instruction before any other lines are recognized. This takes somewhere around 750 usec. on a 4 usec. CPU.

Except for the fact that the level 0 exit from MIC has been modified, there is no other good reason to restrict level zero to just the communications adapter, although not too many people will accept responsibility if that restriction is not honored.

IV. GENERAL DESCRIPTION OF SYSTEM

The system may be divided into four logical areas:

1. Subroutine LINOP
2. CAISS -- A subroutine containing
   a. Supervisory Routines
   b. ISS - Interrupt Servicing Subroutines
3. 1053 Support
4. CATST - The error coreload.

Each of these areas is described briefly below.

Subroutine LINOP

This subroutine, callable from FORTRAN, is the user's means of requesting a communication line operation. The user has a choice of having an indicator set for him at the completion of

the requested function (LINOP), or he may have the system execute a CALL QUEUE (LINWQ), placing a designated mainline coreload on the TSX queue in addition to setting the indicator. LINOP, which is used here to designate both calls, contains a line-operation-request queue which the Supervisor may interrogate, plus the necessary queue maintenance routines.

CAISS Subroutine

This subroutine is defined as an ISS Subroutine, allowing the definition of a separate entry point for each ILSW bit used, plus a CALL or LIBF entry point. The ILSW bit entry points are entries to the ISS portion of CAISS. The CALL entry point is defined at Skeleton Build as the servicing subroutine for a Program Setable Interrupt that is assigned to the Supervisor portion of CAISS. Brief description of the Supervisor and ISS portions of CAISS follow:

Supervisor

The Supervisor is not a separate program or subroutine as in LINOP, but is a set of routines, contained within the CAISS package, to provide services to the Interrupt Servicing Subroutine (ISS). The Supervisor is entered by a CALL LEVEL, said level being below that of the ISS itself. This allows the Supervisor to perform its tasks without interferring with the servicing of 1896 CA interrupts. The Supervisor builds

line operations for CAISS from the user's queue entry, does data-table chaining, halts typing on 1070's to allow scanning, removes completed line-operation requests from the queue, saves error information for the error coreload, and in general acts as an assistant to CAISS and an intermediary between the user and CAISS.

ISS

These routines service the interrupts of the 1896 Communication Adapter, sending and receiving data to and from the terminals. To TSX it appears as a special I/O Subroutine.

1053 Support

The 1053 Support consists of modification to TYPEN in TSX, so we may intercept typewriter messages just prior to their output to non-existent 1053's attached to the 1800. These messages are placed on the disc, since the non-existent 1053's are defined as always being busy.

When a line is free and has no queue entries waiting, CAISS initiates a special TYPEN function to recall these messages from disk. These messages are then converted to BCD code, fill character (T) counts are inserted, and a CALL LINOP is performed. The 1896 Support System thereafter handles this entry as any other.

An additional feature called Type-Stop was put in to allow the 1070 users to halt the typing of a 1053 message, perform control functions and/or input scanning, then resume the typing where it was interrupted.

CATST - The Error Coreload

Errors are handled in a rather elementary fashion to get around core limitations. When an error is detected, the contents of the work area for that line are transferred to an error buffer, and the whole operation is retried. After a set number of retries, the operation is aborted and terminal is marked down. Any other queue entries for that terminal are attempted and if any are successful, the terminal is set back up. The abortion signals for the queueing of the error coreload - priority and error parameter of 1. This coreload prints out a message on a designated unit, can dump the error buffer and other information on the list printer, keeps counters updated, and tests terminals that are marked down by the simple method of addressing them. If the terminal answers, it gets put back up on line. The error coreload is also queued up periodically by CAISS for counter update and terminal tests.

Design Consideration

The structure of the support is the result of trying to satisfy the considerations mentioned above in Section I, plus a few factors which were not known at the initial specification. These factors were:

a. The specific hardware design of the 1896 Communication Adapter and its interface to the 1800.

b. The need to service at least six 1200 baud lines. This speed means there is only 7.5 milliseconds between characters on a line and 1/6th of that allows us only 1.25 m/sec. per line for a maximum service time on the CA interrupt level before we start to lose data. On a 4 microsecond 1800, this is approximately 65 instructions, using San Jose's estimate of 20 usec/instruction. For this reason, a high percentage (90+) of the coding is in one-word, indexed instructions, which run approximately 10 usec/instruction, allowing us about 120 instructions (which merited a minor sigh of relief).

Because of this time limitation, we have, wherever possible, delegated jobs to the Supervisor, letting the 1896 - CA idle if necessary.

IV. LINE OPERATION REQUESTS

Functions:

All line operations are initiated from CALL statements to a skeleton resident line operation routine. The following line functions will be provided:

1070 Function Codes:

0 - Conditional TAKE from zero multiplexer address

5 - Unconditional TAKE from zero multiplexer address

6 - Conditional TAKE from user specified multiplexer address

7 - Unconditional TAKE from user specified multiplexer address

9 - SEND to user specified multiplexer address

11 - SEND to 1053 Printer

A terminal multiplexer may be set to either random or sequential operating mode, and the 1053 Printer Output Channel may be selected on a direct send (specify cuntion Code 11).

1050 Functions (Device Selection Codes)

Polling (Take from 1050)	Sending to the 1050
0 - Any input component	1 - Printer 1
5 - Keyboard	2 - Printer 2
6 - Reader 1	3 - Punch 1
7 - Reader 2	4 - Punch 2
	9 - Any or all output components that are in a steady status

2740 Function Codes

Being a relatively simple device, the 2740 functions are relatively simple. A function of 0 (zero) requests data from the 2740 keyboard, its only input device, while any non-zero value will send data to the 2740 printer. Period.

Call Statements:

Three line operation subroutines are provided:

a. LINOP (line operation) which maintains an indicator for the line operation status.

b. LINWQ (line operation with queue) which queues a mainline coreload when the line operation is complete and maintains the operation status indicator.

c. TYPSP (type stop) which interrupts an 1896 message of priority 255 in favor of a more important operation on that line, and then comples the message when the line is again available.

Subroutine LINOP is called either by one of the following call sequences

1. FORTRAN: CALL LINOP (LF, LP, LL, LT, NDATA, IND)

2. ASSEMBLER:

```
CALL LINOP
DC   LF
DC   LP
DC   LL
DC   LT
DC   NDATA
DC   IND
```

Note: FORTRAN ARGUMENTS LF, LP, LL, LT, LTSXQ. may be integer constant or variables, other arguments are integer variables. The assembler constants are the addresses of the arguments.

Subroutine LINWQ is called by either of the following call sequences:

1. FORTRAN: CALL LINWQ (LF, LP, LL, LT, LDATA, IND, LTSXQ, QUEUE, CLDNM)

   NOTE: Program names QUEUE and 'CLDNM' must appear in an EXTERNAL statement in the calling progam.

2. ASSEMBLER:

```
CALL    LINWQ
DC      LF
DC      LP
DC      LL
DC      LT
DC      NDATA
DC      IND
DC      LTSXQ
CALL    QUEUE
CALL    CLDNM
```

Subroutine TYPSP is called by either of the following call sequences:

1. FORTRAN CALL TYPSP (LL)
2. ASSEMBLER:

```
CALL TYPSP
DC      LL Address of the line number
```

Definition of parameters for LINOP, LINWQ, and TYPSP:

a. LF - Function code, 0 - F_{16} (0 - 15_{10}) stored as a one-word integer, only the low order of 4 bits are used.

b. LP - Priority to be assigned to this line operation by the 1896 Line Control Supervisor. LP is a one-word integer, between 0 and 254. LINOP uses the 8 low order bits as a positive number. Zero (0) is considered the highest priority and 254 the lowest. FORTRAT 'WRITE' statements to a 1053 are automatically given priority 255.

c. LL - Line number, a one-word integer between 0 and 15, only the low order 4 bits are used. This number references the Communication Adapter (CA) attached to that ILSW bit specified by LL. The 1800 Interrupt handling technique implies an order or priority with ILSW bit being the highest if two or more interrupts are simultaneous on the same level.

d. LT - Terminal on the specified line. LT can range from 1 to 16 with LINOP or LINWQ converting to the alpha addressing character A-P. Terminals on a line are required to be assigned starting with A and proceeding consecutively. Neither LL nor LT may reference a higher numbered line or terminal than has been assigned at system generation time or a LINOP error will result.

e. NDATA - This is the low core (FORTRAN high subscript) address of user's data table. Data table format is explained below.

f. IND - is an integer indicator which will be set at various stages of the line operation. The value IND is set to depend on the status of the line operation.

IND = 1 - Line queue was full, unable to enter request

IND = 2 - Request entered in line queue successfully

IND = 3 - Line operation complete with no errors (CLDNM has been queued)

IND = 4 - CALL paramenter in error, request not entered

IND = 5 - Repeated line operation failure, request cancelled (CLDNM has been queued)

IND = 6 - Line or terminal down (inoperative), request canc.

The user may elect to have the real-time clock returned with the operation complete (3 or 5) indicator. In this option, user must dimension the indicator as two adjacent words (as DATA IND, IND2/0,0/). Indicator value will be returned in IND and clock in IND2, the next lower core address. Format of CALL LINOP is not changed.

g. LTSXQ - This integer is the priority used if the user wishes the 1896 Line Control Supervisor to issue a CALL QUEUE at the completion of the line operation. LTSXQ may have any value from 1 to 32767. It cannot be zero.

h. QUEUE - A dummy parameter which indicates the following argument is a coreload name.

i. CLDNM - is the name of a core load to be used by CAISS in a CALL QUEUE (CLDNM, LTSXQ,X). (X is assigned by the user at system generation and determines if the lowest priority entry is replaced or the queuing is ignored when a queue overflow occurs.) Note that the coreload is queued on an unsuccessful line operation

(IND indicator of 5), as well as on a successful operation (IND indicator at 3).

The type stop call is provided for rapid recognition of interrupts. It will suspend typing in favor of more important operations on a line. CALL TYPSP will stop typing at completion of the current character and will allow the line queue to be scanned for other line requests. When no more entries remain in the line queue, typing will be continued at the point of its interruption. This call should be issued from any interrupt servicing subroutine that requires line access in less time than the maximum typing requirement. For example, a 600 baud line will require in excess of ten seconds to complete an 130 character message.

TYPEWRITER OUTPUT

The 1800/1896 Process Communication and Control System provides two methods of typewriter output to the 1053:

1. Normal FORTRAN 'WRITE' operations with a 'FORMAT' statement.
2. Direct line operations CALLED by the user.

FORTRAN 'WRITE' STATEMENTS:

The TASK program of TSX has been modified to handle output to 1053 typewriters on the 1896 system from the normal FORTRAN I/O subroutines. Typewriters are assigned logical unit numbers for FORTRAN when the TSX system is built by the user, and the user initiates a message with the 'WRITE' and 'FORMAT' statements. In TSX, all FORTRAN typewriter messages or message units are buffered to disk, occupying one sector per message unit. Control is returned to

the user program immediately after buffering. Buffered 1053 messages are recalled by the RECAL subprogram whenever their 1896 line is idle; therefore, typewriter messages have the lowest priority of all line operations, (priority 225). The length of time between 'WRITE' statement and message typing is dependent upon the number of requests for that line which are in the line-queue. Messages are recalled from the buffer on a first-in, first-out basis; alert messages will be recalled before normal messages. In TSX, alert messages are recognized by the first character in the 'FORMAT' statement being a "Ribbon Shift Up". If a typewriter, or its line, is down, a backup typewriter specified by the user can be used. Error conditions such as disk buffer overflow are handled by a modified TYPEN program of TSX. A maximum of 16 typewriters, one of which must be an 1800/1053 typewriter, may be supported by the 1800/1896 modified TSX system.

Because both 'Printer Start' and 'Carriage Return-Line Feed' operations are very time consuming, the user may take the option to permanently wire 'ON' his 1053 typewriters and to have the FORTRAN generated CRLF at the beginning of each typewriter message, automatically deleted upon recall. With this option, message units may not be used, and it is the user's responsibility to place CRLF where needed. If the option is not taken at system generation time, a printer start character is automatically placed before the message unit or CRLF of the message when the message is recalled from disk buffer.

Direct Line Messages:

An emergency 1053 message may be sent with priority over other operations on that 1896 line by using CALL LINOP or LINWQ and a data table containing the message. Function code is an 11 in the user's CALL, conversion code is 4, and random mode bit (ICM in data table) must be on. Data format is valid 1070 character in left 8 bits and filler count in right 8 bits of each data word.

SEQUENCE OF OPERATIONS

The following sequence of operations is initiated by the user with CALL LINOP or LINWQ:

1. User CALL from normal or interrupt program.
2. a. Request processing by LINOP on same level as user, set user indicator.
   b. Enter request in line-queue.
   c. Set programmed interrupt for CAISS supervisor.
3. a. Recognition of programmed interrupt.
   b. Search line-queue for highest priority request.
   c. Build line-operation tables for terminal and multiplexer addressing.
   d. Trigger first interrupt from Communications Adapter.
4. a. Level 0 interrupt for CAISS
   b. Device addressing.
   c. Data transmit and receive.
   d. Longitudinal redundancy check.
   e. Clear line and set programmed interrupt for CAISS supervisor.
5. a. Recognition of programmed interrupt.

b. Set line not busy, set user indicator.

c. Call diagnostic coreload if error has occurred.

6. User program recognizes operation complete code in user indicator.

DATA TABLES (Diagram 5)

The data tables required by the 1896 system are modeled after the normal 1800/1130 data tables. The address transmitted in the call sequence points to the low core end of the table, requiring in FORTRAM the high subscripted end. The first word in the data table is a word count specifying the number of words of data or the number of characters to be transmitted or received, depending on the conversion code. The high order bit is the chain indicator bit if this option is included. If the chaining option is not included, this bits presence will be ignored. The system uses the low order 14 bits as a positive integer.

The second word specifies the conversion code in the high order 4 bits, and for 1070 systems, the multiplexor address and mode in the low order 12 bits. The conversion codes available are:

0 - Unpacked, one character per word.

1 - Packed Digital, two characters per word.

2 - Unpacked Digital, one character per word.

3 - Analog Input (1070 only)--the converted binary value of one analog input point is stored in each word.

4 - 1053 output, the character to be printed is in the high order 8 bits, the number of filler characters to follow is in the low order 8 bits.

5 - Packed, two characters per word.

The difference in digital or normal conversion is the handling of the BCD zero, consisting of the 82 bits, which is converted to a binary 0 in Digital conversion, but left as a binary A (10_{10}) in normal mode; plus that Digital conversion concerns itself with only the four low order bits and ignores any higher order bits.

The Chain Address word must be present and set correctly if chaining of data table is used. The contents of this word is used with no checking as the address of the next data table. Bad things could come of an incorrect setting here. Note that in MPX, a load address function is provided to allow setting of this chain address that is remarkable similar to the one provided by the 1070 support package.

SKELETON CORE REQUIREMENTS

Approximate core requirements for the skeleton resident portions are given below. These core requirements are approximate and I will not be held responsible for them.

TASK modifications for Typewriter support via FORTRAN I/O: 620 Words.

Subroutine LINOP/LINWQ plus the queue: 330 Words.

The type stop subroutine TYPSP: 30 Words.

CAISS Subroutine: 1200 Words plus 100 each for the chain on Send and Type-Stop Options plus 40 Words per line attached.

The sum is about 2240 words for a reasonable system.

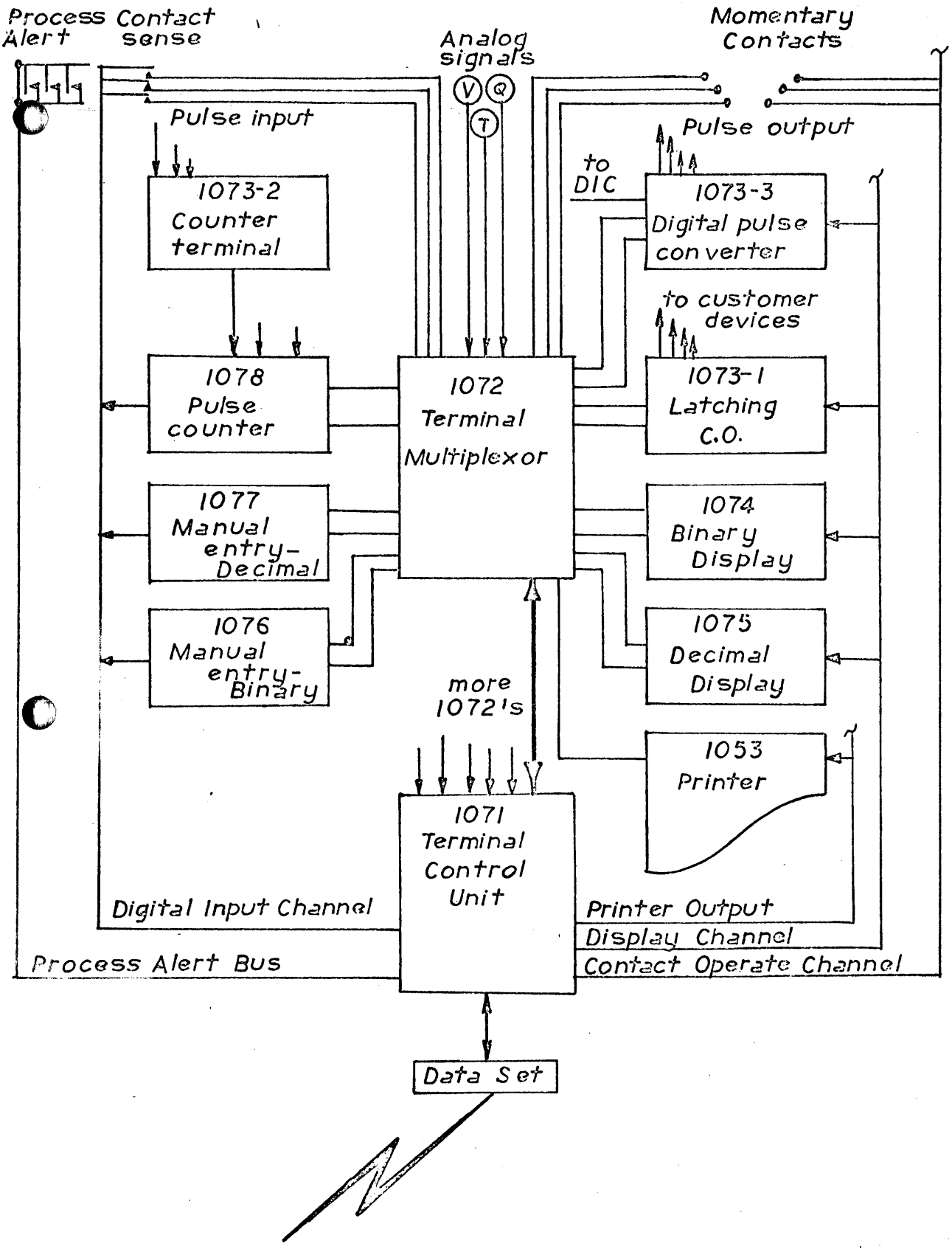
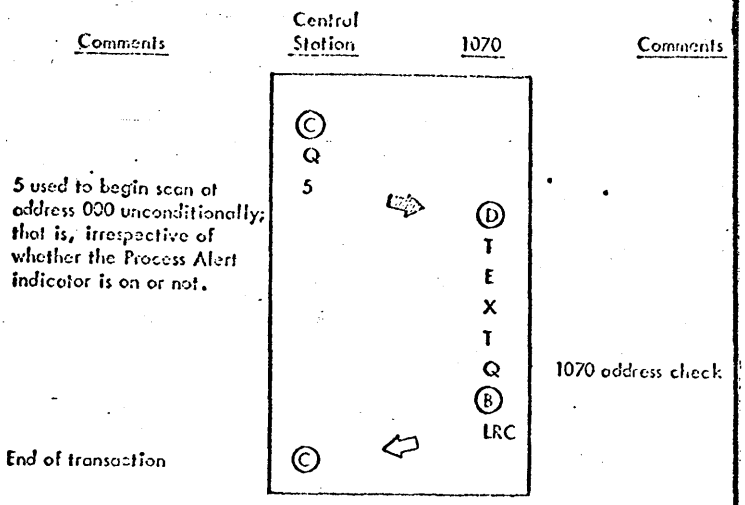
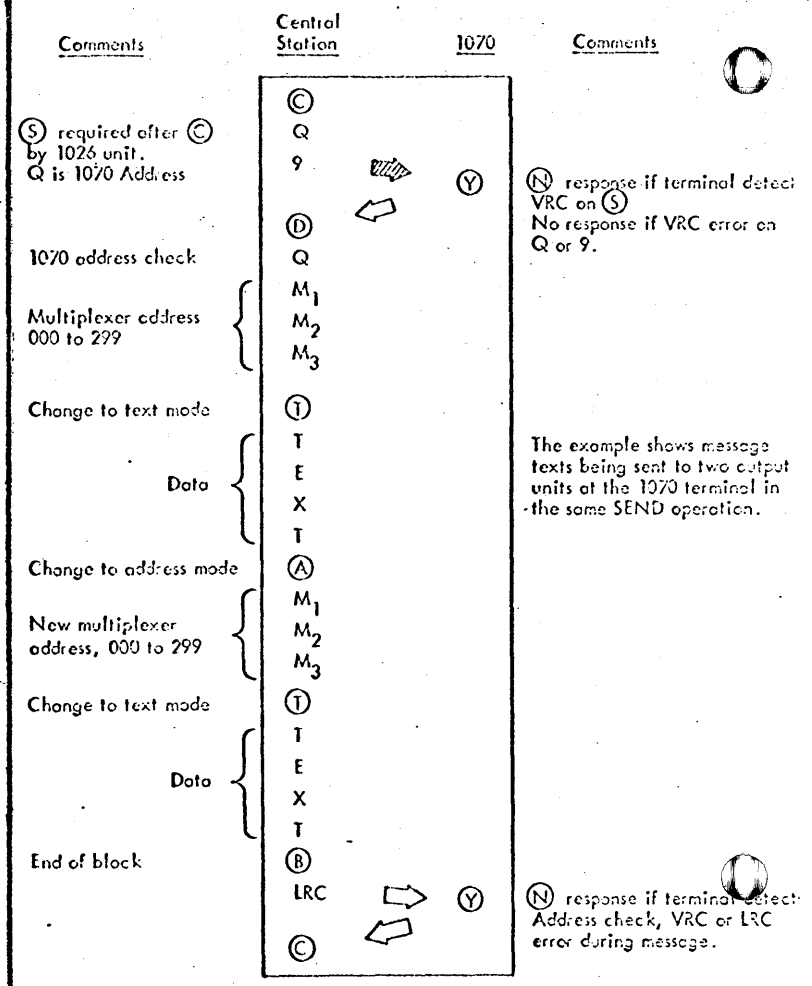


DIAGRAM I: 1070 SYSTEM

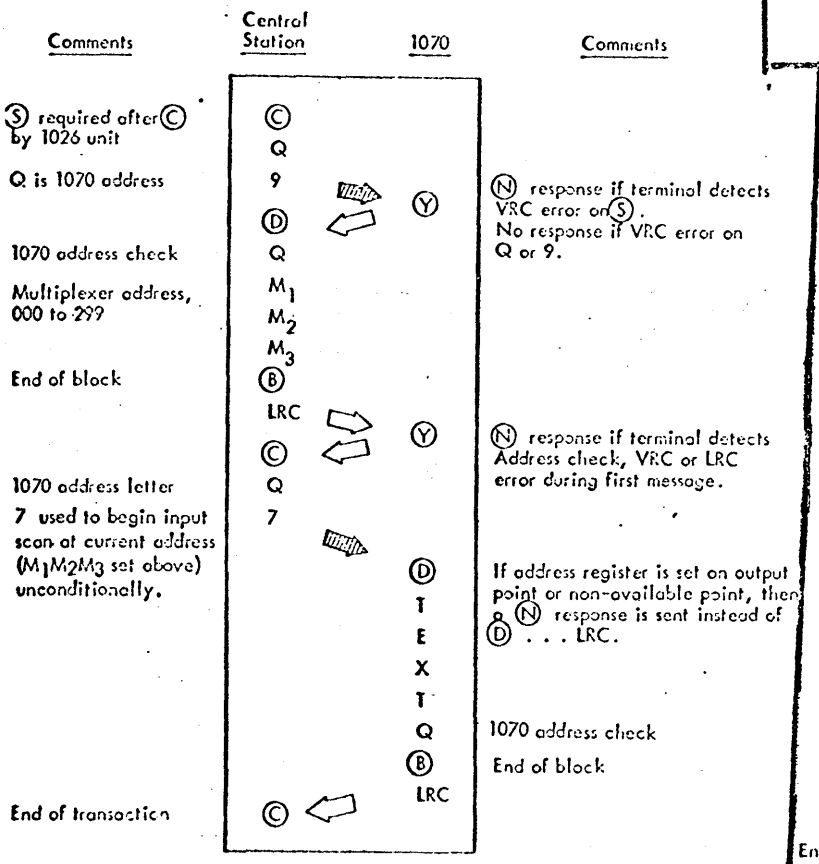
UNCONDITIONAL TAKE FROM MULTIPLEXOR ADDRESS 000



SEND TO THE 1070



ADDRESSED TAKE FROM 1070



CONDITIONAL POLLING OF 1070

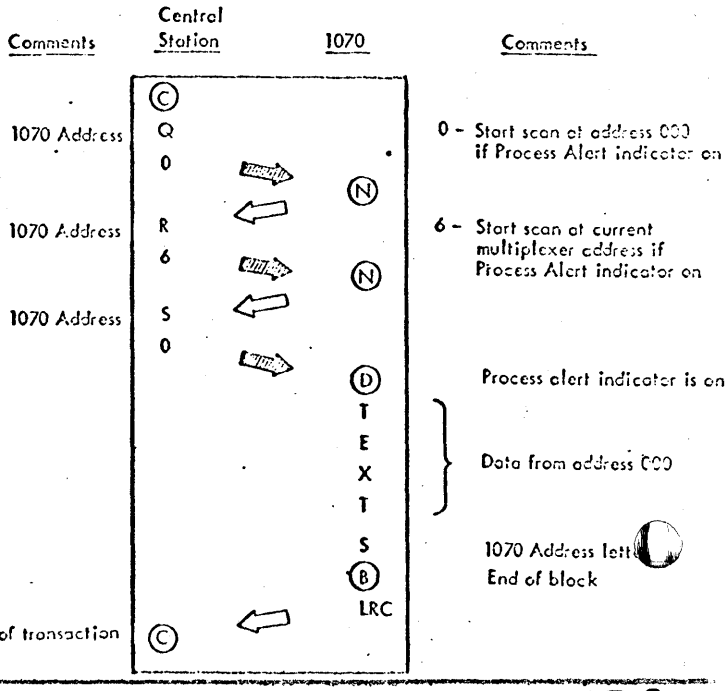
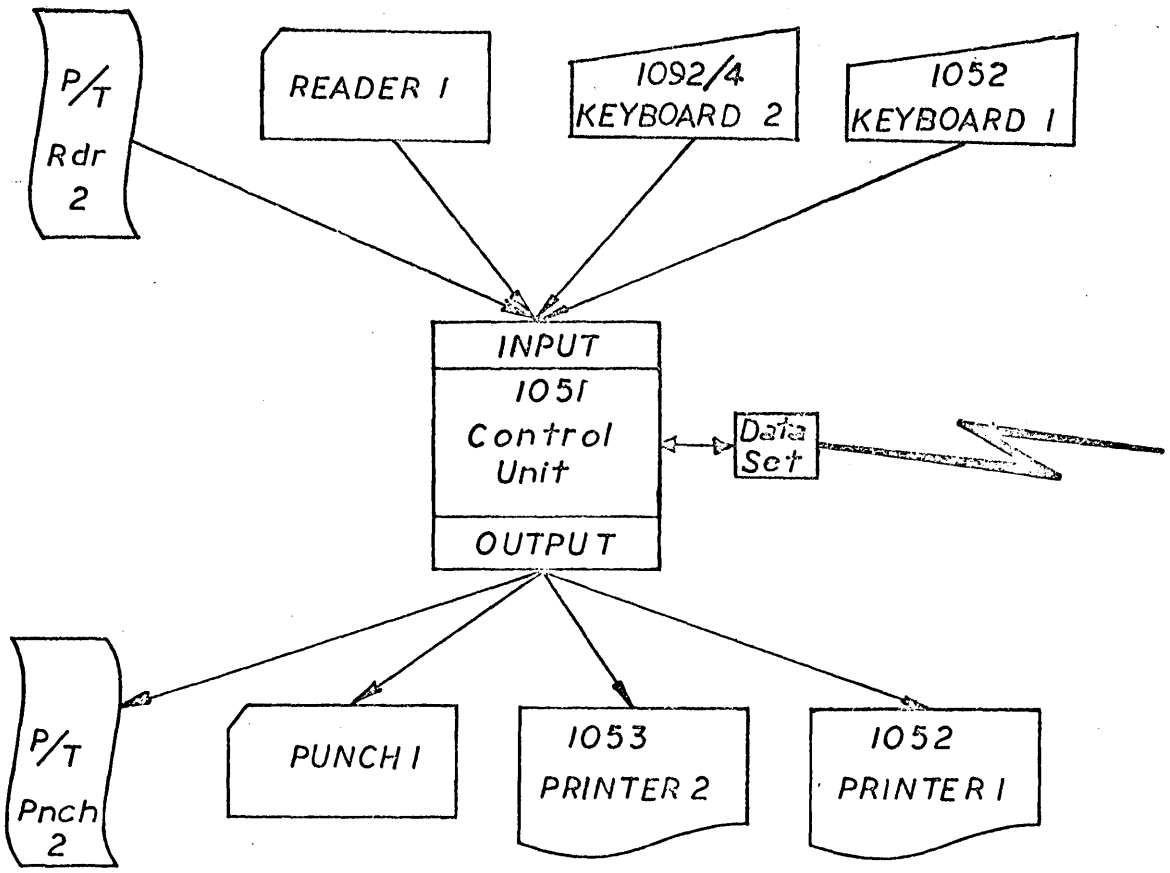
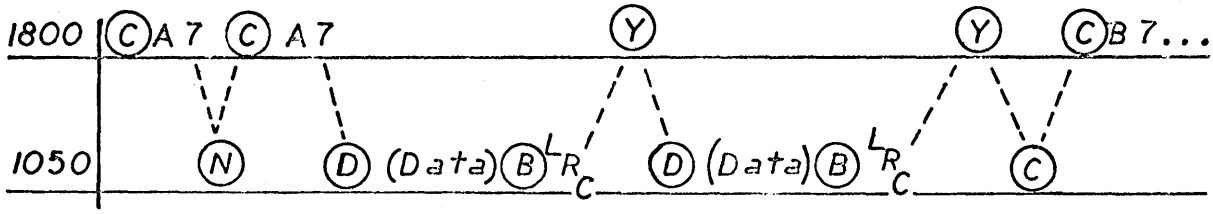


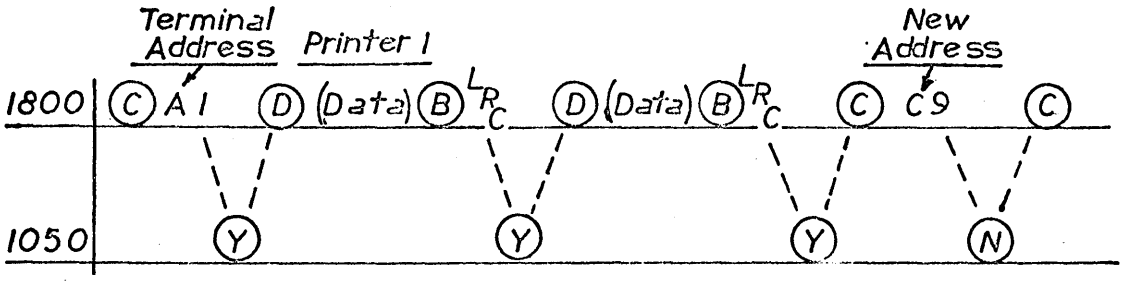
DIAGRAM 2: 1070 DATA FLOW.



POLLING SEQUENCE



SENDING SEQUENCE

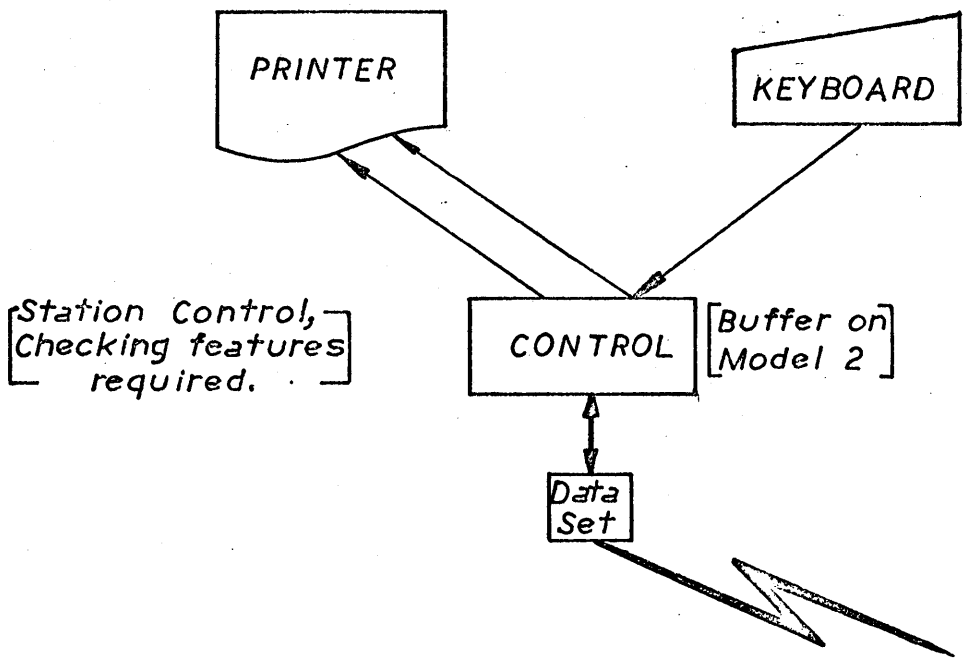


Terminal Ready

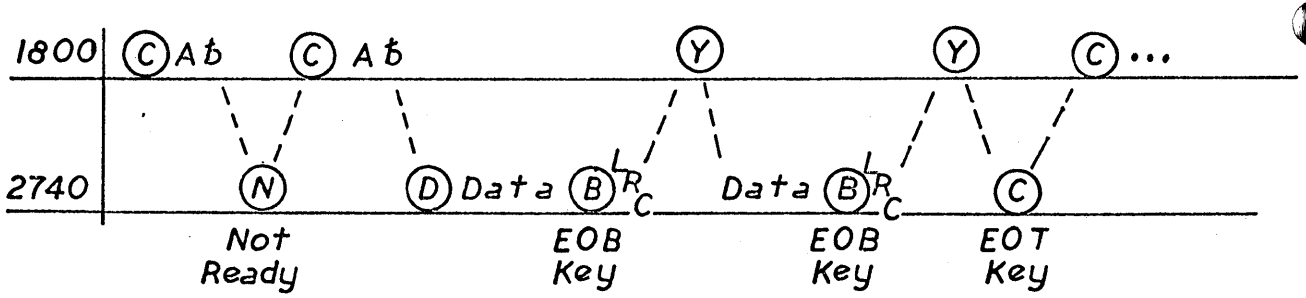
LRC OK

Terminal Not ready —or— Parity check in address.

DIAGRAM 3: 1050 SYSTEM



POLLING



SENDING

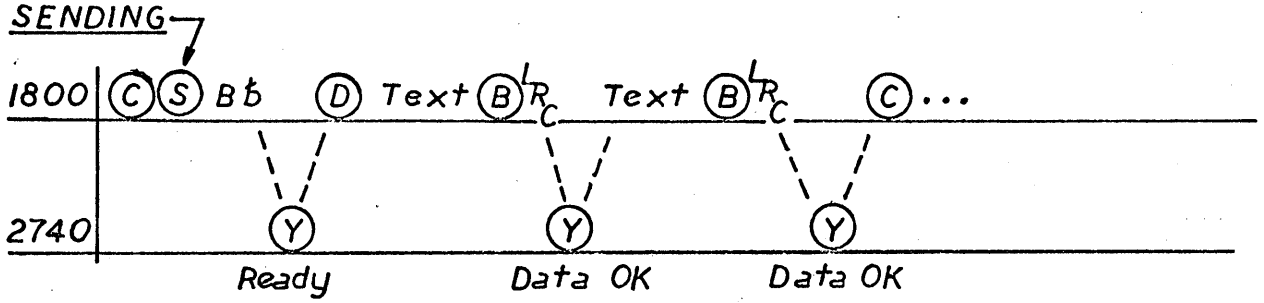
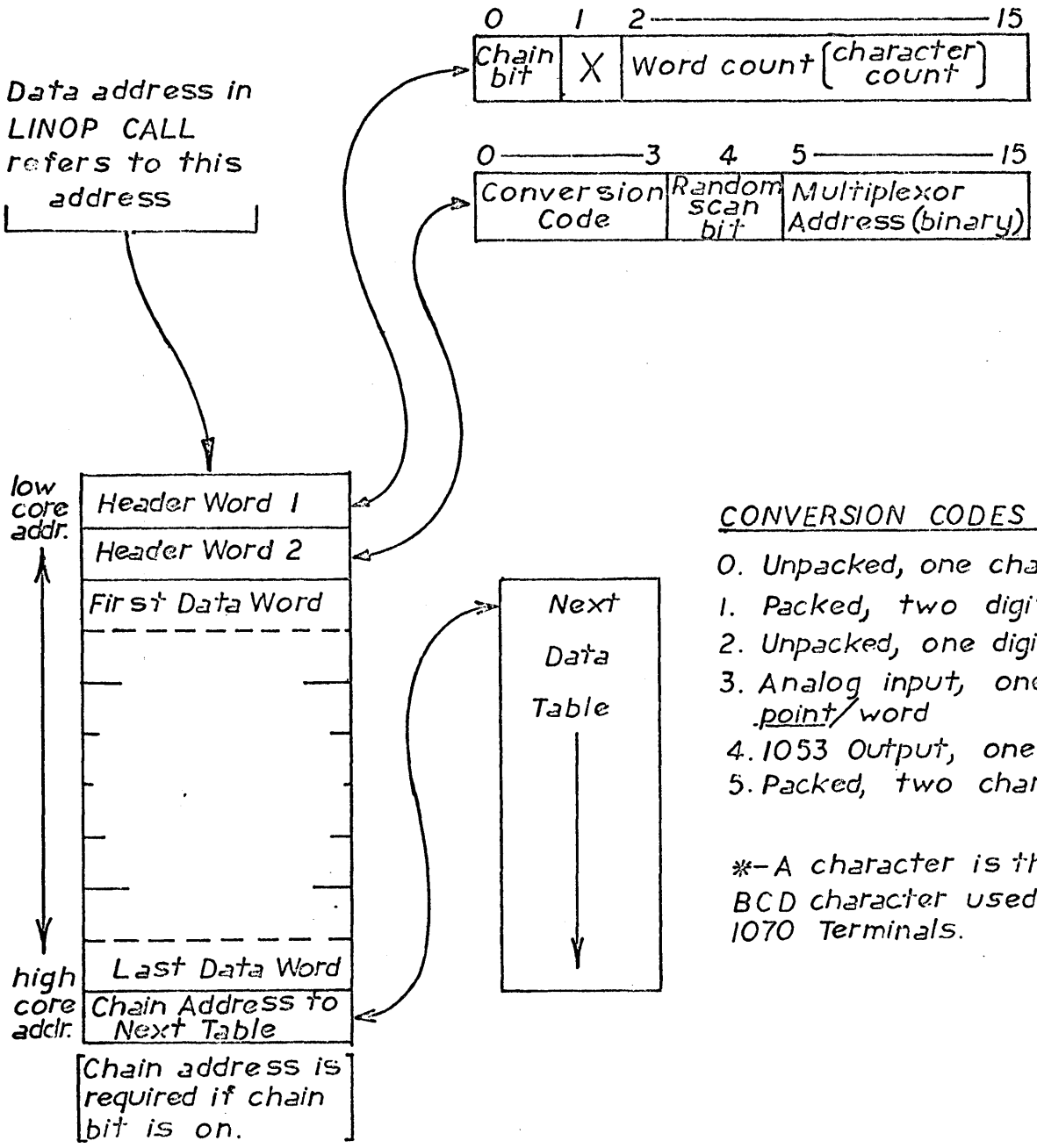


DIAGRAM 4 : 2740 SYSTEM

DATA TABLE LAYOUT
FOR 1800/1896 SUPPORT
CAISS



CONVERSION CODES

0. Unpacked, one char/word*
1. Packed, two digits/word
2. Unpacked, one digit/word
3. Analog input, one MPX point/word
4. 1053 Output, one char/word
5. Packed, two char/word

*-A character is the 6-bit BCD character used by the 1070 Terminals.

	Unpacked 0——9 10-12—15	Packed 0-2-4——7 8-10-12——15
BCD—	x——xBA8421	xxBA8421xxBA8421
Digital—	x——xx8421	xxxx8421xxxx8421

1053 Output - 0-2——7 8——15

xxBA8421 | FLCNT

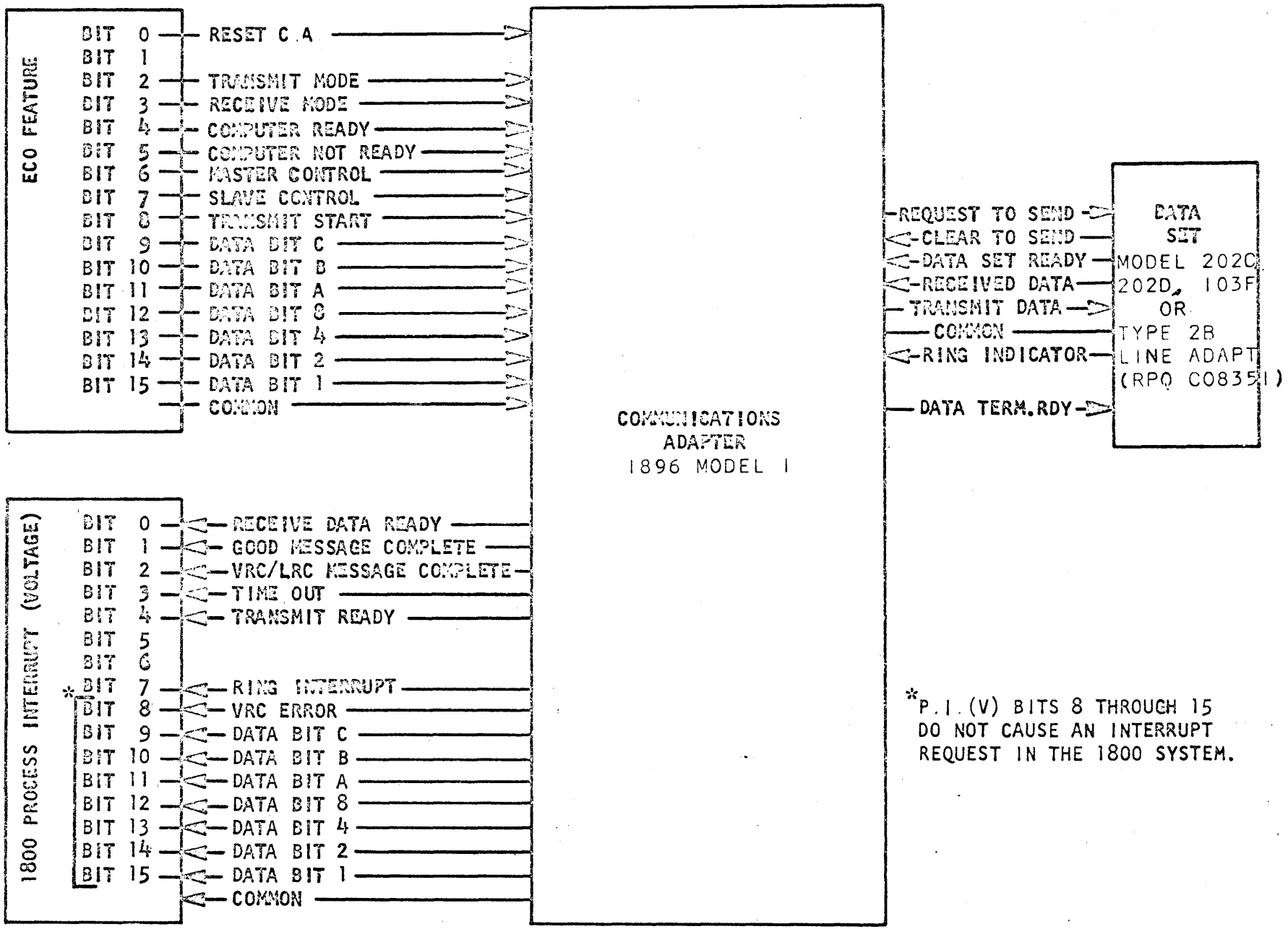
[No. (T)'s after char.]

rls

181.

DIAGRAM 5

DIAGRAM 6: HARDWARE INTERFACE



*P.I.(V) BITS 8 THROUGH 15 DO NOT CAUSE AN INTERRUPT REQUEST IN THE 1800 SYSTEM.

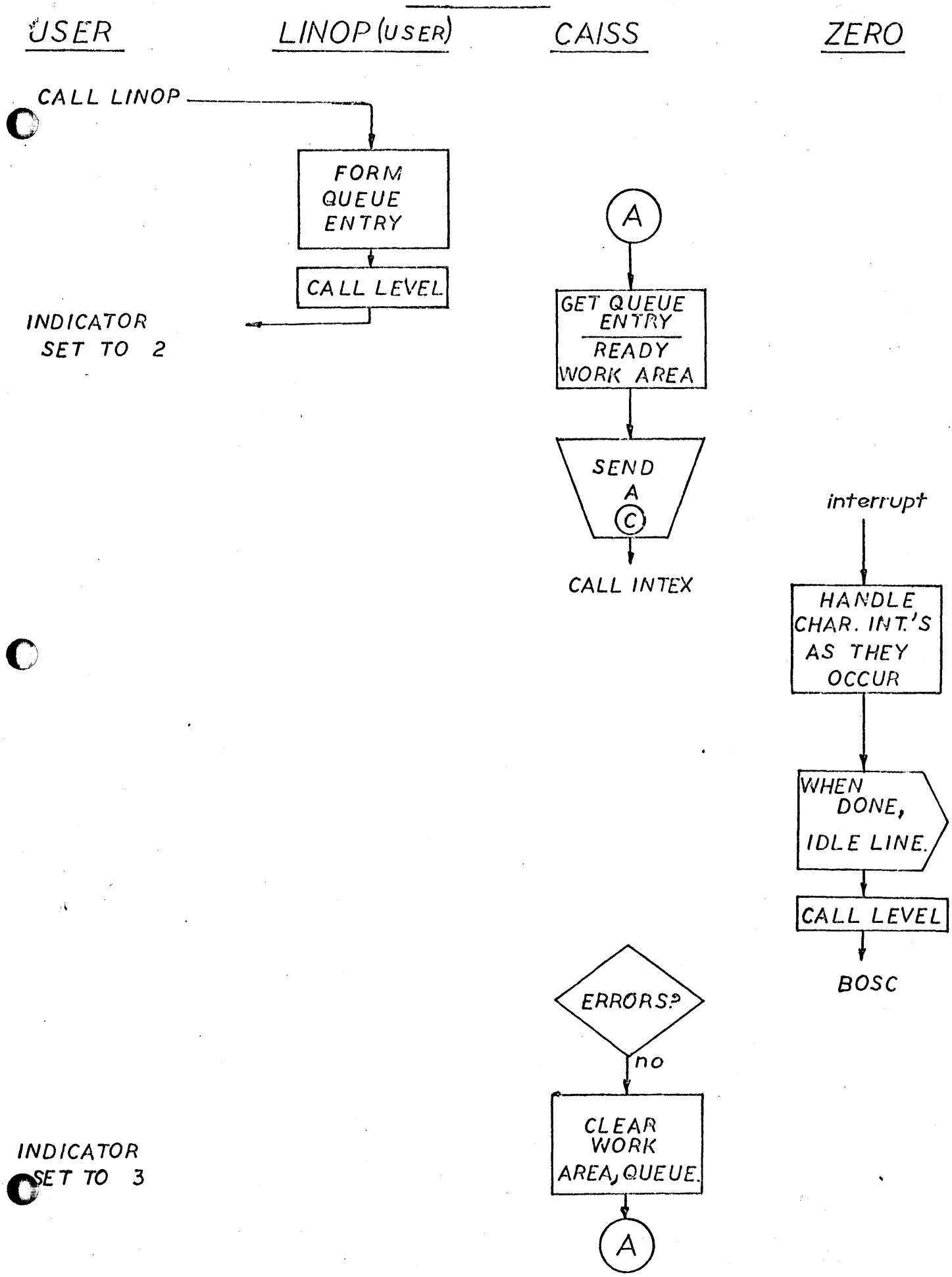


DIAGRAM 7: DATA FLOW

ABSTRACT

FAST FOURIER TRANSFORM
WITH APPLICATIONS FOR THE IBM 1800

The Fast Fourier Transform technique as developed by Cooley and Tukey has had a widespread effect in the field of time series analysis. However, some difficulty has been encountered by potential users in determining exactly how the technique works. An effort to explain the derivation in detail will be made.

Also, a description will be given of an analysis package program in which the Fast Fourier Transform technique is used to yield amplitude/phase spectra, power spectra, cross power spectra, and auto correlations.

FAST FOURIER TRANSFORMS

The algorithm for the computation of complex Fourier coefficients, as introduced by Cooley and Tukey[1], has had a widespread affect in the area of time series analysis. However, some difficulty has been encountered by potential users in determining exactly how the technique works. We will derive a special case for transforming a sequence with two factors and attempt to generalize the case for r factors.

INTRODUCTION

It is advantageous to review briefly the classical Fourier transformation in order to fully appreciate the fast Fourier transform (FFT). Fourier's theorem says that any time series X(t), which has a finite number of discontinuities and is periodic (i.e., X(t) = X(t+T)), can be considered to be a linear superposition of sine and cosine terms whose arguments are integral multiples of the fundamental frequency $\omega_o = 2\pi/T$. Mathematically this is to say

$$X(t) = \frac{1}{2} a_o + \sum_{n=1}^{\infty} (a_n \cos n\omega_o t + b_n \sin n\omega_o t) \qquad (1)$$

If we let

$$\cos n\omega_o t = \frac{1}{2}\left(e^{jn\omega_o t} + e^{-jn\omega_o t}\right)$$

and

$$\sin n\omega_o t = \frac{1}{2j}\left(e^{jn\omega_o t} - e^{-jn\omega_o t}\right)$$

We may rewrite (1) as

$$X(t) = c_o + \sum_{n=-\infty}^{\infty} c_n e^{+j\,\omega_n t} \qquad (2)$$

Here we have let $c_o = \frac{1}{2}a_o$, $c_n = (a_n^2 + b_n^2)^{1/2}$, $j = \sqrt{-1}$, $\omega_n = n\omega_o$. Equation (2) is known as the complex Fourier series of f(t). Upon examining (2) it becomes evident that we can describe the sequence X(t) completely in terms of the c_n and ω_n. This is known as the frequency domain representation of the time series X(t). The transformation from the time domain to the frequency domain is Fourier's transform and is written, where T is the time length of the series $X(\tau)$.

$$F(\omega_n) = \frac{1}{T}\sum_{\tau=o}^{T} X(\tau)e^{-j\omega_n t} \qquad (3)$$

The function $F(\omega)$ is in general complex, and

$$F(\omega) = R(\omega) + jI(\omega) = |F(\omega)|\, e^{j\phi(\omega)}$$

The complete frequency domain representation of X(t) is given by

$$|F(\omega)'| = (R(\omega)^2 + I(\omega)^2)^{1/2} \tag{4a}$$

and

$$\phi(\omega) = \text{TAN}^{-1} \left(\frac{-I(\omega)}{R(\omega)}\right) \tag{4b}$$

These equations represent the amplitude spectra and phase spectra of the given time sequence $X(\tau)$.

LIMITS OF INTEGRATION

Equation (3) represents the finite discrete form of the classical Fourier transform of the series X(t). The summation is performed over the full time range of the series $t = o \rightarrow T$. In order to satisfy the orthogonality conditions[2], which assure the validity of the transform, we must assume that the lowest frequency component we can transform is $\omega_o = 2\pi/T$. We must also bear in mind that when we use finite discrete sequences of X(t), sampled at uniform intervals Δt, we are dealing with a band limited function subject to conditions imposed by the sampling theorem. The sampling theorem says that the highest frequency we can reproduce at a sample rate $\Delta\tau$ is the Nyquist frequency ω_n.

$$\omega_n = 2\pi/2 \cdot \Delta\tau \tag{5}$$

If we have N points, spaced $\Delta\tau$ apart, we can rewrite T as

$$T = (N-1)\Delta\tau$$

and

$$\omega_o = \frac{2\pi}{(N-1)\Delta\tau} \qquad (6)$$

Since we transform only on integral multiples of ω_o, we can readily determine the number of frequency components K, that we can expect to find in a sequence X(t) which is of total time length T at a sampling rate $\Delta\tau$

$$K = \frac{\omega_n}{\omega_o} = \frac{2\pi/2\Delta t}{2\pi/(N-1)\Delta\tau} = \frac{N-1}{2}$$

The sampling theorem goes on to assert that

$$\omega\left(\frac{N-1}{2} + i\right) = \omega\left(\frac{N-1}{2} - i\right) \quad 0<i<\frac{N-1}{2}$$

Which is to say that when we transform on frequency values which lie between $(\frac{N-1}{2})\omega_o$ and $(N-1)\omega_o$, we can expect an exact replication of the results we obtained in the region of values $0 \leq \omega_n \leq (\frac{N-1}{2})\omega_o$ except that they are reversed in frequency.

We will see later that the FFT indexes frequency components from 0 to $(N-1)\omega_o$. This is necessary in order for the algorithm to be useful in performing the inverse transform.

THE FAST FOURIER TRANSFORM

The calculation of the complex Fourier coefficients from equation (3) requires a number of operations proportional to N^2 where N is the number of points to be transformed. Obviously this would require an appreciable amount of computer time when N is of a high order of magnitude. The technique as given originally by Cooley and Tukey[1], and further developed by Sande and Gentlemen[3] reduces the number of operations required to the order of $N \log_2 N$ where N is conveniently composite (a power of 2).

THE ALGEBRA

We may rewrite $F(\omega_n)$ from equation (3) in terms of the point indicies if we let $\omega_n = \frac{2\pi n}{N}$.

Substituting in equation (3)

$$F(n) = \sum_{\tau=0}^{N-1} x(t)\, e\left(\frac{nt}{N}\right) \qquad n=o,1,\ldots N-1 \qquad (7)$$

We have introduced the notation;

$$e(x) = e^{-j2\pi x}$$

Consider equation (7) for the case where N is the product of two integers AB. Notice that we may define any frequency index n in terms of A,B, and two new variables, a',b'.

$$n = b' + a'B$$

Similarly expressing the time index in terms of a,b.

$$t = a + bA$$

Where

$$a,a' = o,1,\ldots A-1$$

$$b,b' = o,1,\ldots B-1$$

Substituting for t and n in equation (7) get

$$F(b'+a'B) = \sum_{a=o}^{A-1} \sum_{b=o}^{B-1} x(a+bA)\, e^{\left(\frac{(b'+a'B)(a+bA)}{AB}\right)}$$

$$= \sum_{a=o}^{A-1} \sum_{b=o}^{B-1} x(a+bA)\, e\left(\frac{ab'}{AB} + \frac{aa'}{A} + \frac{bb'}{B} + a'b\right)$$

Since $e(a'b) = 1$

$$= \sum_{a=o}^{A-1} \sum_{b=o}^{B-1} x\ (a+bA)\ e\left(\frac{ab'}{AB}\right) e\left(\frac{aa'}{A}\right) e\left(\frac{bb'}{B}\right)$$

We factor terms not depending on the inside summation to get

$$= \sum_{a=o}^{A-1} e\left(\frac{aa'}{A}\right)\ e\left(\frac{ab'}{AB}\right)\ \sum_{b=o}^{B-1} X(a+bA)\ e\left(\frac{bb'}{B}\right)$$

Notice that the inside summation represents a Fourier transform of the B point sequence

$$W_a(b) = X(a+bA)$$

The result of this shorter transform is multiplied by a shifting factor $e\left(\frac{a'b}{AB}\right)$ before becoming the A point input to the second transform over a.

We have shown how an AB point transform can be defined in terms of 2 transforms of length A and B respectively.

Sande has shown the expansion for the case where N has three factors ABC.

$$F(c'+b'C+a'BC) = \sum_{a=o}^{A-1} e\left(\frac{aa'}{A}\right)\ e\left(\frac{ab'}{AB}\right)\ \sum_{b=o}^{B-1} e\left(\frac{bb'}{B}\right)\ e\left(\frac{c'(a+bA)}{ABC}\right)\cdots$$

$$\sum_{c=o}^{C-1} e\left(\frac{cc'}{C}\right)\ x(a+bA+cAB) \qquad (8)$$

Notice that as the number of factors we assign to N increases the shifting factors applied to each short transform become more complex. We can show a general form for r factors of N by letting

$$N = t_1 t_2 \ldots . t_r \qquad u_1 = o,1,\ldots\ (t_1-1)$$

$$u_r = o,1,\ldots\ (t_r-1)$$

So that in general

$$t = u_1 + u_2 t_1 + u_3 t_1 t_2 + \ldots u_r t_1 t_2 \ldots t_r \qquad (9)$$

Similarly we may let

$$n = v_r + v_{r-1}\ t_r + v_{r-2}\ t_r\ t_{r-2} \qquad (10)$$

Where the following correspondence exists between the expansion for N having 3 factors and the generalized notation

$$\{A,B,C\} \rightarrow \{t_3,t_2,t_1\}$$

$$\{a',b',c'\} \rightarrow \{v_1,v_2,v_3\}$$

$$\{a,b,c\} \rightarrow \{u_1,u_2,u_3\}$$

Sande has written the generalized expansion for N having r factors by making the definition

$$Q_r = t_1 t_2 t_3 \ldots t_r$$

Using this substitution in (9) and (10) we get

$$t = u_1 + u_2 Q_1 + \ldots + u_r Q_{r-1}$$

$$m = \left(\frac{v_r}{Q_r} + \frac{v_{r-1}}{Q_{r-1}} + \ldots + \frac{v_1}{Q_1}\right) Q_r$$

We will express the first j terms of t as

$$t'(j) = u_1 + u_2 Q_1 + \ldots u_j Q_{j-1}$$

The Sande-Tukey decomposition of the general transform expression uses two identities which result from the algebra. The first of these is an expression for the generalized exponential term;

$$e\left(\frac{nt}{N}\right) = e\left[\sum_{j=1}^{r} v_j \sum_{k=1}^{j} u_k \frac{Q_{k-1}}{Q_j}\right]$$

The second is the generalized expansion for the summation on t

$$\sum_{\tau=o}^{T-1} = \sum_{u_1=o}^{t_1-1} \cdots \sum_{u_r=o}^{t_r-1}$$

Combining these results we can get a generalized expression for N having r factors

$$\sum_{\tau=o}^{T-1} e\left(\frac{nt}{N}\right) x(t) = \sum_{u_1=o}^{t_1-1} e\left(u_1 \frac{v_1}{t_1}\right) \left[e\left(v_2 \frac{t'(1)}{Q_2}\right) \sum_{u_2=o}^{t_2-1} e\left(u_2 \frac{v_2}{Q_2}\right) \right.$$

$$\left[e\left(v_3 \frac{t'(2)}{Q_3}\right) \cdots \left[e\left(v_r \frac{t'(r-1)}{Q_r}\right) \sum_{u_r=o}^{t_r-1} e\left(u_r \frac{v_r}{t_r} X(u_1 + u_2 Q_1 + \right. \right. \right.$$

$$\left. \left. \left. \cdots + u_r Q_{r-1} \right] \cdots \right] \right]$$

If we consider the expansion for N=2.2 we see that one stage of complex arithmetic is avoided in the inside summation. For the case where N has 3 factors, ABC, the interior summation is extremely simple; however, we are faced with shifting factors in the subsequent summations which require sine and cosine terms to be evaluated. The shifting factors take on a complex configuration as the number of summations is increased. Programming is simplified by using the 3 factor expansion in all cases and letting C equal 2 or 4 so that we may avoid a factor of 2 or 4 complex operations.

REFERENCES

1. J.W.Cooley and J.W.Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," Mathematics of Computation, vol. 19, No. 90, (1965) pp. 297-301

2. H.P. Hsu, "Fourier Analysis," Simon and Shuster, New York, 1967, pp. 197-200

3. W.M. Gentlemen and G. Sande, "Fast Fourier Transforms - for fun and profit", 1966 Fall Joint Computer Conf. AFIPS Proc., vol. 29, Washington, D.C., Spartan, 1966, pp. 563-578.

4. C. Bingham, M. Godfrey, J. Tukey, "Modern Techniques of Power Spectrum Estimation," IEEE Transactions on Audio and Electro Acoustics, vol. AU-15, No. 2, June 1967, pp. 56-65

(6)

NIMS

THE AEROSPACE SCIENTIFIC DATA REDUCTION MONITOR SYSTEM

By

Charles R. Aumann

1 December 1967

Mathematics and Computation Center
Aerospace Corporation
San Bernardino, California

196.

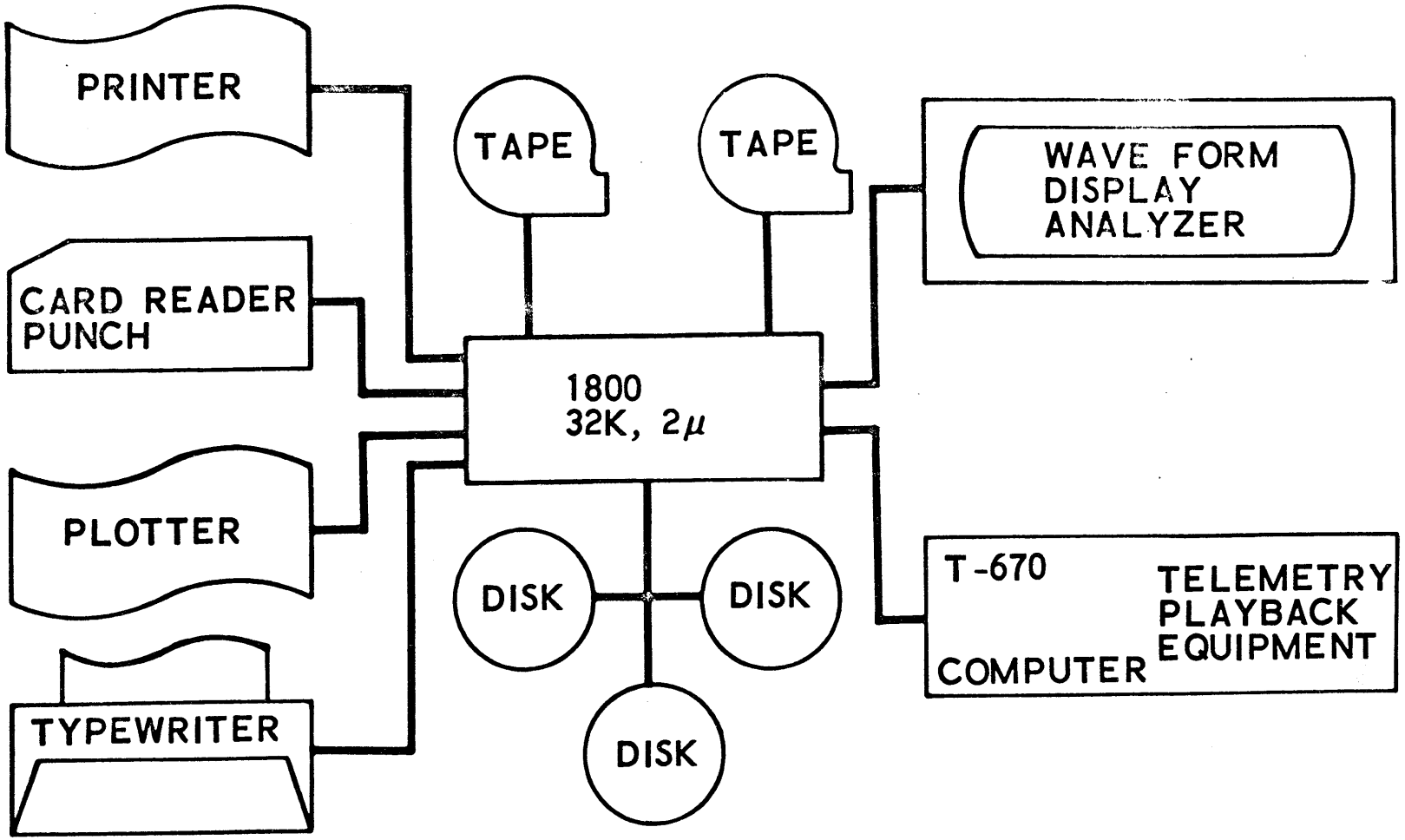
1. INTRODUCTION

The IBM-TSX System was designed for an IBM-1800 operating in a real time environment as a process controller. The Aerospace IBM-1800 is used primarily as a post flight data reduction system interacting with telemetry ground station equipment. This demands the full interrupt facilities of the sytem but in a batch processing mode of operation such as offered by the TASK stand alone type system. As a result the NIMS Monitor System was developed to combine the features of the process and non-process modes of operation into a general mode of operation in which the full interrupt and peripheral facilities of the machine are available to the active application program under execution.

A schematic representation of the Data Reduction computing complex is shown in Figure 1. The IBM-1800 is a 32K core size machine with 2 microsecond cycle time. Also attached are 2 magnetic tape units, 3 disk drives, a 1442 card/reader punch, a 1443 printer, a 1627 plotter, and a 1816 typewriter-keyboard. This system services the telemetry playback equipment through a Telemetrics T-670 PCM decomutation computer and a special purpose display terminal called a Waveform Display Analyzer.

The prime mission of the 1800 system is high speed data logging from telemetry tapes of either analog or digital data plus driving the attached graphical display and recording equipment.

IBM - 1800 System Configuration



2. COLD START CONSIDERATIONS AND MODIFICATIONS

The initial endeavor to provide a more convenient monitor system for the data reduction programs involved at attempt to return to a one card cold start procedure. In addition, it was considered desirable that an automatic cold start of restoring the systems skeleton from disk be accomplished between jobs instead of a task reload.

A decision was made that defective cylinders on a disk pack would not have to be handled by the cold start routine. This decision was based on the fact that there have been no defective cylinders on any of 12 disk packs used in the computer center in 18 months of operation. If a disk pack does develop a defective cylinder, it will be discarded or refurbished.

This allowed the LDR-Cold Start Load Card Program and the CLD-Systems Cold Start Program to be rewritten in very simplified form such that a one card loader would initiate a complete core loading cold start from the systems disk.

The NIMS-TASK program was also modified such that the action of pressing the immediate stop, start buttons on the console would cause the same action as loading the cold start card. This stipulates that core storage not be storage protected at any time. Figure 2 presents a table of the modifications required to provide the new cold start capability.

3. PROCESS/NON PROCESS CONSOLIDATION

In performing the analysis of what type of system would be most desirable, the general conclusion pointed toward a non-process offline TASK system with a full interrupt processing capability at that level.

Table of Modified Programs for Cold Start

1. LDR - COLD START LOADER PROGRAM
   - EXTENSIVE MODIFICATIONS
   - 27_{16} WORDS
2. CLD - COLD START PROGRAM
   - EXTENSIVE MODIFICATIONS
   - $E8_{16}$ WORDS
3. TASK - SKELETON PROGRAM
   - MINOR MODIFICATION TO TASK RESTORE ROUTINE

The NIMS monitor system was founded as a TASK offline system with certain protective features in the core-load builder modified to allow the use of interrupt service subroutines, the masking subroutines, and timer subroutines with non-process coreloads. Thus, because TASK is the basic building block, the TASK support programs such as TRACE, CORE DUMP, DISK DUMP, etc., are available with this system.

The core-load builder is constructed such that the special purpose operation codes associated with different type core-load builds are separated in their table by delimiting constants. For example, a process type operation code could not be used in anything but a process coreload. It was a very simple thing to change those delimiters such that the operation code tables for the process and interrupt coreload functions were included in the non-process operation code table.

The final modification negated the test in the non-process supervisor for the type of coreload being executed. Figure 3 represents a table of the modifications necessary to provide the consolidation of process and non-process coreload functions.

4. TRACE DEVELOPMENT

The TASK utility programs have been included as a subset of the NIMS monitor system. All utility programs were considered adequate for debug tools by the data reduction programmers with the exception that it was felt that the TRACE output was too difficult to interpret.

Table of Modifications to Consolidate Process and Nonprocess Functions

1. CLB - CORE LOAD BUILDER PROGRAM

   - MINOR MODIFICATION TO OPERATION CODE TABLE LIMITS

2. NPS - NONPROCESS SUPERVISOR PROGRAM

   - MINOR MODIFICATION TO REMOVE TEST FOR PROCESS CORELOAD

This problem lead to the development of inserting additional coding in the TRACE routine in TASK to provide a mnemonic format conversion for generating the output for the 1443 printer. An example of output generated while running under the TRACE mode is shown in Figure 4. The prime features of this development are a much easier format to read and the calculation of the effective address that the instruction operation code will be referencing.

5. SOURCE PROGRAM FILE STORAGE SYSTEM

Significant effort was expended in providing a source program file storage capability in conjunction with the NIMS monitor system. This capability provides programmers the means for storing source language programs for subsequent retrieval during checkout activities. The programs are stored on the file such that alter cards could be read through the card reader to modify the source program before the assembler performs its function. This activity was accomplished in two pnases. In the first, the system components such as TASK, ASM, FTN, and DUP were modified to accept input from magnetic tape. In the second, a series of programs were written to generate a systems tape, provide an alter capability, and perform a periodic update of the source program systems tape.

The key to providing the systems routine with the capability of accepting input from magnetic tape involved including the coding of the MAGT routine in the TASK skeleton and providing a one word logical switch in fixed core for interrogation by the input

Trace Output

LOC	OPCD	F	TAG	DISP	ADDR	EFFA	MEMORY	ACCU	MO	IX*1	IX*2	IX*3	CO	OFLW
38BF	LDX	L	2	00	0040	0040	1674	0000	0000	0040	000B	20B3	NO	NO
38C1	BSC	I		80	38AE	2097	70FC	0000	0000	0040	0040	20B3	NO	NO
2097	MDX			FC		2094	4400	0000	0000	0040	0040	20B3	NO	NO
2094	BST	L		00	38AE	38AE	2097	0000	0000	0040	0040	20B3	NO	NO
38B0	STX		2	0F				0000	0000	0040	0040	20B3	NO	NO
38B1	LDX	I	1	80	38AE	0000	7001	0000	0000	0040	0040	20B3	NO	NO
38B3	LDD			1E		38D2	0000	0000	0000	2096	0040	20B3	NO	NO
38B4	LD		1	00		117D	F1F1	0000	0000	2096	0040	20B3	NO	NO
38B5	RTE			CC				B000	0000	2096	0040	20B3	NO	NO
38B6	STO			0C		38C3	000B	000B	0000	2096	0040	20B3	NO	NO
38B7	LDX	I	2	80	38C3	000B	145E	000B	0000	2096	0040	20B3	NO	NO
38B9	BSI	I	2	80	38C4	38E0	0000	000B	0000	2096	000B	20B3	NO	NO
3A88	BSC	L		20	3A8C	3A8C	4C80	0000	0000	2096	000B	20B3	NO	NO
3A8A	MDX	I		FF	38AE	2096	B000	0000	0000	2096	000B	20B3	NO	NO
3A8C	BSC	I		80	3A86	38BB	7402	0000	0000	2096	000B	20B3	NO	NO
38BB	MDX	L		02	38AE	38AE	2095	0000	0000	2096	000B	20B3	NO	NO
38BD	LDX	L	1	00	0040	11BD	4480	0000	0000	2096	000B	20B3	NO	NO
38BF	LDX	L	2	00	0040	0040	1674	0000	0000	0040	000B	20B3	NO	NO
38C1	BSC	I		80	38AE	2097	70FC	0000	0000	0040	0040	20B3	NO	NO

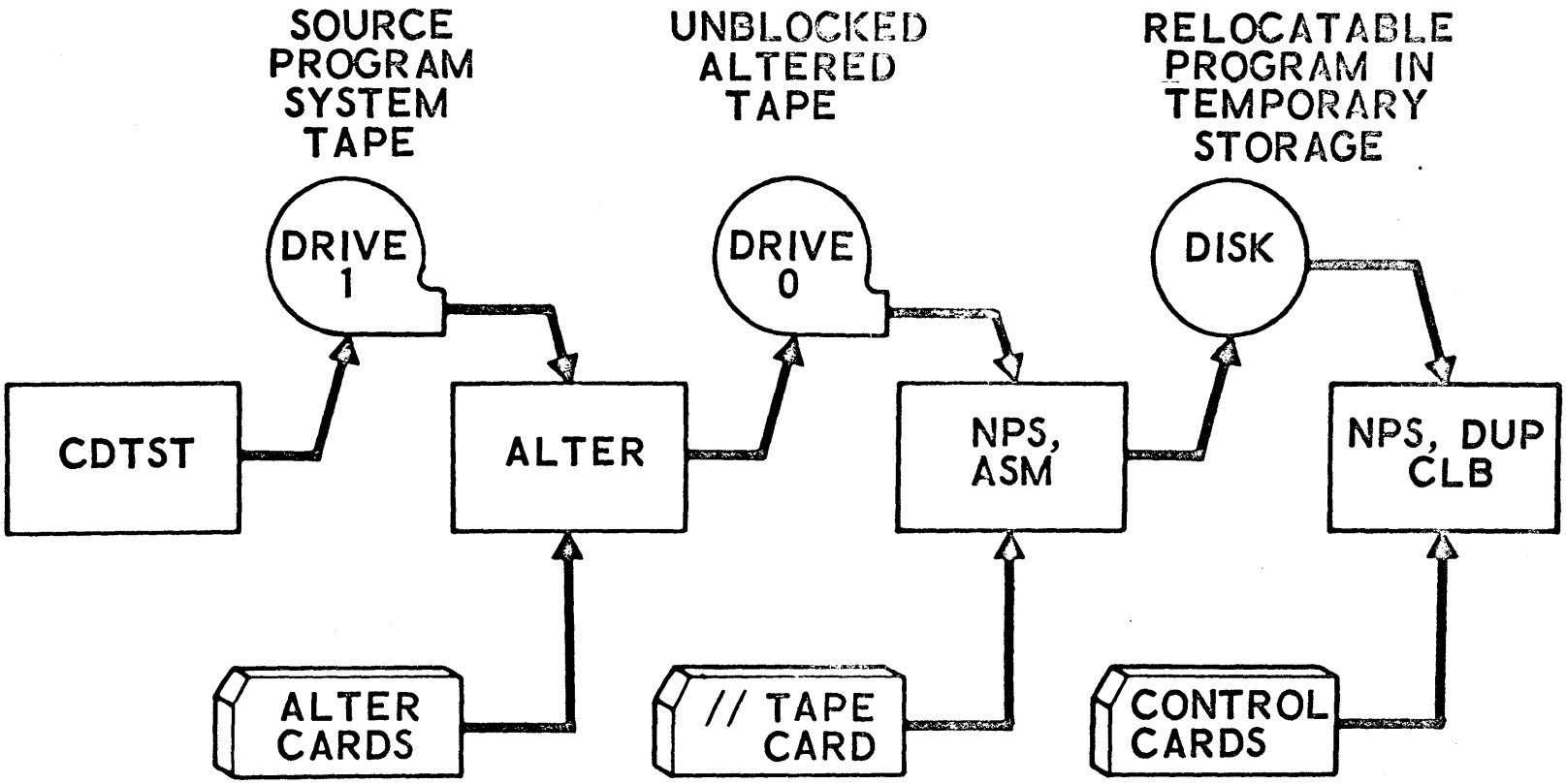
segments of the systems routines. This switch forces the I/O routines in the system to read tape drive 0 when set by the non-process supervisor after reading a // TAPE card from the card reader. The switch will be reset for the I/O routines to read from the card reader if a // CARD image is read off of magnetic tape.

It was also necessary to modify each of the I/O routines in the assembler, FORTRAN compiler, disk utility program, and the non-process supervisor. Since CARDN is included in the NIMS-TASK, the coding of CARDN in each of the systems routines was replaced with the coding necessary to test the card/magnetic tape switch and perform the magnetic tape input operations. Depending on the routine, it was necessary as part of the magnetic tape input function to convert the card image from tape either back to card image format, a special one character right justified EBCDIC, or leave in standard two word EBCDIC.

Figure 5 is a pictorial representation of the NIMS magnetic tape system showing how the support programs interface with the systems programs to perform a complete job operation.

The CDTST-Card to Source Program System Tape Program is used to place the initial version of a source program on the system tape. This procedure involves searching the system tape for the last record, back-spacing the tape to prepare for the write operation, and writing the card images from the card reader in blocked tape format of 200 cards per block. The termination card of the program causes a new end-of-tape record to then be generated.

NIMS Tape Assembly Procedure



The ALTER-Source Program System Tape Alter Program has the function of allowing the programmers the capability of altering source statements as it is unblocking the program from the source program system tape to a scratch tape in preparation for input into the assembler program. Figure 6 shows the three data card formats that can be used with the ALTER program. The alter function of the alter cards is determined by the character in column 72 as follows:

R means replace the card image on the source program system tape having the corresponding sequence number in columns 77-80 with this card image.

I means insert this card image in front of the card image on tape with the same sequence number in columns 77-80.

D means delete the card images on tape inclusively between the two sequence numbers in columns 73-80. If only one card image is to be deleted, the sequence number should be punched in columns 73-76.

The output magnetic tape on drive 0 is generated in 80 character EBCDIC format for direct input into the ASM system assembler. This is triggered by a // TAPE card following the last alter card in the card input stream.

After the assembler has finished with the assembly process and stored the relocatable program in temporary storage on the disk, the next record is read from the tape input stream on drive 0. This record is a // CARD image transferring control back to the card reader. The program that has just been assembled can then be executed or permanently stored on the disk.

Sample Alter Card Formats

21	27	32	35	46	72	77
	MDX	L	ABC, -4	DECREMENT ABC TABLE	R	0051
XYD	LDD	1	TEMP		R	0193
	STD	2	TAPE-1		R	0194
SW1	NØP				I	0512
	LIBF		MAGT	REWIND TAPE	I	0719
	DC		/5000		I	0720
					D0999	
					D10041021	

208

The third program associated with the NIMS tape assembly procedure is UPDAT-Source Program System Tape Periodic Update Program. This program will utilize submitted alter decks as input to generate an updated version of the source program system tape. The programs that were modified by this run will be resequenced on the new source program systems tape.

6. SYSGØ - NIMS PROTECTION PROGRAM

Early in the operation of the TSX system, a programmer inadvertently destroyed the system on the disk to a degree that it was necessary to do a new system generation from cards. It was decided at this point that the system disks would be copied on magnetic tape with a capability implemented to restore the disks from this tape.

The SYSGØ program was written to provide the disk to tape operation, the punching of a self-loading execution card, and the subsequent tape to disk regeneration process. The first step of the execution of the program involves storing the first 12 locations of core in a table for the regeneration phase of the program. A split-binary self loading card is then punched that is used to load the system skeleton and the SYSGØ program into core from magnetic tape during the regeneration phase. Next core storage from location 0 through hex 3000 is written out on tape. This record contains the system skeleton as well as the SYSGØ program. As a final operation the system disks are then copied to magnetic tape 20 sectors in a record.

After a SYSGØ tape has been generated, it can be saved for later use to restore new systems disks. This is accomplished by clearing core storage and loading the self-loading card punched by the disk to tape phase of the program. The loader card causes the first record from the tape containing the system skeleton and the SYSGØ program to be read into core storage starting at location 12. The final instruction on the card is a branch to the appropriate entry point in the SYSGØ program that will copy the remaining tape records on the systems disks. Figure 7 presents a schematic representation of the operation of this program.

7. MINOR SYSTEM DEVELOPMENTS

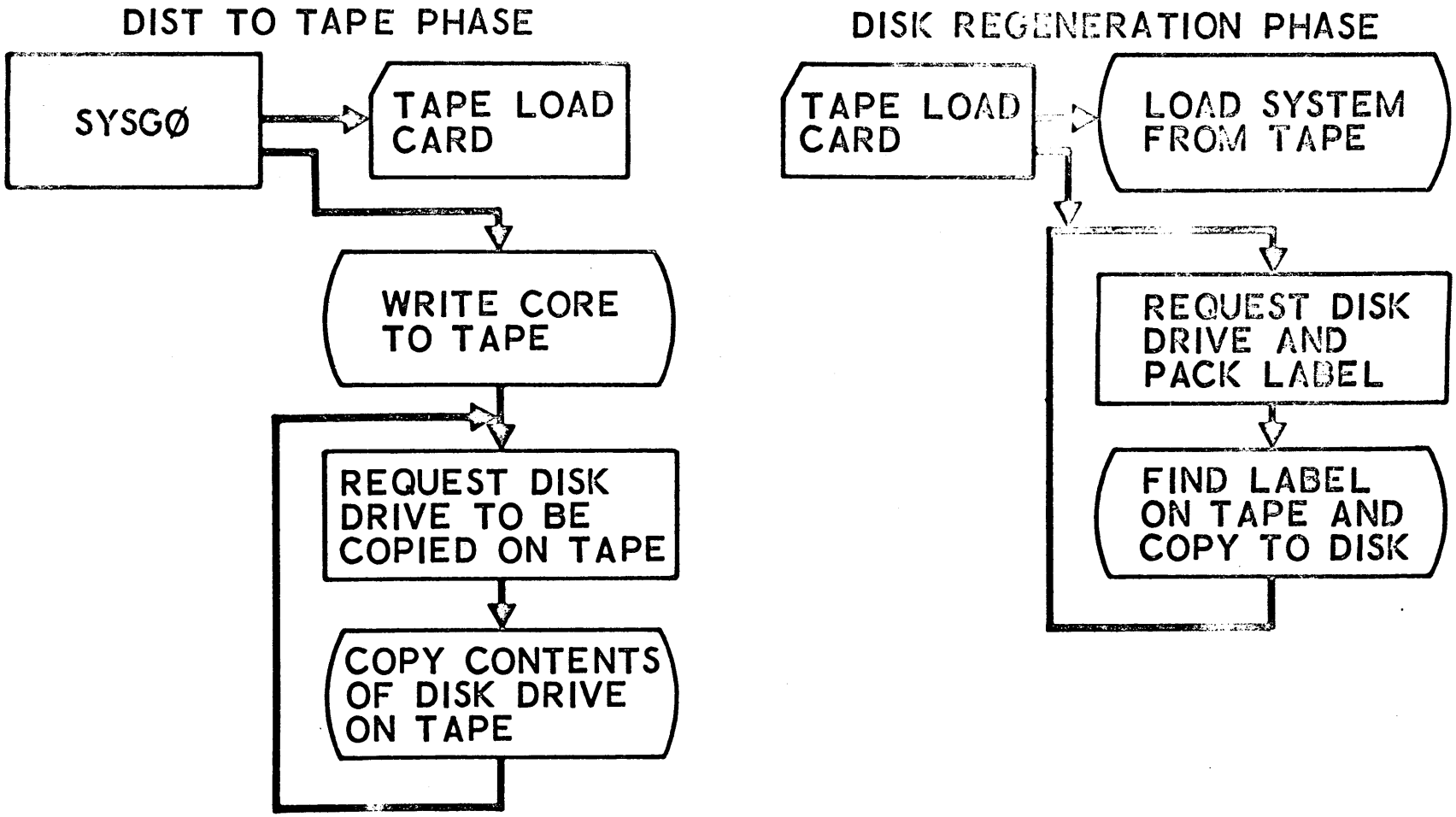
Additional features have been added to the NIMS monitor system that extend the performance of the system or provide a more simplified operating procedure.

The standard ASM Assembler Program has been replaced by an IBM released MACRO Assembler Program called MASM. This has been found to be a powerful tool for the programmers providing extended programming capability.

The NIMS-TASK program was modified to provide the programmers and operators with the ability to execute stored coreloads by typing in the program name on the typewriter. This activity is initiated by pressing the console interrupt button with no sense switches on. The typewriter will request that a 5 character program name be entered through the keyboard with the following message:

TYPE IN FIVE CHARACTER PROGRAM NAME

SYSGØ System Storage Procedure



When the name is typed, the NIMS-TASK program generates a card image to execute a fixed coreload and transfers control to the non-process supervisor. The non-process supervisor with the card image in its input buffer then performs a FLET search and proceeds to execute the coreload.

The data reduction programmers requested that a current date be printed on the assembler and compiler output listings. This was implemented in NIMS-TASK and the non-process supervisor such that any control message printed by NPS would have a date loaded from the fixed area of TASK included on the end of the line. The date is stored in TASK by executing a program called DATE where the date is requested by the typewriter as follows:

TYPE IN DATE -DD MMM YY-

When the day, month, and year are typed in, the program converts them to print codes and stores them in the fixed area of TASK.

8. SUMMARY

Every attempt was made in the implementation of the NIMS monitor control system to provide the data reduction programmers and operators with a convenient system oriented to telemetry data reduction. This endeavor has been accomplisned and further activities will be directed to additional facilities for the convenience of the programmers and operators.

INVESTIGATION OF
ABNORMAL OPERATING CONDITIONS

M.4.8.
(1)

COMMON MEETING
San Francisco
December 11, 1967

Kenneth R. Andersen
Senior Field Engineering Specialist
Area 12 Technical Assistance Group
San Francisco

Investigation of Abnormal Operating Conditions

A few short years ago many of our data processing system error recovery procedures were hardware dependent. That is to say the program hung up until the hardware circuitry recognized that the condition causing an error had been corrected or reset. 1401 systems stopped with a Process, Reader, Punch, Printer, or Tape error-indicator lamp on the operator's console, directing the operator to inspect a particular device.

The reader may indicate a "reader check" (i.e., card read or checked in error), "punch stop" (i.e., card misfed or jammed), or a tape unit may not be ready. The system could not continue until the operator had corrected the cause of the error by refeeding the error card, correcting the feed problem, or making the tape unit ready.

The point I'm trying to illustrate is that the problems were fairly obvious and the improper restarting of the system difficult when viewed from todays's environment.

The design, architecture, and flexibility of System /360 makes this type of operation impractical. Error information is presented to the program and further visual indications to the operator are dependent on the system control program. Since, for example, a system /360 Model 30 can run under BPS, BOS, TOS, DOS, OS, emulators, or a user written control program, an

operator must be knowledgeable of the specific system in use. Let me give you an example. A model 30 operator has just observed the system in the wait state. Hitting the start or interrupt button produces no effect on the system. What has happened? There are not outward error indications! No messages have been printed on the typewriter! If BPS or an Operating System is controlling the system certain bytes of main storage will contain a coded message indicating the reason for the halt. Only prior knowledge will direct the operator to display this area. Furthermore, interpretation of this data can be in zoned decimal, binary, or packed format.

Basically there are 3 main areas a system can be divided into. The "Central Processing Unit" (CPU), or heart of the system; the "Channel," over which the CPU communicates with I/O devices; and the "Input/Output Units." A fourth area could be "Non Operational Units." These may be off line units or devices never installed on the system. Any address not on the system will result in Non-Operational Status if used.

I would like to define the effect of problems in each of these areas.

Central Processing Unit errors are caused by bad data on data busses or in key registers of the CPU and result in a logout via system micro-programming. From 12 to 256 bytes of information and status at the time of failure are recorded and the new machine check PSW is loaded. Appropriate indicators are

set in storage by the control program as operator information and the system is placed in the wait state with all interrupts disabled. Channel failures are errors that occur during communication between the CPU and an I/O device. This could happen if the CPU signals a device and the device fails to respond within a specified time. A programmer requesting a unit to read or write no data could also result in a channel error. Channel failures may result in a micro-program logout with appropriate indicators set in storage as operator information. This area should be investigated closely as failures can result from improper channel programming.

Input/Output errors pertain to specific devices (tapes, disks, reader, punch, printer, etc.) and always result in a sense command being issued to that unit. This sense operation will transfer up to six bytes of information from the unit to the CPU for analysis by the control program. Error recovery will be performed dependent on the error. Tape read errors result in a back-space and reread with a TIE if possible. Each time this sequence has been performed eight times, a tape cleaner routine is executed. The tape is backspaced 3 records and forward spaced twice. This runs the record over the tape cleaner blade removing any excess oxide. After 100 rereads are performed, it is considered an unrecoverable I/O error and the job may be cancelled. Tape write errors result in a backspace, erase and rewrite. Disk read or write errors cause a retry unless a defective track is found. A defective track will cause a seek to an alternate track and a read or write. Unit record devices require operator action at the device and a proper

response to the programming system. The operator should examine the sense information for the cause of error.

If the job had cancelled under the Disk or Tape Operating System all the pertinent information will be logged on the typewriter; however, it must be decoded with respect to the specific I/O unit to obtain all the meaningful information. The same error on BPS or BOS would result in the wait state. Unit record type devices will require operator intervention. Restart procedures require the proper reference SRL. Operating guides for all devices should be available to operators.

Proper maintenance of materials can save countless hours and headaches.

Initialize all disk packs before use. Notice that the plant has affixed a label indicating tested bad tracks. These tracks should have alternate tracks assigned during your initialization. This is necessary because surface analysis performed prior to shipment tests the complete usable surface for each track. Subsequent use of these tracks will be unpredictable and dependent on individual R/W head tracking on each disk drive.

Tape must also be properly initialized, even in an unlabeled shop. Tapes are always checked by logical IOCS to ascertain if label information exists. An improperly initialized tape my result in a permanent tape read error and cancellation of your job. This can be avoided by writing a TM

at the beginning of all unlabeled tapes. Remember the seven or nine track tape marks can only be read error free on their respective drives. Mark all reels plainly for use on one tape drive (i.e., seven track Drive 183). Operators should clean tape drives as often as necessary to minimize tape errors.

Card decks should be given proper care. Plastic edged decks are best in high usage circumstances. Master decks should never be used except for reproduction. This is your best protection against card decks that ran yesterday and program check today. Be certain you have an IBM card gauge and check for punching off registration periodically. A high incidence of reader checks or one card that cannot be read also require checking with the card gauge.

What methods should be used during isolation of an abnormal operating condition? This depends on a great many variables; however, a certain basic attitude should prevail. How can I obtain the maximum useful and meaningful error data for analysis by the programmer or IBM engineer?

The first point should go without saying, but all too often goes unheeded. Stop! Determine what the system will tell; you may have to ask (i.e., display appropriate core locations). Remember, a hardware failure does not necessarily produce a red light indication. On DOS or TOS only by displaying Main Storage locations 0-3 can it be established that an error

has occurred. On BPS and a BOS hex location X '32' contains the error indicator. What action the operator should take is described in the operator's manual. If a machine or channel failure has occurred, a logout will take place. The system is placed in the wait state to allow the operator to run SEREP. The "System Environment Recording Edited Printout" program is a stand-alone card deck that will interface with IBM control programs, determine the type error that has occurred, and print out all pertinent information and status of the system at the time of failure. This program will be provided by your Customer Engineer at installation. SEREP will print out from 12 to 256 bytes of logout area (dependent upon CPU type), old PSW's, new PSW's, CAW, CSW, GP regs, Floating Point Regs., or sense information.

It is especially useful when a customer engineer is not on site. It is essential that all operators be versed in rocognizing a SEREP request. A core dump or re-IPL at this time will destroy any information of value.

SEREP will pinpoint invalid use of channel programming, use of non-operational devices, and bad I/O devices. All SEREP runs should be saved for analysis.

Only after the operator has collected all meaningful data for subsequent analysis should a run be attempted. This is particularly important on intermittent failures.

If a SEREP run was not requested a core dump should be taken. Save all core dumps, console logs, output, input, program decks. Documentation

to the extent of recreating the failure is best.

Try to isolate the culprit by changing devices, restoring disk packs, using other card decks, or if possible run the job on another system.

Of course if the operator believes a hardware system failure exists, he should follow the normal procedures for notifying his customer engineer. If the problem seems to be in programming all the pertinent data should be returned to the programmer.

Experience indicates the best approach when system time is available is to rerun a failing job under the supervision of the person or persons responsible for analyzing the problem. This quickly determines any procedural or operating problem and gives you first-hand knowledge of the events leading up to a specific problem.

Many times what one person feels is irrelevant is the key to the problem. If your programmers after thorough analysis conclude that IBM's programs are not meeting specifications or a new release level produces a change in a program's operation, contact the applicable IBM representative.

If your programming system is not at a current level, check the latest announcement letters. The problem may be fixed by updating your system. Many

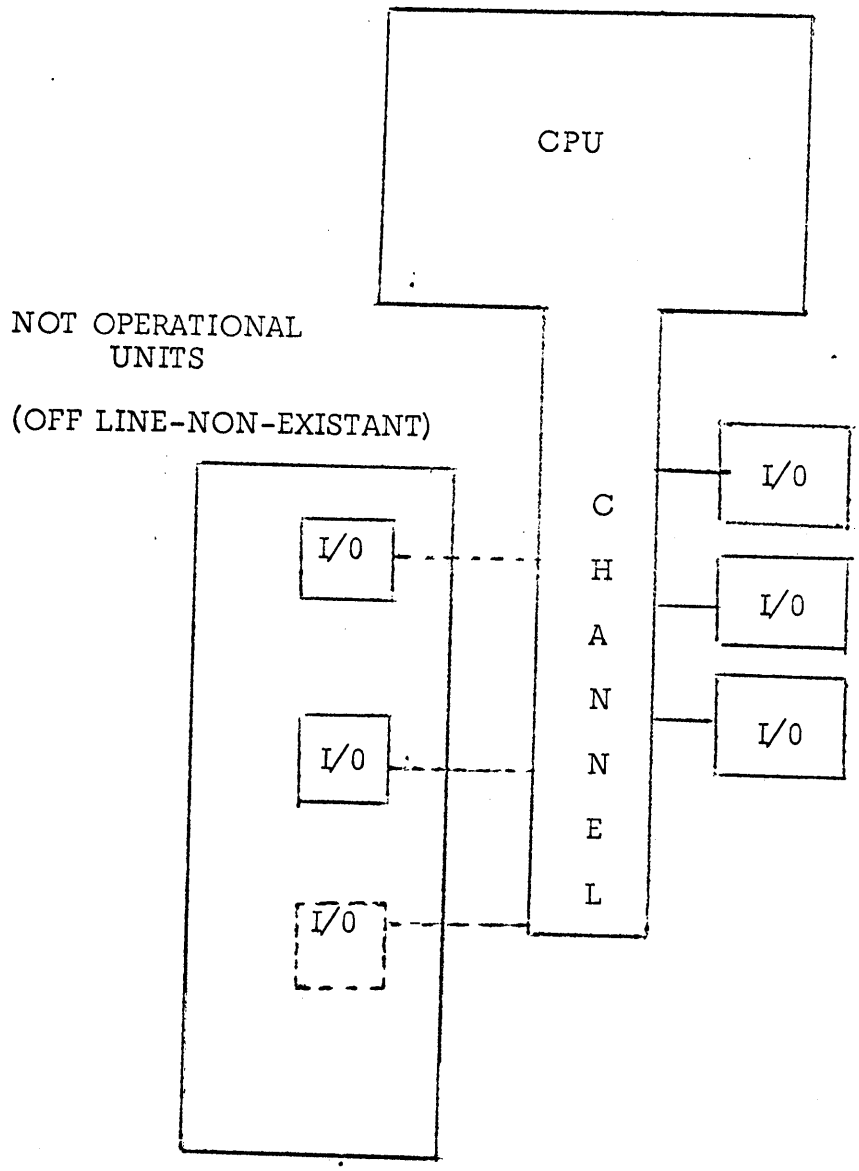
problems are now fixed with product temporary fixes (PTF's). These are patches or modules available to correct problems before they can be included in an official release of the programming system. If no PTF is available Field Engineering will determine a temporary fix or circumvention and submit an APAR to resolve your problem.

A meeting with both your hardware and software customer engineers at installation time to establish adequate PM schedules and error recording, and reporting procedures can insure a maximum system availability. The smooth running of today's system requires quick and accurate diagnosis of hardware, program, or operational problems.

Our best solution to this is continued co-operation.

INVESTIGATION OF ABNORMAL OPERATING CONDITIONS

SYSTEM


CPU
NOT OPERATIONAL
UNITS
(OFF LINE-NON-EXISTANT)
I/O
I/O
I/O
C
H
A
N
N
E
L
I/O
I/O
I/O

CPU PROBLEMS

CHANNEL PROBLEMS

LOG OUT VIA MICRO PROGRAM

a. 12 BYTES MOD 30
b. 256 BYTES MOD 40 UP

NEW MACHINE CHECK PSW PLACES SYSTEM IN WAIT STATE

APPROPRIATE INDICATORS WILL BE SET IN STORAGE

a. OPERATOR
b. SEREP

I/O UNITS

SENSE INFORMATION IS READ FROM THE I/O DEVICE IN ERROR

APPROPRIATE ERROR RECOVERY IS ATTEMPTED

TAPE READ ERROR

a. TIE IF POSSIBLE
b. BACKSPACE REREAD
c. TAPE CLEAN ROUTINE

TAPE WRITE ERROR

a. BACKSPACE -- ERASE -- REWRITE

DISK ERROR

a. RETRY
b. IF DEFECTIVE TRACK SEEK TO ALTER NATE - RETRY

UNIT RECORD

a. OPERATOR ACTION AT UNIT
b. OPERATOR RESPONCE TO SYSTEM CONTROL PROG

TAPES

MUST ALWAYS BE PROPERLY INITIALIZED

TAPE MARK AS FIRST RECORD

VOLUME AND HEADER LABLE
FOLLOWED BY TAPE MARK

OPEN ALWAYS CHECKS FOR VOLUME LABLE
FOR MAXIMUM USER PROTECTION

UNREADABLE RECORD WILL RESULT IN
UNRECOVERABLE I/O ERROR

7 - 9 TRACK TAPES

ONLY READ ERROR FREE ON RESPECTIVE
DRIVES

a. INCLUDES TAPE MARKS

MARK ALL TAPES CLEARLY

EXAMPLE: 7 TRACK MODE x '90'
7 TRACK MODE x '68'

WORK TAPES SHOULD BE PERMENENTLY ASSIGNED

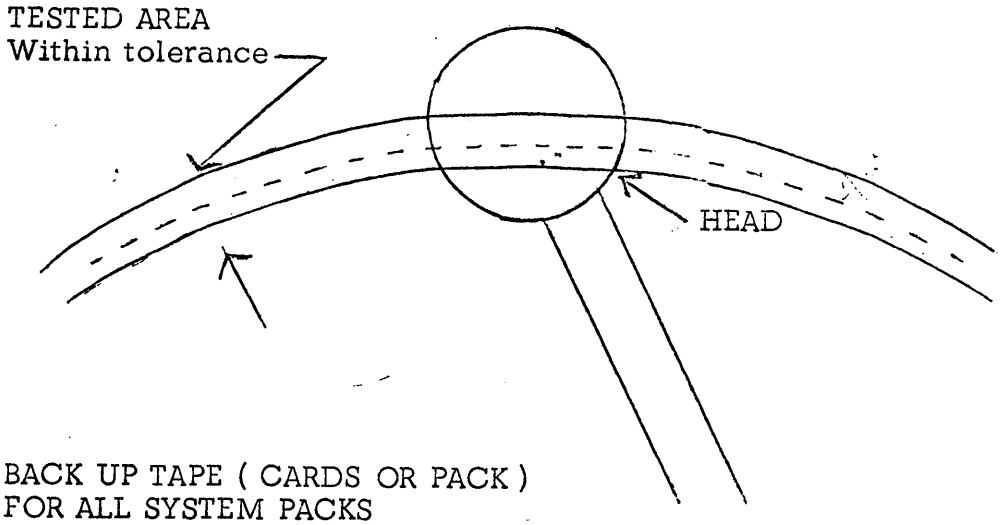
EXAMPLE: 7 TRACK WORK DRIVE 183

DISK

INITIALIZE ALL PACK PRIOR TO USE

FLAG ALTERNATE TRACKS AS INDICATED ON TEST LABLE

FAILURE TO DO THIS MAY RESULT IN UNPREDICTABLE ERRORS ON THOSE TRACKS



BACK UP TAPE (CARDS OR PACK)
FOR ALL SYSTEM PACKS

CARDS

HIGH USE DECKS PLASTIC EDGED

MASTER DECKS FOR REPRODUCTION ONLY

> BEST PROTECTION AGAINST PROGRAM THAT WORKED YESTERDAY - PROGRAM CHECK OR MALFUNCTION TODAY

CHECK PUNCH REGISTRATION PERIODICALLY WITH CARD GAUGE

> DON'T OBSERVE PRINTING ON CARD FOR REGISTRATION

READER CHECKS REQUIRE CHECKING WITH CARD GAUGE

S ystem

E nvironment

R ecording

E dited

P rintout

PROGRAM TO PROVIDE PRINTED ERROR INFORMATION AT TIME OF FAILURE

LOG OUT AREA INFORMATION

a. 12 BYTES TO 256 BYTE DEPENDING ON CPU MODEL

CORE LOCATIONS 0 - 128

a. OLD PSW'S
b. NEW PSW'S
c. CAW
d. CSW
e. GP REGS
f. FP REGS

STAND ALONE CARD DECK
SUPPLIED BY CUSTOMER ENGINEER

ESPECIALLY USEFUL WHEN CUSTOMER ENGINEER NOT ON SITE

PROGRAMMING SYSTEM WILL REQUEST SEREP RUN

CPU FAILURES

CHANNEL FAILURES

BPS - BOS UNRECOVERABLE I/O ERROR

BPS - BOS NOT OPERATIONAL UNIT

INVESTIGATION

ON ALL ERRORS GATHER ALL AVAILABLE DATA PRIOR TO CONTINUING OPERATION

LOOK IN APPROPRIATE CORE LOCATION INDICATOR BYTES

BPS - BOS BYTE x '32'

DOS - TOS BYTES 0 - 3

ONLY INDICATION MAY BE SYSTEM STOPPED IN WAIT STATE ALL INTERRUPTS DISABLED

TAKE SEREP RUNS WHEN REQUESTED

IF SEREP NOT REQUESTED TAKE STAND ALONE CORE DUMP

SAVE: CORE DUMPS
CONSOLE LOGS
PRINTED OUT PUT
INPUT DATA
PROGRAM DECK
TAPE

DOCUMENT THE FAILURE AND IF POSSIBLE SET UP A RUN THAT WILL REPRODUCE THE FAILURE

ONLY AFTER FULL DOCUMENTATION SHOULD CIRCUMVENTION BE ATTEMPTED

PROGRAM SUSPECT

RETRY NEW COPY FROM MASTER DECK

RESTORE SYSRES (DISK - TAPE)

TAPE ERRORS

CLEAN DRIVE

TRY TAPE ON OTHER UNIT

NEW TAPE

DISK ERRORS

NEW PACK

OTHER DRIVE

RUN JOB ON SIMILAR SYSTEM

HARKWARE FAILURE - NOTIFY CE

BEST APPROACH IS TO RUN JOB UNDER SUPERVISION OF ALL CONCERNED

a. HARDWARE CE
b. SOFTWARE CE
c. SE
d. PROGRAMMER
e. OPERATOR

PROGRAMMING SYSTEMS

ANNOUNCEMENT LETTERS

PTF

APAR

INSTALLATION: MEET WITH CE'S (HARDWARE - SOFTWARE)

PM SCHEDUALS

ERROR RECORDING

REPORTING PROCEEDURES

SMOOTH OPERATION REQUIRES

QUICK AND ACCURATE DIAGNOSIS

HARDWARE

SOFTWARE

OPERATION

BEST SOLUTION

COMMUNICATION

CO-OPERATION

M. 4 9

UNIVERSITY OF MASSACHUSETTS

COMPUTER SCIENCE PROGRAM

CURRENT RESEARCH TOWARD THE STANDARDIZATION AND FORMAL DEFINITION OF PL/I

by

JOHN A. N. LEE

Prepared for the Winter Meeting, COMMON Users Group

San Francisco, December 11-13, 1967

ABSTRACT

In April 1966, a subcommittee of USASI Task Group X3.4.2 was established to investigate the standardization of PL/I. Since that time, two working committees have established with explicit charters:

Group I: The Resolution of the Language PL/I

and

Group II: Formal Definition of PL/I

This paper deals specifically with the work of the committee on Formal Definition and its relations with the definitional task forces within IBM. This paper will discuss the techniques of definition as proposed by the committee, the uses to which the definition is to be put and the implications of the work of this committee on the standardization of other computer languages.

A familiarity with PL/I is not presumed.

The need to program digital computers in abstraction as opposed to the subjective wiring of boards has lead to the development of a variety of programming languages. These non-natural communication systems are classified, broadly according to their orientation toward the human user or toward the moronic detailed instructions of a machine. Whereas the human desires a language akin to that which is either his mother tongue or one of the other artificial languages with which he has been associated, many of our current languages have been developed from machine dependent codes towards these desires and have not yet reached the half way point. Although there are many proponents of the concept of ultimately using natural language as the means of describing a computational (or more specifically, a transformational) process, these forms of input suffer from two major defects: a) they are verbose, and b) they suffer from many possible ambiguous contextural translations. Further, we currently do not thoroughly understand the syntax and semantics of natural languages. In any case, mathematical notation (language) has developed over the past few hundred years in spite of the existance of natural language.

Existing analyses of programming language, such as found, for example, in the ALGOL 60 report [1], place emphasis on the syntactic form of the concepts and definitions. In contrast, the approach being proposed by the subcommittee X3.4.2C2 of the USASI, concentrates on the role of constituents of a language as commands or instructions when the program is executed in a machine. This implies that the programming language is to be considered in relation to a machine, be that machine actual or abstract. Classical abstract machine models such as Turing machine or finite automata, which are obvious bases for the definition of language semantics, lack many of the salient features of an objective situation which are relevent to the description of the execution of a program. For example, Turing machines lack the random accessibility of current memories, thus throwing the burden of (say) table look up to the wiles of the definer.

Conversely, some models imitate machines more accurately but are unsuited to the general definition required in the specification of a standard. For example, if there exists an executer (compiler, interpreter, or translator) for PL/I on a particular machine, then that executer is de facto definition. In paraphrasing De'Carte (I think, therefore I am) a compiler can be said to execute and therefore to define. On the other hand, an assembly lising of a compiler cannot be said to constitute a nationwide standard and in any case contains many algorithmic processes of analysis, recognition and (in particular) generation which are irrelevant. Further as the state of the art compilation advances, it must be expected that the efficiency of compilation and the optimization of the generated code will improve, thereby presenting an improved definition.

However, a compiler transforms an input in one non-natural language to another non-natural language which is not a standard. Further only the execution of this target language can reveal the semantic nature of the input (or source) language. We must, therefore, raise the question as to how a natural language description of the syntax and semantics fills the basic necessities of a specification. In particular, a definition which may be used as a standard must possess the following qualities:

- it must be	rigorous complete
- it must display	structure underlying concepts
- it must distinguish between	language defined elements implementation defined elements undefined elements

Current experience with ALGOL 60 [1] and FORTRAN [2] shows that the natural language descriptions of a programming language do not meet all those requirements. Ambiguities exist in the current documentation standards of these languages which have been inserted as a result of the use of a natural language as a descriptor. Witness the list of "trouble spots" reported by Knuth [3].

Similarly, natural language specifications tend to overlook the possibility of the effect of executing a procedure in which the values of certain entities fall outside the domain of the specification. That is, current specifications only define the semantics of a program that exactly fits the task for which it was designed. For example, the FORTRAN specification (sec 7.1.2.1.3) states:

> "7.1.2.1.3 Computed GO TO Statement. A Computed GO TO statement is of the form:
>
> GO TO ($\underline{k}_1$, $\underline{k}_2$ ....., $\underline{k}_n$), i
>
> where the $\underline{k}$'s are statement labels and is $\underline{i}$ an integer variable reference. See 10.2.8 and 10.3 for a discussion of requirements that apply to the the use of a variable in a computed GO TO statement.
>
> Execution of this statement causes the statement identified by the statement label $\underline{k}_j$ to be executed next, where j is the value of $\underline{i}$ at the time of the exeuction."

However, no mention is made of the action to be taken when the value of the integer variable $\underline{i}$ is outside this range. Yet this situation can occur during execution without protection or detection at compile time. In my own shop, we use four different versions of FORTRAN which treat this problem of values outside the range of definition in different manners.

If a description is to demonstrate the structure of a language both at the syntactic level and the semantic level, then the interrelation of both the elements of the language must be fully expanded. For example, consider the interrelation of COMMON, DIMENSION and EQUIVALENCE in FORTRAN. A verbal explanation of this interaction is verbose and contains many its, elses, buts and excepts, the subject phrases of which are not readily accessible. On the other hand, an algorithm for specifying the interrelation of these statements is definitive though lacking the ease of readability.

Yet, why should a specification be generally readable? In no way can we consider that specification or a standard to be a teaching instrument and further,

a specification or standard must be aimed at the designers and inspectors of standards and not the general public. For example, a building code is couched in the standard terms of the civil engineer or the building inspector and is not of general concern to the person using (say) an auditorium. If such users have questions regarding a building code, then they rely on the technical ability of the experts.

Thus a standard specification which is based on the technological tools of the profession is a valid specification. However, in the computer industry there are insufficient people trained to have an understanding of the theory of programming to be able to act as the experts for the general user. This is the fault of our educational system and cannot be used as a deterrent to the development of standards based on the state of the art.

The purpose of a standard definition exceeds the bounds of merely forcing a minimal conformity upon implementers. A definition should also act as an information system to answer such questions as:

> "Is ..... a legal part of the language?"
>
> "What happens if .....?"
>
> "If this element is added to the language, does it create any ambiguities?"
>
> "Does the compiler provided by the manufacturer conform to the standards?"

Thus in some respects a standard must be an information retrieval system, in other respects, it needs to be a model translator and executer. However, there is a point beyond which this standard cannot pass, that is, there is a limit to that which can be thought of as closing the credability gap. For example, arithmetic must be considered to be implementation defined and thus outside the scope of the definition. However, the questions, "What does 0.1 + 0.2 mean?" can be answered as "0.1 + 0.2 is an expression, where 0.1 and 0.2 are real decimal fixed point constants and + is the infix operator representing the operation

of addition." Then the question, "What is the result of evaluating the expression 0.1 + 0.2?" might be "The result of evaluating the expression 0.1 + 0.2 is a representation of 0.3 to the same base and scale of the operands and the precision of the greater of the precision of the operands." Thus a binary machine which evaluates the expression and produces a result of 0.2988 will be conforming to the standard to the same extent as a decimal machine producing the result 0.3. Thus having considered these arguments, the task group reported [4]:

> "It is our belief that with the development of syntax directed compilation as a standard technique, the formalization of PL/I will aid in any standardization that takes place not only by providing a document for people to read but also by providing part of the actual input to be used in the construction of a PL/I compiler. While we recognize that an implementer may choose not to build his compiler in this way, we at least will simplify matters for those who make this choice."

This approach was not based on an abstract concept but as an acceptance in part of the formal definitions of PL/I developed by IBM at Vienna, Austria and Hursley, England. Both definitions are based on the concept of a PL/I machine in which an infinite memory is organized as a combination of tree structures and push down lists. The execution of a program is defined in terms of the changes in the state of the PL/I machine. The two PL/I machines developed by Vienna and Hursley are not identical, that for Vienna being closer to an actual machine than that of Hursley. Fundamentally, the two schemes are similar to the schematic shown in Fig. 1.

The primary portion of the definitions is a Concrete Syntax which specifies the legal statements of the programming language and is based on Backus Naur Form plus special nomenclatures for repetition, grouping and options, and a notation for specifying lists. For example:

program :: = external-procedure

major-prefix-list :: = {(major-prefix-element[,major-prefix...element]...):}...

delete-statement :: = DELETE { , file option . key-option. [event-option]};

Both definitions use this concrete syntax as a specification for constructing legal elements of the language and as a guide for the analysis of the concrete text (input strings). The reversibility of this syntax is still in question and eventually it may be found necessary to replace the single definition, by a syntax for the construction of legal elements of the language and another as a guide to the analysis of concrete text.

Given a concrete text which satisfies the rules of construction and which has been analyzed to show the syntactic functions (components), the input must be analyzed next to show the underlying structure. Since much of the structure is evident from the order in which syntactic components are recognized in the analyzer, the tasks of analysis, recognition and generation may be combined into a single transformational process, the target of which is an intermediate language. In the Hursley definition, this intermediate abstract text is a tree-like structure in which all implied declarations, elements resulting from default conditions and functional relationships are added so that the text is complete. This target text is described by an abstract syntax which is itself a tree-like structure. The translator, which accepts the concrete text and under the direction of the concrete and abstract syntacti develops the abstract text, performs several tasks:

- parses the concrete text
- removes redundancies (punctuation, comments, etc)
- reorders internal procedures, declarations, etc
- expands attributes
- makes all names fully qualified
- inserts declarations for labels and entry names
- forms a complete set of attributes for each identifier by observing:
    - defaulting rules
    - contextual declarations

- built in functions
- implicit declarations

Once a valid abstract text has been created, the semantics of this text are demonstrated by the use of an interpreter which shows the changes in the state of the PL/I machine. The interpreter uses as its input, the abstract text and the data sets.

Although the authors of these definitions are to be complimented on the completion of a magnificent task, certain special requirements and differences of opinion necessitate the reorganization of these definitions to conform with the need to develop a standard. Initially, it was recognized that if a formal style of definitional standard is to be proposed then it must be based on a standard notational method applicable to all languages. In particular, it was felt that the concept of specialized abstract machine such as PL/I machine, or a FORTRAN machine, or an ALGOL machine, could not be proposed as the standard but differing constituent of each specification. Instead, a standard machine should be developed which would be capable of "executing" any programming language and associated data sets. Further, if a standard is to be capable of answering questions regarding both the syntax and semantics of a language, the system must be capable of not only analyzing input strings but also synthesizing strings given (say) an abstract text.

On the premise that a standard should be implementable as a software system, the task group chose to propose that the intermediate text should be a canonic program which is closely associated with the syntax of the input text. The schematic of the proposed standard is shown in Fig. 2. This schema utilizes several mechanical (algorithmic) components which are to be common to all language definitions and which, therefore, must be developed as part of the tools of standardization. Therefore, the descriptors of the language being defined must be developed and standardized themselves.

The concrete and abstract (or canonical) syntacti can be expressed in a modified Backus Naur form, or at least some variant of which is expected to be acceptable as a standard. The actual syntacti metalanguage being developed for the X3.4.2C2 task group, is a linear machine readable metalanguage of which the following statements are examples (see the previous IBM specifications):

<program> → external procedure

<major prefix list> → 0,I*<major prefix element> 0,I*<major prefix element>:))

<delete statement> → DELETE (3,3* [<file option |<key option>|(0,1* event option>)])

where the enclosure of metacomponent names in braces eliminates many of the problems of ambiguity in the metalanguage. The parenthesis groups are of the general form of an implied DO list in a FORTRAN input/output statement with the variation that the limits of the repetition precede the list and the increment is always unity--thus the construct <n>→(1,3*A) means that the metaresult <n> may be formed from 1 to 3 concatenations of the terminal A. The same metalanguage may also be used to define the syntax of the canonical intermediate text and it is further anticipated that a transformational grammar can be developed which uses the same basic nomenclature.

However, the terminology of the analyzers, synthesizers and the executer cannot be readily expressed in Backus Naur Form though some variant of AMBIT [7] would seem to be appropriate. The "execution" of the input text using the schema shown in Fig. 2 is merely one approach to the problem of definition. An alternative technique would be to define a fundamental language in detail according to Fig. 2 and then to define transformational grammars on other languages to transform them to this fundamental language. The choice of this fundamental language could be one of the biggest political problems to face the data processing section of USASI.

In fact, if PL/I is all it is purported to be, then it is the obvious fundamental language. However, the growing band of advocates of APL (or Iverson

Notation) have a candidate for this position which has already been shown to be adequate for the formal description of a machine [5] and hence should be adequate as the target language of syntax directed translator.

The type of linear executer described by the Hursley and Vienna definitions and by definition executer of X3.4.2C2 or the translator described in the previous paragraph, cannot adequately describe a language which is self modifying or self generative. However, a feed back loop from the "executer" or "interpreter" to the syntax analyzer will solve this problem.

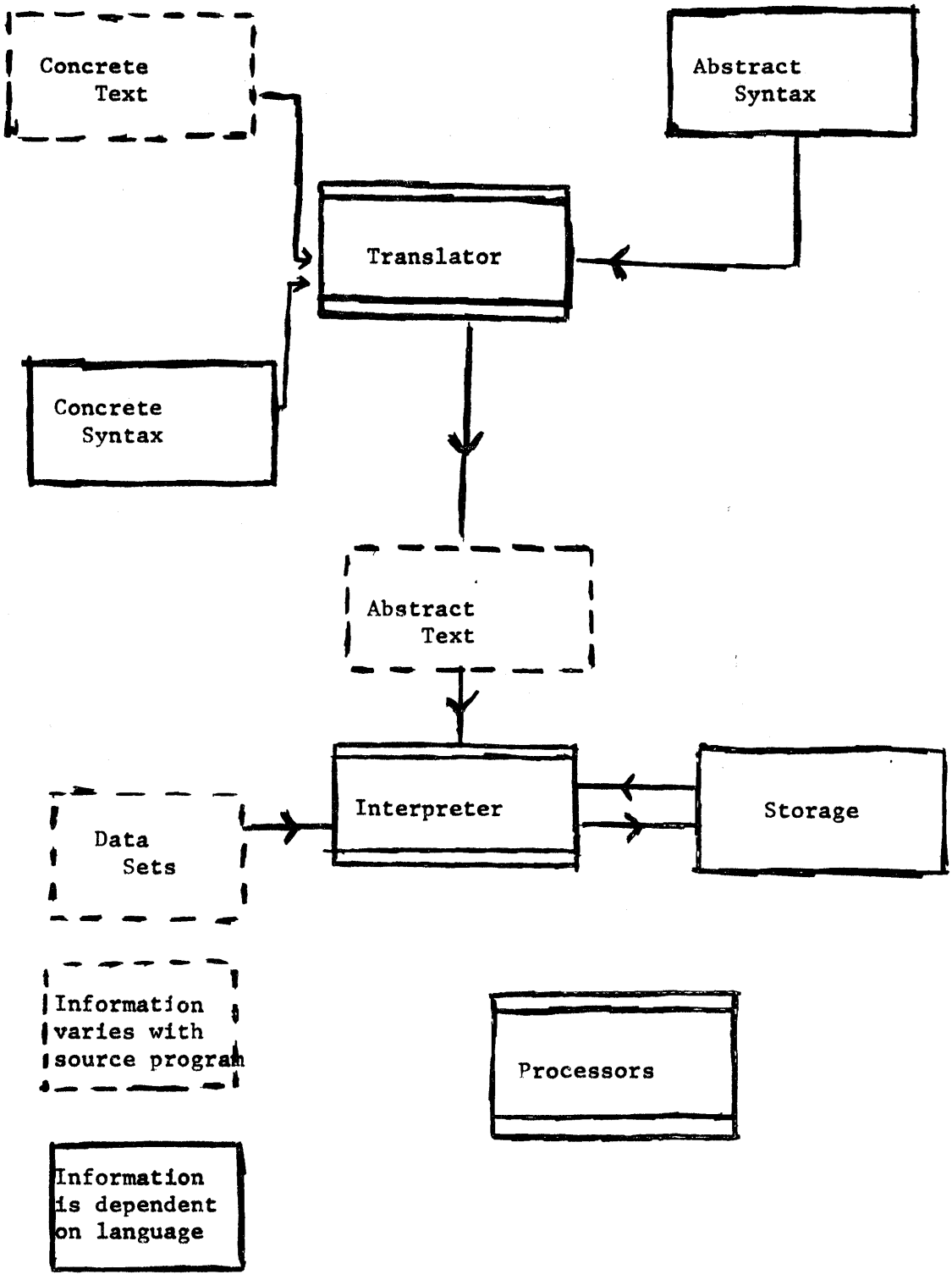
On the face of it, a definition executer does not appear to be as efficient as a standard compiler for the straight forward task of recognizing an input structure. This conclusion is based on the premise that no matter how efficiently the specification is designed, the analyzer portion of the executer will spend some considerable percentage of its time exploring blind alleys. Further, the keys of string recognition may be camouflaged beneath the wealth of associate syntax, every element of which must be verified before even a preliminary recognition can be made. For example, the first three characters of each BASIC [6] statement are a key to the type of statement following. As another example, a compiler implementer has at his disposal certain machine dependent information which he may utilize to improve the efficiency of the recognizers. In particular, the sorting order of characters permits the testing of characters in a relative manner so that a binary tree can be built, in which the length of the maximum branch (representing maximum number of comparisons to be made in recognizing a character) is a minimum. The maximum tree branch needed in a USASI FORTRAN compiler for the recognition of a keyword is five. A syntax directed translator does not have this ability and thus must chain through a linear sequence of alternatives till the list is exhausted. That is, for example, to recognize that the character 1 is not an element of the alphabet requires only five comparisons using a binary tree search in a compiler (Fig. 3) but 26 comparisons in a

syntax directed analyzer. Under these conditions, a definition executer may never replace the specialized compiler and with the given purpose of definition, the speed and efficiency of execution is not in competition. As programming languages become more permissive, the task of defining a language in prose will become more difficult and there will be a corresponding increase in the complexity of a compiler. However, the definition execution and syntax directed translator can handle this expectation.

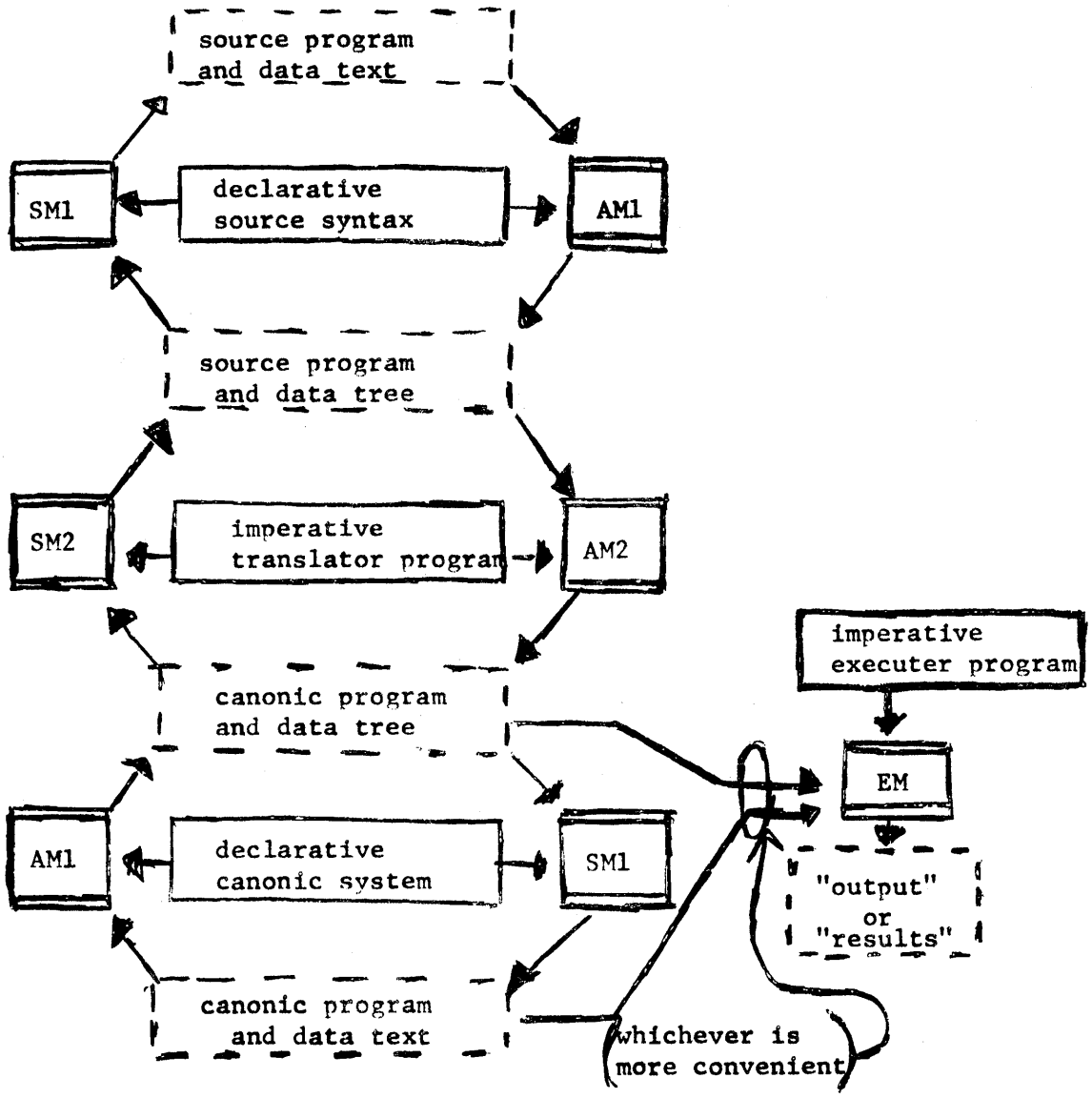
SUMMARY

In this paper, I have tried to explain the concept of a language definition based on the techniques of syntax directed compiling and translation. Whilst a standard definition of a computer language may be incomprehensible to the majority of the programming community of today, those who are concerned with implementing systems conforming to the standards will be members of the minority (by definition). A definition executer as described herein will not only define the syntax and semantics of a language but also outline the translatory techniques of a compiler. In comparison, a building code contains parallel partitions whereby the design criteria and techniques of construction are specified.

Programming languages are intended to be languages of precision. Yet we are relying currently on languages of imprecision to specify the syntax and semantics of those same programming languages. If for no other reason than for consistency and perhaps to maintain the image of the industry, the formal definition of language will replace the current prose definitions.



IBM Definition Schematic
(Fig. 1)



Key:

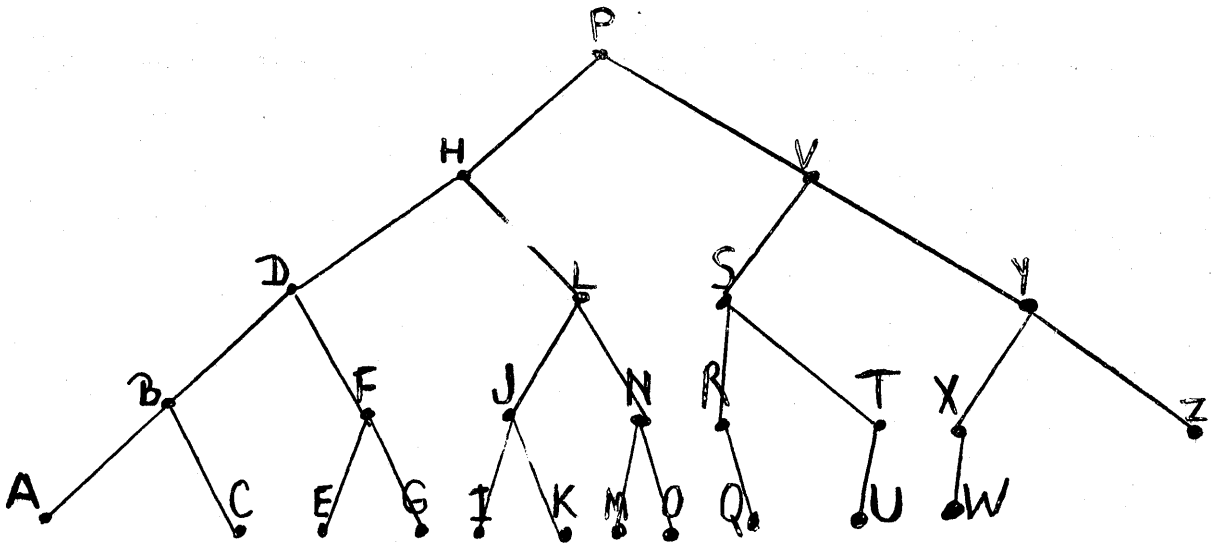
Information varies with source program

SM = Synthesizer Machine
AM = Analyzer Machine
EM = Executer Machine
(These are fixed over all languages)

Information varies with language being defined

DIAGRAM FOR THE X3.4.2C2 DEFINITION EXECUTER

(Fig. 2)



Alphabetic Recognition Tree

(Fig. 3)

REFERENCES

1. Naur, P. & Woodger, M. (Eds). Revised report on the algorithmic language ALGOL 60, Comm. ACM V6 pp 1-20.

2. Anon. FORTRAN vs Basic FORTRAN, Comm. ACM V7 pp 590-625.

3. Knuth, D. E. The Remaining Trouble Spots in ALGOL 60, Comm. ACM V10 pp 611-618.

4. Abrahams, P. et al. Report of Ad Hoc Task Group on Formal Definition, Private Communication to X3.4.2C2, May 18, 1966.

5. Folkoff, A. D. et al. A Formal Description of System/360, IBM Systems Journal #3, 1964.

6. Lee, J. A. N. The Anatomy of a Compiler, Reinhold Publishing Company, New York, 1967 Appendix C.

7. Christensen, C. Examples of Symbol Manipulation in the AMBIT Programming Language, Proc. 20th Natl. Conf. ACM, Cleveland, 1965.

T.11

A "SMALL" OS SYSTEM

As Described to Wade A. Norton (1125)
By Max L. Allen (1164)

Rust Engineering (1164) of Birmingham, Alabama, and Pittsburgh, Pennsylvania, a company which specializes in engineering design and construction, has recently become a division of Litton Industries.

Their work includes (1) Design and construction of process mills—paper mills being a good example, (2) Renovation of office buildings, (3) Maintenance and construction work for government, and (4) Design work for government, including some launch facilities at Cape Kennedy.

Rust has offices at Calhoun, Tennessee; Montreal, Quebec, Canada; Vancouver, B. C., Canada; Brussels, Belgium; and Mexico City, Mexico—in addition to their principal offices in Birmingham and Pittsburgh.

Rust employs about eight hundred people each in the Birmingham and Pittsburgh offices; and, even though the mix in the jobstream is quite different at the two installations, they are able to generate a single OS system to meet the requirements at both places.

In Birmingham the mix is approximately 65%-35% engineering vs. business data processing, whereas in Pittsburgh the ratio is about 15% to 85% (in favor of commercial work). Some of the work in Birmingham —quite properly classified as engineering—is engineering management information, rather than design or analysis.

In Birmingham it is possible to schedule very little of the work because report periods vary for different contracts. Near the end of a report period, they likely are making several runs with their management info tools.

In Pittsburgh one of their big jobs is labor costs, which is often run several times a day.

However, staffing is not too different in the two localities, with each employing about ten or twelve analysts and programmers. In Birmingham they have two machine operators; in Pittsburgh, four for two shifts, therefore employing two per shift just as in Birmingham.

On the other hand, there is a distinct difference in the number of keypunch operators—due to the difference in the amount of input data. In Birmingham three keypunch operators are employed by the computer center, whereas in Pittsburgh (where the mix is 15%-85% commercial) the accounting department has ten to twelve keypunch operators on its payroll.

You may ask—and properly—why I present this paper. I am the first to agree that the real author is Max Allen of Rust's Birmingham computer center.

Unfortunately, he has been unable to make either the Cincinnati meeting or this one. In Cincinnati this paper was presented by title to the OS Committee. That group agreed that the subject matter is far more appropriate to the full S/360 Project. Thus it appears on the agenda for today.

Rust's computer department exhibits a certain rugged individualistic spirit—of just the sort we want displayed by COMMONers—to dig into the uncharted and build there something of real value.

And, having mapped the previously unknown, they have demonstrated to the fullest a second trait we wish of all COMMONdom—the willingness to share with others their knowledge, their system, and, indeed, test time.

I can truthfully say that we would most likely be in some stage other than the final one of conversion from a 1620 system to S/360-40 OS, had it not been for Rust's pioneering of OS and their willingness to share with us the fruits of a then hard-won prize. (My only reluctance at making the last remark is that it might be construed in some quarters as implying that this paper is in payment of a moral debt. That is not the case! At no time has Rust ever implied that somebody owed them anything; furthermore, this paper was requested by your former OS Committee chairman, Mrs. Barbara F. Young.)

And certainly, Rust's work deserves reporting—OS is for the MOD 30 and MOD 40, and both COMMON and IBM need to be told this fact.

The remainder of this paper fills in the details as told to me by Max Allen of the Rust computer center (and observed by me in the course of Southern Services' conversion efforts).

Max's position at Rust is as the Number One man on the systems and applications side of the activity, and in the absence of the manager he assumes these functions, also.

This manager, who was sweating the dollars and cents of OS pioneering while Max perspired over the bytes and bits of the work, is Bill Mylius. Bill is in charge of computer facilities at both Birmingham and Pittsburgh. And, while this story is basically one of implementation and told by Max, the policies reported in the telling of the story are Bill's.

Rust tried DOS and it did not give the flexibility they needed. This was especially true with respect to structuring of programs larger than one core load. CALL LINK under DOS FORTRAN helps, but it does not solve things. In reality it creates greater rigidity.

Another initial consideration in the choice of OS is that the Project Management program and ICES (MIT) were both written to run under OS. Later developments of ICES push it out of 64k and only further justify the choice.

Core requirements for OS of slightly less than 16k were evaluated against both the 6k DOS requirement and the 10k needed for DOS-2 with fixed scheduler partitions. It was decided, Max said, that the additional core devoted to OS more than paid for its dedication in a more powerful system, "particularly since the linkage editor is so flexible in how one may structure his program."

Still another item which influenced Rust's decision at the time was that they were enthused over the prospects of using fullblown PL1. Their experience to date has been that it is clumsy in both compile time and run time, but very easy for the programmer. At the time (November 1966), PL1 was not supported under DOS. It is known that PL1 has been and is continuing to be improved. That as yet they have not gone to it as their primary language has not caused them to regret their choice of OS.

In fact, they would make the same choice today—OS over DOS—which they made more than a year ago. (And, I might add, today they would have a lot more support from IBM in its implementation.)

Rust considers the following as disadvantages of OS or, more properly, as part of the price paid for the more powerful system:

1. There is lots more to learn about OS than about any other system for driving a 360.

2. OS can do lots more, but this flexibility also creates problems of choice.

3. To applications programmers, JCL presents great problems. There are no clear-cut logical patterns to follow.

4. Operator requirements are not easy for any method of driving S/360. It is a complicated machine. But with OS, the training of operators is considerably more difficult.

5. Rust paid the price, also, (in November 1966) which IBM's lack of field knowledge about OS cost. IBM Birmingham was reluctant to send their people to school. Rust's CE worked on OS when he had nothing else to do, and APAR answers were such as, "It looks like you got a bad tape." Rust started to SYSGEN on November 8, 1966, with Release 6. They obtained their first running version on December 23, 1966, with Release 7. In between there was much effort expended—mostly by Rust people—before IBM finally imported a man from outside the local office.

Actually IBM did not really begin to support OS in Birmingham until two Mod 40 installations—Alabama Power Company and Liberty National Life Insurance Company—were being implemented. (Southern Services, Inc., (1125) whom your speaker represents, has merged its operations with Alabama Power, an affiliated company.) It turned out that much of this trouble, which took more than a month to find, was caused by the Linkage Editor which was losing a CSECT out of the nucleus.

6. When Max went to OS school in Cincinnati, there was not a single course offered by the New Orleans region. Because of the aerospace industry, some training in OS had been offered in Huntsville, but this was not under region sponsorship. In Max's opinion, the course at Cincinnati was a good course. Since that time, IBM has been schooling their people regularly and offering courses on a broader geographical basis. The know-how is available now; the marketing emphasis for OS with the Mod 30 and Mod 40 is lacking. In both Max's and my opinion, this is something IBM should correct.

Among the things they bought were

1. Improved Job Management.

2. Better Data Management.

3. More powerful Linkage Editor.

4. Identical systems for Birmingham and Pittsburgh even though the hardware and addressing differed slightly. In Birmingham their punch is a 1442 addressed at 00A, while in Pittsburgh it is a 2540 addressed at 00D.

5. Under OS, Rust has found that it is almost impossible for an operator to lose control of OS by a misstep at the console.

6. Combining items 2 and 3 has made it possible to catalog programs and data and put them on disks for exchange between Birmingham and Pittsburgh.

7. Without the power of OS to put programs elsewhere than on the system residence volume, Rust's programs would have required multiple system residence packs a la the 1620.

Rust did not run extensive timing tests, because they were not felt necessary in weighing OS against DOS. Enough was done (with programs which constituted a large part of the workload) to bear out the already known generality that OS FORTRAN(E) compile-link and compile-link-go were both superior to their DOS equivalents. They used only FORTRAN because this was the only language in which they felt sufficiently

proficient at the time. They have since used COBOL some, too, while they wait for a better-performing PL1. They checked compile-link using a Plant Material Thermal Balance program, which is their biggest job. They found that OS did the job in 8 minutes compared to 30 minutes for DOS. Using a Steel Stack Design program, they found a compile-link-go advantage of 22 percent while using OS.

When they started, all they wanted was enough system to do some testing (and it was the hardest to come by). In the interval December 1966 to December 1967, Rust generated five releases. SYSGEN is now easily done in a manner that suits Rust's needs better than the cookbook method described in the SYSGEN manual. However, this is stated as a plus for Rust's experience and not as a criticism of following the book—which procedure will get your OS for you.

You can have a basic system for about 14k of core storage. This will include PCP (the sequential scheduler with all options transient instead of resident), support for a reader, printer, punch, one disk (for system), and four tapes. It would include the access methods BSAM, QSAM, and enough BDAM to access programs and compilers. It would not include a timer, indexed sequential access methods, nor direct access methods for applications data sets.

Rust's system requires $3F08_{16}$ bytes (or $16{,}136_{10}$). To be exact, it is slightly more than 16M bytes, but slightly less than 16K bytes where $K = 1024 = 2^{10}$. In this amount of storage, Rust provides for PCP, interval timer, three disks, no tapes, two punches (as described above), a 2501 card reader, and a 1403 printer. They have the standard access methods QSAM, BSAM, BPAM, BISAM, QISAM, and BDAM. They have resident SVC entries and trace table entries. If IBM ever supports just a plain timer, Rust will take it over the interval timer and thereby save a little more core.

The full direct access method, the indexed sequential access methods, and the trace table entries were included, because Rust believes they buy enough in supporting FE work to justify their inclusion.

At the moment, Rust can do the work required of their system. But, looking to the future, they see more core needed, greater processor speed, and more devices to support. And they look upon their experience with OS as a great boon to future growth. A possible first step might be the addition of a second selector channel, if spooling under MFT-2 is shown to give reasonable performance at the 65k level.

Presented to the 360 Project
at the San Francisco Meeting
by Wade A. Norton, COMMON 1125

Applications

T. 1. 7 (a)

1) Working title of the paper: The 1620 as a Data Collector or Software, the Crutch Hardware Boys Lean On

2) Author: Guy A. Gallaway

3) Address and User code: Sacramento Peak Observatory
Sunspot, New Mexico 88349
User #5053

4) Position: Programmer

5) Time requirements: 30 minutes

6) Special Equipment: Projector for 3 1/4 by 4 inch glass mounted (lantern) slides.

7) Level: Intermediate

8) Machine: 1620 II was used, but the presentation will be for general information.

9) Abstract: The paper will cover the data collection methods used at the Sacramento Peak Observatory. The principle reasons for designing a computerized data collection system were:
1) Improve existing methods which used summary punches.
2) Gain experience and insight into problems associated with data collection in preparation for third generation equipment.

The paper will also cover the roles played by the programming staff in this type of operation. Namely, software as a diagnostic tool for the hardware boys in developing data reduction/collection instruments, and software as a useful and flexible tool for the research scientist.

Three methods for data collection and reduction by computer have been tried.

1) On line, one point at a time, testing each point for parity and structural errors and storing it on the disk before the next point is generated.

2) On line, a record at a time, locking the source device out until the data has been tested and stored.

3) Off line, using 7 track incremental tape recorders, with testing and reduction being done during slack times.

SESSION T.1.7

NAME	USER #	PHONE	COMPANY REPRESENTED & ADDRESS
R. E. Buckley		652-2694	Monsanto Co., Decatur, Ala.
G. F. Schoditsch	3438	(314) MA1-4000	Monsanto Co., 1700 2nd St., St. Louis, Mo. 63177
George Polyzoides		285-6655	Hooker Chemical Corp. Niagara Falls, N.Y.
Theo. E. Bridge	1100	(215) KI5-7500	Catalytic Const. Co 1528 Walnut Philadelphia, Pa
E. J. Uczen		(815) 895-5194	Anaconda Wire & Cable N. Cross Street Sycamore, Ill. 60178
S. Barr		(212) 571-3977	Western Electric 222 Broadway New York
N. Kurek		(213) 341-3010	Canoga Electronics Los Angeles
D. Adams		(415) 961-1111	Ames Research Center Moffett Field, Ca
A. J. Bims		(805) 296-6681	Ryan Aero 2701 Harbor Drive San Diego

SESSION T.1.7

NAME	USER #	PHONE	COMPANY NAME & ADDRESS
Michael Robert Hale		(213) WE3-7356	Human Factors Research 1112 Crenshaw Blvd. Los Angeles, Calif.
R. D. Rosenstein	1117	(412) 621-3500 Ext.7138	Crystallography Lab. Unive of Pittsburg Pittsburg, Pa 15213
R. L. Ferral		(916) 442-1201	USWB Sacto RFC Sacramento, Calif.
J. W. Wheeler		(919) 199-2311 Ext.357	Western Electric Co. 801 Merritt Drive Greensboro,N.C.
L. V. Punder	7072		Biological Station Nanaimo British Columbia Canada
Joseph Sloboda		(313) 238-1631 Ext. 450	Flint Community Junior College Flint, Michigan 48503

For Figures

1620 DATA COLLECTION

or

SOFTWARE, A CRUTCH THAT HARDWARE BOYS LEAN ON

Guy A. Gallaway

Sacramento Peak Observatory
Air Force Cambridge Research Laboratories
Sunspot, New Mexico 88349

Presentation at the Winter Meeting of

COMMON

in San Francisco, California, 12 December 1967

1620 DATA COLLECTION

(or, Software, a Crutch That Hardware Boys Lean On.)

This paper will describe, in varying degrees of detail, three different methods of data collection. Each of these methods was developed and used at the Sacramento Peak Observatory, Sunspot, New Mexico. The Observatory is part of Air Force Cambridge Research Laboratories and performs basic research in solar physics. Two of the methods were on-line to a computer and the third, still in use, used an incremental tape recorder as intermediate storage.

The overall goal of the project was to experiment and compare the use of real-time collection with remote collection on an intermediate device, such as mag tapes. We originally intended to continue this experimental approach with our third generation equipment before committing ourselves to either method. But now it looks like we will be doing a combination of both.

To this end we felt we should implement these systems on an established machine. In this way both the hardware and software people could, and did, gain experience and insight into the problems involved.

The immediate goals were to develop a more reliable system and establish a complete operating system. An earlier method made use of a summary punch and was very unreliable. We also wanted to try and develop some systematic procedures for handling this type of data, and to have available a standard set of reduction/utility routines for processing the data. But this approach usually meets with opposition from the individual scientist who wants a "tailor-made" system for his project, the definition of which will usually change every few months. Since we are planning on implementing the same devices on newer machines, this attitude means that we are faced with the propsect of never-ending systems development. We have found that after the initial development an equal amount of time will be spent in making changes to meet specific desires.

The goals of the system were always being jeopardized by frequent modifications in "mid-stream". From the time we started to implement data collection, up until now (last week) there have been continuous changes to the system. These cover the full range from the data format to reduction procedures. None of the systems were developed as a unit and then examined and evaluated. This lack of a commitment to full development has been demonstrated to waste considerable manpower and machine time.

The source of the data was photographic films, typically filtergrams and spectrograms, of various solar phenomena. We use a scanning microphotometer to read the films. Film densities are converted to a 3-digit decimal number and sent to the computer

or the tape recorder. The computer used was a full 1620 II with binary features. The two on-line systems employed the paper tape channel for data transfer. We have an on-line plotter, but no conflict occurs since the plotter uses the write paper tape instruction and collection uses the read paper tape instruction. Termination of the data was done by sending a record mark on the End of Line (EOL) line.

SYSTEM #1

The first method was the only one requiring a real time response. The characteristics of this system were:

1. A 1-2-2'-4 BCD format.
2. One data point at a time transmission.
3. Packed data.
4. Termination of the data string by a counter control.

This strange BCD form, 1-2-2'-4, was a "left-over" from the previous system that used a pulse counter with this format. This added the problem of encoding the data to the normal data handling problems. This was the first instance of a hardware problem for which the software provided the solution. Figure 1 defines the formats used and shows how the paper tape lines were used. A minimum of 20% of data handling time was devoted to the encoding. The data was stored on the disk in one sector blocks of 31 points. Only every other point was flagged making the data appear as 16 6-digit words. ($\overline{1}2312\overline{3}123123$)

The coice of two points per word of data was made to achieve minimum disk storage and still insure disk compatability with Fortran. That is, given a fixed point word size of six, and the array IDATA dimensioned at 16, the statements, RECORD (J) IDATA and FETCH (J) IDATA, will reference one sector of 96 digits. The unpacking of the data was done in a Fortran subroutine.

The maximum data rate, 20 points per second, left ample time available for the programming required to handle each point. Total programming time requirements using the longest encoding routine, worst case, took only 2640 microseconds. The greatest time component was disk revolution time, worst case conditions being 44 mils. Figure 2 gives a detailed description of the timing involved.

Originally there was to be an ID number sent as the first data point. But in order to speed up the process of implementing the system we used the ID number as an input argument between the main-line Fortran program and the collection program. This ID number was then the first point of each disk block, making each block 32 points long.

Termination was to be done three ways. Two of these were similar, both using sense switches. The setting of one switch would indicate that the collection had been completed normally and reduction should begin. The other switch would indicate an operator error and for the current collection to cease and re-start. The third method, and the only one implemented, was by having the user specify in his Fortran program the number of points to be collected. This number was another input argument for the collection routine.

What turned out to be the most significant advantage in getting the system going was the "off-the-shelf" hardware. Our engineers are "time shared" with the whole Observatory. They are responsible for building and maintaining a very large number and variety of complex equipment. And as frequently happens under such circumstances the person that yells the loudest gets his work done the fastest. So by having working hardware we were able to complete the system sooner than if the hardware had to be built.

The disadvantage of this system was its slowness - not only the data rate but the over-all system. Considerable time was required in the Fortran programs to get the data from the disk and unpack it. This, however, is a familiar story to programmers, minimizing storage at the expense of execution time.

One problem in the early stage of development was the 1620. We had learned from IBM that each character signal should be maintained 10+ microseconds. The first try was a signal of 12μ, which got "lost" in the CPU. A diligent pursuit of the signal with scope and probes convinced the EE's of the soundness of their design. The data left their black-box and entered the computer. To them it was unfortunate that the program, or programmers, couldn't find the data. Even elaborate overlay and dump routines failed to completely convince them that the data wasn't reaching core. Finally, the extension of the signal to 20μ's maintained the characters long enough for them to reach core. We were then able to collect data.

SYSTEM #2

The second on-line system used new hardware and software. The new hardware was designed and built by our own engineers. We wrote new but similar software. This system had several entirely new features:

1. BCD compatability.
2. Transmission of records of data.
3. A source generated 6-digit ID.
4. Variable data rates.
5. Starting the source device with the computer.

BCD compatability solved the encoding problem and cut down on the required programming.

By allowing the transmission of a record of data we had to limit the total number of points collected in each record. We decided to use 1/2 of the available core, 24,000 digits. This was equal to 4000 data points three digits long, each point being converted to a six digit fixed point number. Conversion to the six digit size was done in the collection program. Considerable time was saved by eliminating the slow unpacking routines in Fortran. We still limited the size of the data blocks on the disk to one sector of 16 points.

The hardware generated ID number was an aid to the file management problem. It was to be the first six digits of the data string. The number would be chosen by the user and selected on thumb wheel switches, and read out of a register.

Also, a control was installed to allow the user to digitize at a variety of rates from 1 to 100 points per second.

The collection of variable length records required the "locking" out of the source device. This was to be done by allowing the device to send only upon receipt of a computer generated signal. The Exponent Overflow indicator was chosen as this trigger. This would require turning the Exponent Check light, EXP CK, on for 5 mils immediately preceeding the read instruction.

The software was quickly written and debugged due to the similarity with the previous system. The construction of the hardware took considerably longer, mostly due to the manpower problems mentioned before. And before the system was operational there were more than a few changes in definition.

The first problem was the EXP CK light. The 5 mil signal was not long enough to turn on the data source. A testing routine was quickly written to vary the time and 250 mils was determined to be the shortest reliable length.

The next problem was the ID number. The ID register read out too slowly. For any data rate greater than 20 points/second the first few data points would be lost. Designing a faster read-out didn't completely solve the problem, so the ID number was moved to the end of the data string, the last six digits.

Moving the ID solved the problem of losing points but caused another problem, which is still in the system. For some undetermined reason the ID number is frequently, in half or more of the cases, seven digits long. In the majority of these the extra character appears to be the seventh, last, digit. But there are many times when the first and second digits are the same, both flagged, making the first character appear to be the extra one. But with either case there are six good digits which can be selected, by programming for the ID.

Here were two hardware problems, ID generation, and device control which required a software solution for implementation.

In the early development of this system there were random parity problems. Several test programs were written to display these characters and examine them with the binary instructions. There were also structural errors in the data, that is, one of the three digits would be dropped. The design was changed and these problems reduced. (They still occur, infrequently, and are a good indication of some temporary problem in the hardware.)

We were still having problems with the EXP CK trigger and decided to use the computer logic of the read instruction. With the able help of our IBM CE this was done, and with considerably less effort than expected.

SYSTEM #3

This was now compatible with tape recorder design and the recorder was hooked into the system.

The person collecting data could now work at any time, using the computer or the tape recorder. This was, operationally, a better policy since the computer wasn't tied up for long periods collecting data. The tapes were reduced in the evenings and during the day when the computer wasn't in use. The same software handled the data tapes by using the read paper tape instruction.

With the installation of the tape recorder we were also able to reduce binary tapes from a different source. By this time we had learned not to write a full software package before the hardware had evolved into its final form and was debugged. Two short routines were enough to reveal various failures in the hardware. These problems were similar to the other system, lost points and digits, and a new one, a data point in the middle of the ID information.

USE OF THESE SYSTEMS

The second system was modified for one user to eliminate the use of the disk. He collected a constant 500 points. The collection program was changed to format the points and place them directly in the COMMON area of Fortran. This was a considerable time-saver for his application and he was able to collect about 7 million points in one month. This was the only production work done using either system.

Most users wanted their data plotted as soon as possible to determine if they were operating the equipment correctly. Since the Fortran plotting was pretty slow we wrote a short machine language program to read the data and plot it directly from core. As the data

is defined as ranging from 0-999 no scaling was required. The result was plotting done four times faster than in Fortran. We were now inspired to try and hook an oscilloscope onto the computer for even faster plotting. We even went so far as to develop programs to convert the plot commands into a continuous string of digits corresponding to the pen commands. In this way plotting was done by dumping core, with a pen command coming every 10μ. Manpower shortages prevented the hardware boys installing the scope.

The amount of time required in developing utility and debug routines for the hardware was greatly underestimated. Until the hardware worked, a programmer had to be available every time the hardware was tested. One four hour session resulted in the writing of five short machine language programs. In addition an immeasurable number of dump routines were composed and executed at the console typewriter. I do not mean to cast aspersions on the hardware fellows. Their basic design has proven very reliable. It was a lot of fun, and very satisfying, to build the system into a usable tool. The point is, both the hardware and software people must work closely together from the very beginning. It is not unreasonable to expect the consulting of some computer oriented person when designing any data system, since the data will eventually reach a computer.

Our error was in planning on just writing the software package and not budgeting for time to write the programs to debug the hardware. They could have done it with scopes and a bread-board, but a more reliable "unit" is developed when both sides are involved in the operation.

Of the three methods we used, the best, in my opinion, was clearly the use of the tapes. The procedures the user goes through to set up the source device are usually very time consuming. And he will often be reducing more than one type of data at a time, requiring additional set-up time. When the computer was on-line it would often be used less than 50% of this set-up time and never used fully. Production runs could often be done only in large blocks of time, four or more hours. This forced the user to use the computer at night. And more importantly, the on-line system required someone familiar with the machine and programs to be available at all times. (Typical reason for this was the user would send too much data and clobber the program, but he still wanted to save the data in core. This and similar problems required the presence of a programmer during data collection.)

The advantages of the tape system were mostly operational. The user was only competing with other microphotometer users, 1 or 2 people at most and frequently no one, instead of many computer users. The scheduling for the microphotometer was done by the day instead of by the hour as for the computer. The user needed only to know how to run one machine instead of two. And most

importantly, more efficient use was made of the computer. During the set-up period the user could use the computer in 5 to 15 minute blocks as required to test his data and operating procedures. During production runs the bad records can be identified in advance so the programs will ignore them, a considerable time saver. Also the reduction of the tapes can be done by a competent computer operator, with the special cases (extra long records) being handled by special programs, without loss of data.

These reasons would be less important with a machine capable of handling foreground/background programs, but cannot be entirely dismissed. Generally speaking, the data would still be placed on an intermediate device such as a disk or tape before reduction. Under such circumstances careful consideration must be given to each method. Cost is not a small factor. For a few collection operations the extra cost of a computer hardware to do the job would be proportionally higher, when compared to individual devices and controllers at each site. But for a large number of such applications, which can be mutliplexed, the cost is reduced for each device, when done with a computer.

Another important consideration is site locations. Transmitting data at high rates and control of collection devices over large distances is still pretty expensive. This tends to force the collection instruments to be located near the computer. When this is impossible then the on-site system using an intermediate device is one solution.

One way of evaluating a system is by asking, "Would you do it again?". There is no doubt that we would, and gladly. The one production run mentioned before, with 7 million points collected, would have been impossible to do under the previous system, in less than four months, if at all.

We now know, more clearly, what to expect of the basic collection routines, what to expect them to do and what not to have them do. Our new computing equipment will be in use a minimum of five years, probably much longer. Those of us in programming are desperately hoping that the research scientists will completely develop a working system before changes are even discussed. The single biggest factor in extending the development time of these systems was changes in the basic definition of what the system would be.

The only advice I would offer to anyone considering doing this type of work with a General Purpose machine is to see it through to the end, and then evaluate the whole system. Even if you have unlimited manpower and time to continue re-doing the same thing, it is much easier to work with a system which has a few, well-defined, operational peculiarities, than to re-educate every user every time he uses the system.

Sacramento Peak Observatory

Data Collection System #1

Source	Paper Tape
=	EL
4	X (flag)
*	0
Check	Check
*	8
2'	4
2	2
1	1
*	not used

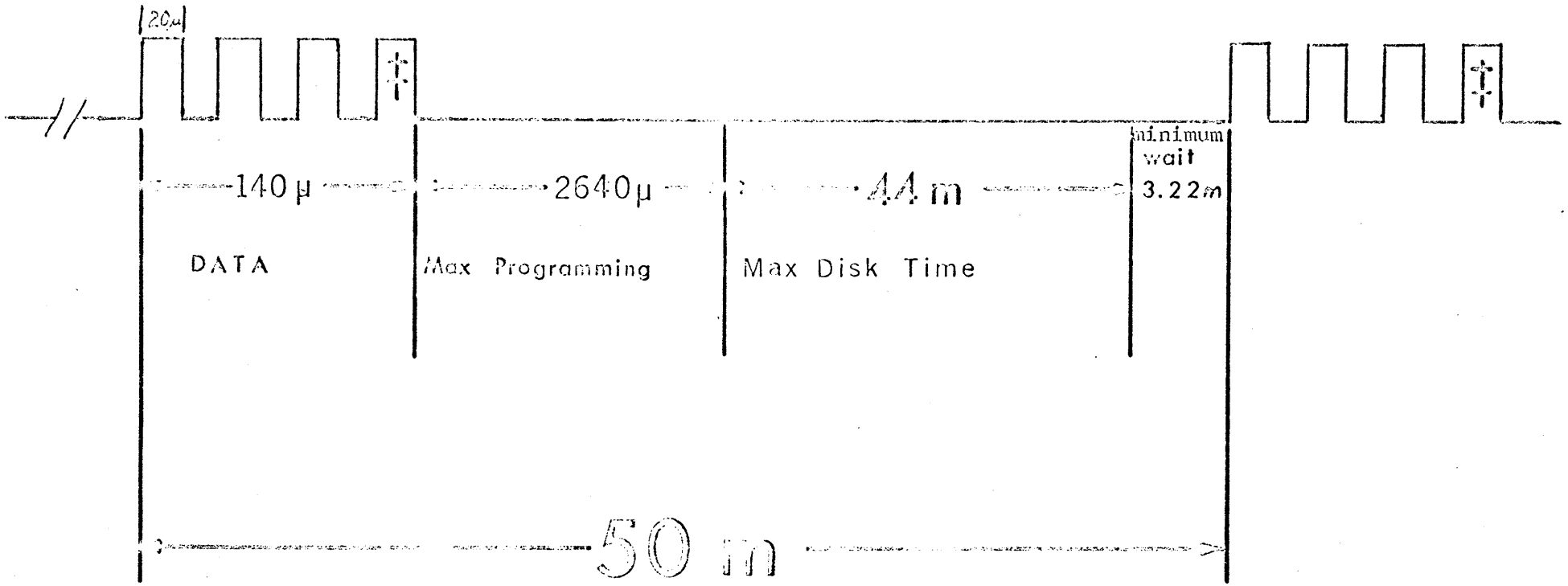
Conversion	
0 - 0	270μ
1 - 1	"
2 - 2	"
3 - 3	"
6 - 4	440μ
$\overline{7}$ - 5	380μ
$\overline{4}$ - 6	260μ
$\overline{5}$ - 7	"
$\overline{6}$ - 8	"
$\overline{7}$ - 9	"

Overhead/Point	1320μ
Worst case point 3*440 =	1320μ
Worst case Programming	2640μ

Digits	Normal BCD Representation	1 2 2' 4 Representation	Paper Tape Channel Input Scheme	Read As:
	1 2 4 8	1 2 2' 4	1 2 4 F	
0	0 0 0 0	0 0 0 0	0 0 0 0	0
1	1 0 0 0	1 0 0 0	1 0 0 0	1
2	0 1 0 0	0 1 0 0	0 1 0 0	2
3	1 1 0 0	1 1 0 0	1 1 0 0	3
4	0 0 1 0	0 1 1 0	0 1 1 0	6
5	1 0 1 0	1 1 1 0	1 1 1 0	$\overline{7}$
6	0 1 1 0	0 0 1 1	0 0 1 1	$\overline{4}$
7	1 1 1 0	1 0 1 1	1 0 1 1	$\overline{5}$
8	0 0 0 1	0 1 1 1	0 1 1 1	$\overline{6}$
9	1 0 0 1	1 1 1 1	1 1 1 1	$\overline{7}$

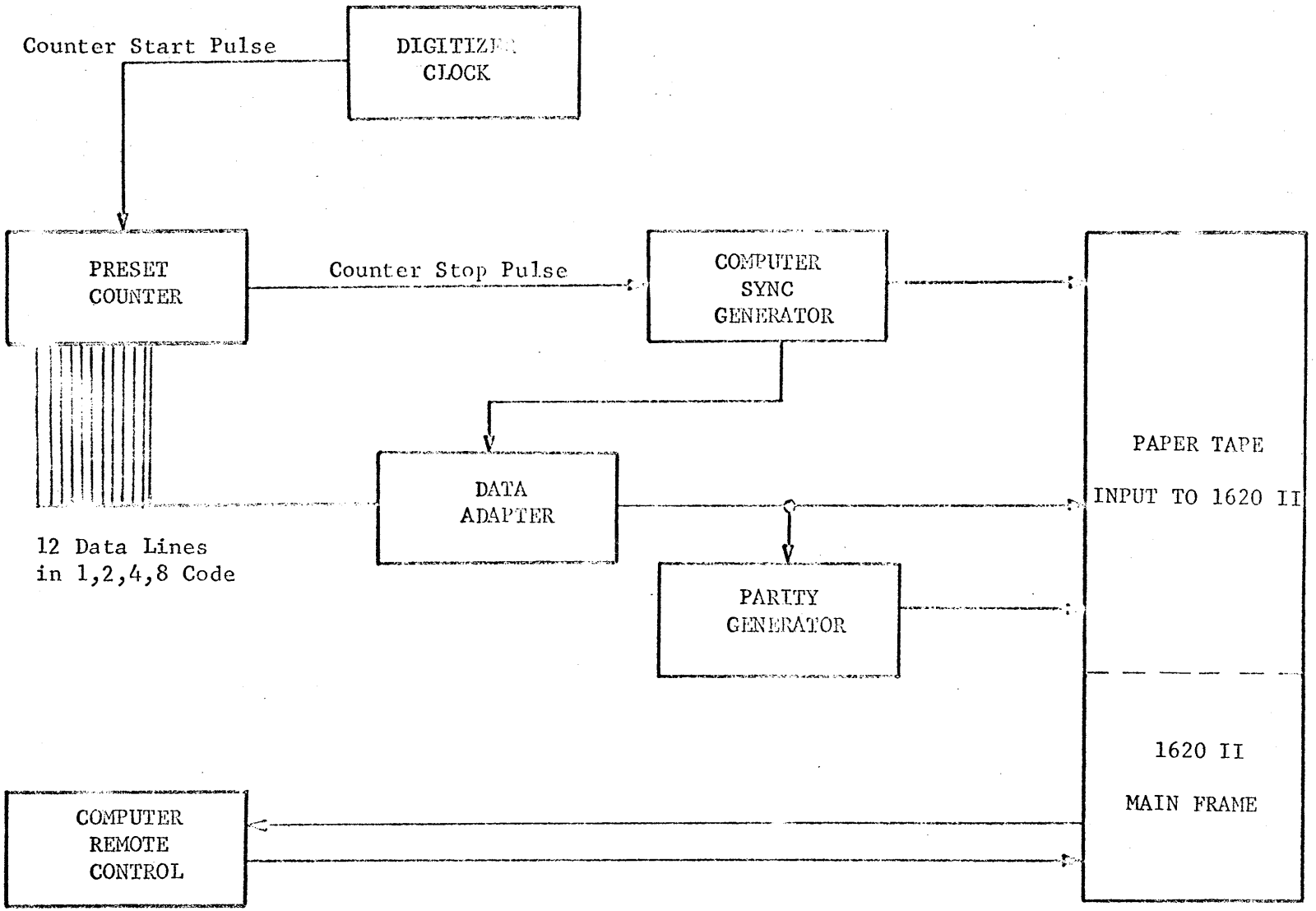
Figure #1

NOT TO SCALE



267

Figure 2


Counter Start Pulse
DIGITIZER CLOCK
PRESET COUNTER
Counter Stop Pulse
COMPUTER SYNC GENERATOR
PAPER TAPE INPUT TO 1620 II
DATA ADAPTER
12 Data Lines in 1,2,4,8 Code
PARITY GENERATOR
1620 II MAIN FRAME
COMPUTER REMOTE CONTROL

Catalytic Construction Company
1528 Walnut St., Phila., Pa. 19102

- Programs for Inverting a Large Matrix on a Small Machine -

by

T. E. Bridge

In figure 1, we have plotted, time to invert a matrix, versus the size of the matrix. We have plotted results for two programs, one using partitioning, and the other using sequential files. The same form of test matrix was used in all cases. Double precision arithmetic (17 digits) was used,

Size of Matrix	25	60	120
Partitioning Method (Table 2)			
Time Minutes	2	19	115
RMS Error %	0.000	.356	1.252
Sequential Files (Table 3)			
Time Minutes	3	29	213
RMS Error %	0.000	0.000	
Direct Access Files (Table 6)			
Time Minutes	4	29	231
RMS Error %	0.000	0.000	0.001

In this paper, we are presenting three programs:

Table 2 To invert a matrix by partitioning the maximum size is 352.
Table 3 To invert a matrix using sequential files. The maximum size is 300.
Table 6 To invert a matrix stored on disk. The maximum size is 300.

Two other programs that were used in testing the above three programs are also given:

Table 4 A program to build a large matrix for testing.

Table 5 A program to multiply the test matrix by its invert, and then compare each element with the corresponding element in the unity matrix. The RMS error is calculated and published.

[over please]

Description of Partitioning as Used by the Program in Table 2

The following formulas, that we use in Table 2 to partition the matrix, are given in "Linear Algebra", by G. Hadley -- Addison-Weekly Publishing Company, 1961 -- on page 109.

A large matrix may be partitioned into four smaller matrices. Let -- a, b, c, d -- represent four small matrices arranged like this to form a large matrix:

$$\begin{matrix} a & b \\ c & d \end{matrix}$$

Note that a and d are square matrices; while b and c may not be.

Let the invert of the above matrix be represented by:

$$\begin{matrix} A & B \\ C & D \end{matrix}$$

Let: e = the invert of d.

The following equations may be used to find each partitioned piece of the invert:

$$A = (a - bec)^{-1}$$
$$B = -Abe$$
$$C = -ecA$$
$$D = e - ecB$$

The program given in Table 3 uses these equations to find the invert of a large matrix. We had only 5K bytes of storage -- 629 double words -- left over with the partitioning program in core. Since 100 words are required for data handling, we find that the large matrix must be partitioned down to a -- 23 x 23 -- before it can be inverted. So, four levels of partitioning would be required to invert a -- 352 x 352 -- matrix.

The levels of partitioning required are:

Matrix size	23	46	92	184	352
Levels required	0	1	2	3	4

We found that the minimum amount of partitioning gave the shortest amount of run time when using this progrma. In other words, it took 20% longer to invert a 100 x 100 matrix using 4 levels rather than the required 2 levels of partitioning. We don't really know what would be the optimum amount of partitioning on a larger machine.

CLEARPRINT PAPER CO. No. T323. LOGARITHMIC: 2¼ BY 2 3/4-INCH CYCLES
PRINTED IN U.S.A. ON CLEARPRINT TECHNICAL PAPER NO. 1000H
TECHNICHART

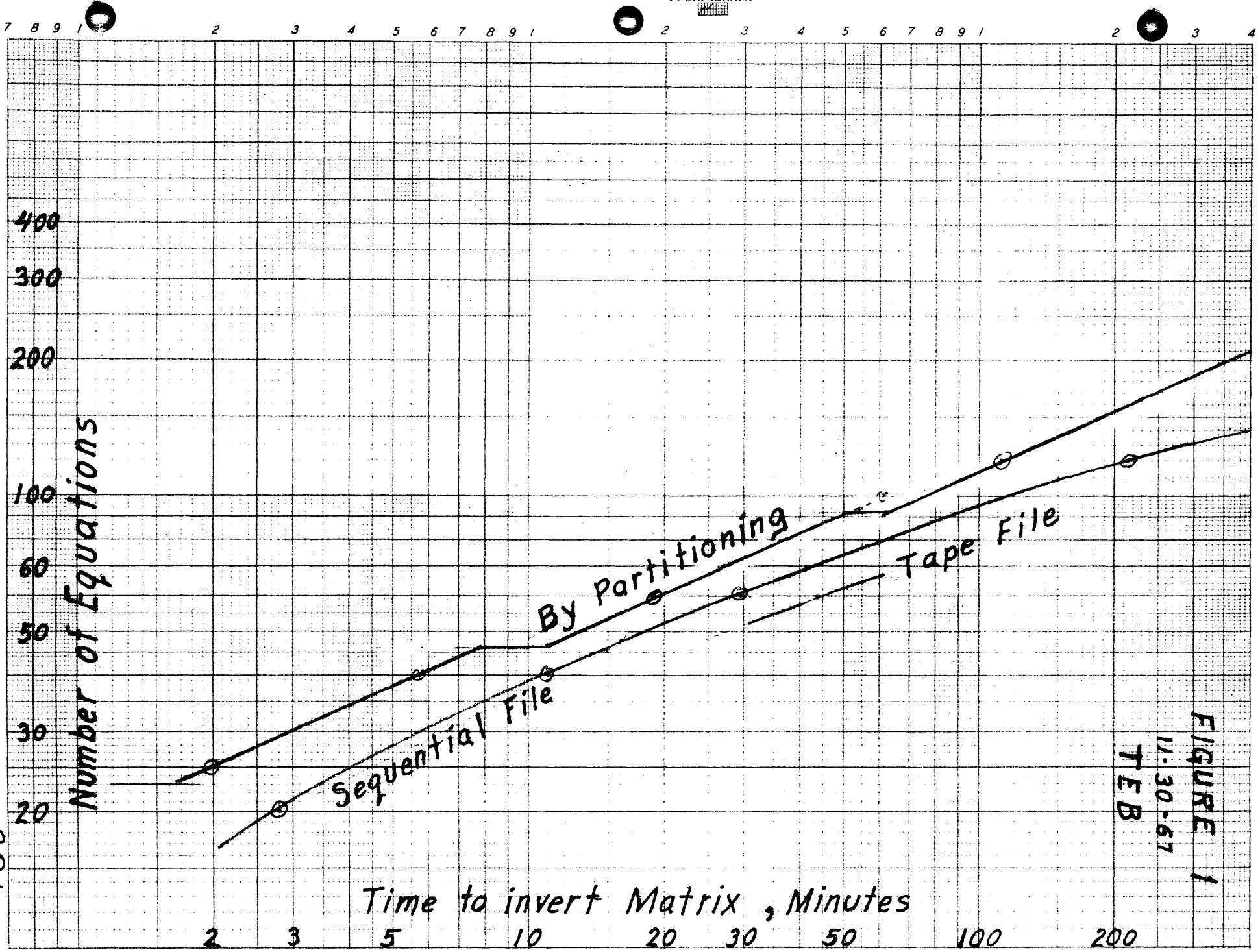


FIGURE 1
11-30-67
TEB

TABLE 1 LISTING OF PROGRAM

THE FOLLOWING PROGRAM IS INCLUDED ONLY TO TEACH THE METHOD OF OPERATION USED IN THE PROGRAMS LISTED IN TABLES 2 AND 3

```
C     THIS PROGRAM WILL SOLVE  N  SIMULTANEOUS EQUATIONS
C
      N1 = N + 1
      DO 1 K = 1,N
      F = X(K,K)
      DO 1 J= K,N1
    3 X(K,J)= X(K,J) / F
      DO 1 I = 1,N
      IF (I-K) 2,1,2
    2 X(I,J) = X(I,J) - X(K,J) * X(I,K)
    1 CONTINUE
      END
```

STATEMENT 2 ABOVE IS THE HEART OF THE PROGRAM. IT IS IN THE INNER LOOP AND IS EXECUTED N ** 3 TIMES

EQUATION K IS THE KEY EQUATION

THIS IS DONE IN STATEMENT 3 .

2) IN STATEMENT 2 , EVERY TERM IN THE KEY EQUATION IS MULTIPLIED BY X(I,K) AND THEN SUBTRACTED FROM THE CORRESPONDING TERM IN EQUATION I THIS WILL GENERATE A ZERO IN COLUMN K. EACH SWEEP OF THE MATRIX WILL GENERATE A COLUMN OF ZEROS IN COLUMN (K) EXCEPT FOR THE TERM ON THE DIAGONAL WHICH WILL BE 1.0 .
BY PERFORMING A SERIES OF VALID OPERATIONS (NOT CHANGING THE EQUALITY) WE GENERATE THE UNITY MATRIX. THE ANSWERS WILL THEN APPEAR IN THE LAST COLUMN .

```
   TABLE  2  LISTING OF PROGRAM TO INVERT A LARGE MATRIX STORED ON          0040
             DISK USING PARTITIONING                                       0041
                                                                           0042
// OPTION CATAL                                                           0043
   PHASE BJPL$04,S                                                         0044
// EXEC FORTRAN                                                           0045
      COMMON N3, N4, N5, N6,JIM,NM,IT,IS,NJ,NP,NC,NX                       0046
C    N3  IS A WORK FILE ON DISC                                            0047
C    N4 IS A TEMPORARY SEQUENTIAL FILE                                     0048
C    N5  IS A SEQUENTIAL FILE FOR STORING MEMBER INFORMATION               0049
C    N6 IS A DISC FILE IN WHICH MATRIX IS STORED                           0050
C    JIM   IS THE SIZE OF THE SMALL MOSAIC OF WHICH THE BIG MATRIX IS MADE 0051
C    NM  IS THE NUMBER OF MEMBERS                                          0052
C    IT  IS FILE TO WHICH PROGRAM WILL RETURN                              0053
C    IS   WILL CAUSE DIAGNOSTICS TO BE PRINTED                             0054
C    NP  IS PAGE NO                                                        0055
C    NC  IS NO OF COLUMNS IN BIG MATRIX                                    0056
C    NX  IS NO OF COLUMNS IN COMPRESSED MATRIX                             0057
      DEFINE FILE  12(1600,200,U,I2)                                       0058
      DEFINE FILE 13( 800,200,U,I3)                                        0059
      DOUBLE PRECISION X (529)                                             0060
      IF(NX -  3) 1,1,2                                                    0061
    1 CALL TB$1 ( NX, 1,X)                                                 0062
    3 CALL LINK (05)                                                       0063
    2 IF(NX-46) 4,4,5                                                      0064
    4 CALL TB$11 (NX,1,X)                                                  0065
      GO TO 3                                                              0066
    5 IF(NX-92 ) 6,6,7                                                     0067
    6  CALL TB$12(NX,1,X)                                                  0068
      GO TO 3                                                              0069
    7 IF(NX-184) 8,8,9                                                     0070
    8 CALL TB$13 (NX,1,X)                                                  0071
      GO TO 3                                                              0072
    9 IF(NX-352) 10,10,11                                                  0073
   10 CALL TB$14(NX,1,X)                                                   0074
      GO TO 3                                                              0075
   11 WRITE (3,100)                                                        0076
      CALL EXIT                                                            0077
  100 FORMAT (  20H MATRIX TOO LARGE          )                            0078
      END                                                                  0079
/*   TEB                                                                   0080
// EXEC FORTRAN                                                           0081
      SUBROUTINE TB$1(N,NC, X)                                             0082
      DOUBLE PRECISION X(23,23),F                                          0083
      COMMON N3, N4, N5, N6, NX                                            0084
      I3 = 1                                                               0085
      NP = NC + N - 1                                                      0086
      DO 13 I = 1,N                                                        0087
      K = NC + I - 1                                                       0088
      CALL DATA (1,1,X,I3, NC, NP, K)                                      0089
   13 I3 = I3 + 23                                                         0090
      DO 1 K = 1,N                                                         0091
      IF (X(K,K))2,1,2                                                     0092
    2 F = 1.D0/ X(K,K)                                                     0093
      X(K,K) = .1D1                                                        0094
      DO 3 J = 1,N                                                         0095
```

```
    3 X(K,J) = X(K,J) * F                                              0096
      DO 1 I = 1,N                                                     0097
      IF (I-K) 4,1,4                                                   0098
    4 F = X(I,K)                                                       0099
      X(I,K) = 0.D0                                                    0100
      DO 1 J = 1,N                                                     0101
      X(I,J)= X(I,J) - X(K,J) * F                                      0102
    1 CONTINUE                                                         0103
      NC1 = NC-1                                                       0104
      NP = NC & N -1                                                   0105
      I3 = 1                                                           0106
      DO 14 I = 1,N                                                    0107
      K = NC1 + I                                                      0108
      CALL DATA ( 2,1,X,I3, NC, NP, K)                                 0109
   14 I3 = I3 + 23                                                     0110
      RETURN                                                           0111
      END                                                              0112
/*    TEB                                                            0113
// EXEC FORTRAN                                                        0114
      SUBROUTINE DATA( JIM, JOE, A, I3, NR1, NR2, NC)                  0115
      DOUBLE PRECISION A(529) , Z(100)                                 0116
      COMMON N3, N4, N5, N6                                            0117
      IF(N5) 31,30,31                                                  0118
   30 WRITE(3,32) JIM, JOE, I3, NR1, NR2, NC                           0119
   31 CONTINUE                                                         0120
   32 FORMAT (/6I8/)                                                   0121
      NF = N6                                                          0122
      NB = 0                                                           0123
      GO TO (1,2,3), JOE                                               0124
    3 NB = 200                                                         0125
    2 NF = N3                                                          0126
    1 CONTINUE                                                         0127
      NK = NC*4-4                                                      0128
      NRB = NR1 -1 + NB                                                0129
      NRE = NR2 - 1 + NB                                               0130
      I = I3                                                           0131
      LB = MOD(NRB, 100) + 1                                           0132
      LE = MOD(NRE, 100) + 1                                           0133
      ME = NK + NRE/100+ 1                                             0134
      MB = NK + NRB/100 + 1                                            0135
      DO 33 M = MB,ME                                                  0136
      LS = 1                                                           0137
      LF = 100                                                         0138
      IF (M-MB) 11,11,12                                               0139
   11 LS = LB                                                          0140
   12 IF(M-ME) 14,13,13                                                0141
   13 LF = LE                                                          0142
   14  READ (NF'M) Z                                                   0143
      GO TO (6,7) , JIM                                                0144
    6 DO 10 L = LS, LF                                                 0145
      A(I ) =Z(L)                                                      0146
   10 I = I + 1                                                        0147
      GO TO 33                                                         0148
    7 DO 15 L  = LS, LF                                                0149
      Z(L) = A(I)                                                      0150
   15 I = I + 1                                                        0151
```

```
      WRITE(NF'M) Z                                                    0152
   33 CONTINUE                                                         0153
      IF(N5) 42,41,42                                                  0154
   41 J = I3 + NR2 - NR1                                               0155
      WRITE (3,100) (A(K) , K = I3, J  )                               0156
  100 FORMAT ( 1P10D13.6)                                              0157
   42 RETURN                                                           0158
      END                                                              0159
/*    TEB                                                             0160
// EXEC FORTRAN                                                        0161
      SUBROUTINE TB$11 (N,NA,X)                                        0162
      DOUBLE PRECISION X (23,23)                                       0163
      NL = N/2                                                         0164
      NU = N - NL                                                      0165
      ND = NA + NU                                                     0166
      CALL TB$1(NL, ND, X )                                            0167
      CALL TB$7(N, NA, X)                                              0168
      CALL TB$1(NU, NA, X)                                             0169
      CALL TB$8(N,NA,X)                                                0170
      RETURN                                                           0171
      END                                                              0172
/*    TEB                                                             0173
// EXEC FORTRAN                                                        0174
      SUBROUTINE TB$7(N,NA,X)                                          0175
      DOUBLE PRECISION X(176,3)                                        0176
      NE = NA & N - 1                                                  0177
      NL = N / 2                                                       0178
      NU = N - NL                                                      0179
      ND = NA & NU                                                     0180
      ND1 = ND - 1                                                     0181
C  W2 = B * C                                                          0182
      K1 = 1                                                           0183
      DO 3 K = ND,NE                                                   0184
      DO 1 L = 1,NU                                                    0185
      DO 2 J = 1,NL                                                    0189
    1 X(L,3) = 0.D0                                                    0186
C     D  IN  (2)                                                       0187
      CALL DATA (1,1,X,177,ND,NE,K)                                    0188
C     B IN (1)                                                         0190
      CALL DATA (1,1,X,1,NA,ND1,J+ ND1 )                               0191
      DO 2 I = 1,NU                                                    0192
    2 X(I,3) = X(I,3) & X(I,1) * X(J,2)                                0193
C     (3) IN w2                                                        0194
      CALL DATA (2,2,X,353,1,NU,K1)                                    0195
    3 K1 = K1 + 1                                                      0196
C   A = A - W2 * C                                                     0197
      DO 6 K = NA,ND1                                                  0198
C     A IN (3)                                                         0199
      CALL DATA ( 1,1,X,353,NA, ND1, K)                                0200
C     C IN (2)                                                         0201
      CALL DATA (1,1,X,177,ND,NE, K )                                  0202
      DO 5 J = 1,NU                                                    0203
C     W2 IN (1)                                                        0204
      CALL DATA ( 1,2,X,1,1,NU,J)                                      0205
      DO 5 I = 1,NU                                                    0206
    5 X(I,3) = X(I,3) - X(J,2) * X(I,1)                                0207
```

```
C       (3) IN A                                                        0208
    6 CALL DATA (2,1,X,353,NA,ND1, K )                                  0209
        RETURN                                                          0210
        END                                                             0211
/*      TEB                                                             0212
// EXEC FORTRAN                                                         0213
        SUBROUTINE TB88 (N,NA,X)                                        0214
        DOUBLE PRECISION X(176,3)                                       0215
        NA1 = NA - 1                                                    0216
        NE = NA + N - 1                                                 0217
        NL = N/2                                                        0218
        NU = N - NL                                                     0219
        ND = NA + NU                                                    0220
        ND1 = ND - 1                                                    0221
C       B = -A * W2                                                     0222
        K1 = 1                                                          0223
        DO 3 K = ND, NE                                                 0224
        DO 1 L = 1,NU                                                   0225
    1 X(L,3) = 0.D0                                                     0226
C       W2 IN (2)                                                       0227
        CALL DATA (1,2,X,177,1,NU,K1)                                   0228
        DO 2 J = 1,NU                                                   0229
C       A IN (1)                                                        0230
        CALL DATA(1,1,X,1,NA,ND1, J+ NA1)                               0231
        DO 2 I = 1, NU                                                  0232
    2 X(I,3) = X(I,3) - X(I,1) * X(J,2)                                 0233
C       (3) IN B                                                        0234
        CALL DATA (2, 1,X, 353, NA, ND1, K )                            0235
    3 K1 = K1 + 1                                                       0236
C       W2 = D * C                                                      0237
        K1 = 1                                                          0238
        DO 4 K = NA, ND1                                                0239
        DO 5 L = 1,NU                                                   0240
    5 X(L,3) = 0.D0                                                     0241
C       C IN (2)                                                        0242
        CALL DATA (1,1,X,177,ND, NE, K )                                0243
        DO 6 J = 1,NL                                                   0244
C       D IN (1)                                                        0245
        CALL DATA (1,1,X,1,ND,NE, J + ND1)                              0246
        DO 6 I = 1, NL                                                  0247
    6 X(I,3) = X(I,3) + X(I,1) * X(J,2)                                 0248
C       (3) IN W2                                                       0249
        CALL DATA (2,2,X,353,1,NL, K1)                                  0250
    4 K1 = K1 + 1                                                       0251
C       C = - W2 * A                                                    0252
        DO 7 K = NA, ND1                                                0253
        DO 8 L = 1,NL                                                   0254
    8 X(L,3) = 0.D0                                                     0255
C       A IN (2)                                                        0256
        CALL DATA (1,1,X,177,NA,ND1, K)                                 0257
        DO 9 J = 1,NU                                                   0258
C       W2 IN (1)                                                       0259
        CALL DATA (1,2,X,1,1,NL,J)                                      0260
        DO 9 I = 1,NL                                                   0261
    9 X(I,3) = X(I,3) - X(I,1) * X(J,2)                                 0262
C       (3) IN C                                                        0263
```

```
C       D = D - W2 * B                                                  0264
      7 CALL DATA (2,1,X,353,ND, NE, K)                                 0265
        DO 10 K = ND, NE                                                0266
C       D IN (3)                                                        0267
        CALL DATA (1,1,X,353,ND, NE, K )                                0268
C       B IN (2)                                                        0269
        CALL DATA (1,1,X,177,NA,ND1, K)                                 0270
        DO 11 J = 1,NU                                                  0271
C       W2 IN (1)                                                       0272
        CALL DATA (1,2,X,1,1,NU,J )                                     0273
        DO 11 I = 1,NL                                                  0274
   11 X(I,3) = X(I,3) - X(I,1) * X(J,2)                                 0275
C       (3) IN D                                                        0276
   10 CALL DATA (2,1,X,353, ND, NE, K )                                 0277
        RETURN                                                          0278
        END                                                             0279
/*      TEB                                                             0280
// EXEC FORTRAN                                                         0281
        SUBROUTINE TB$14 (N,NA,X)                                       0282
        DOUBLE PRECISION X (25,25)                                      0283
        NL = N/2                                                        0284
        NU = N - NL                                                     0285
        ND = NA + NU                                                    0286
        CALL TB$13(NL,ND,X)                                             0287
        CALL TB$7(N, NA, X)                                             0288
        CALL TB$13(NU,NA, X)                                            0289
        CALL TB$8(N,NA,X)                                               0290
        RETURN                                                          0291
        END                                                             0292
/*      TEB                                                             0293
// EXEC FORTRAN                                                         0294
        SUBROUTINE TB$13 (N,NA,X)                                       0295
        DOUBLE PRECISION X (25,25)                                      0296
        NL = N/2                                                        0297
        NU = N - NL                                                     0298
        ND = NA + NU                                                    0299
        CALL TB$12(NL,ND,X)                                             0300
        CALL TB$7(N, NA, X)                                             0301
        CALL TB$12(NU,NA, X)                                            0302
        CALL TB$8(N,NA,X)                                               0303
        RETURN                                                          0304
        END                                                             0305
/*      TEB                                                             0306
// EXEC FORTRAN                                                         0307
        SUBROUTINE TB$12 (N,NA,X)                                       0308
        DOUBLE PRECISION X (25,25)                                      0309
        NL = N/2                                                        0310
        NU = N - NL                                                     0311
        ND = NA + NU                                                    0312
        CALL TB$11(NL,ND,X)                                             0313
        CALL TB$7(N, NA, X)                                             0314
        CALL TB$11(NU,NA, X)                                            0315
        CALL TB$8(N,NA,X)                                               0316
        RETURN                                                          0317
        END                                                             0318
/*  TEB                                                                 0319
```

TABLE 3 LISTING OF PROGRAM TO INVERT A MATRIX STORED ON TAPE

```
// JOB BJPL$
// OPTION CATAL
   PHASE BJPL$13,S
// EXEC FORTRAN
C      BJPL$ PROGRAM TO INVERT UP TO A 300 X 300 MATRIX  STORED
COLUMNWISE IN FILE 12 . FOUR RECORDS ARE USED FOR EACH COLUMN
       COMMON N3, N4, N5, N6,MIJ,NM,IT,IS,NJ,NP,NZ,NC
       DEFINE FILE 12(1600,200,U,IN2)
       DOUBLE PRECISION F,G
       DOUBLE PRECISION C(300), D(300,2), D1(300)
       EQUIVALENCE (D1(1), D(1,1))
       DIMENSION NF(2)
       NF(1) = 4
       NF(2) = 5
       M = 1
       N = 2
       DO 14 J = 1,NC
       J4 = J*4
       READ(N6'J4-3) (C(L), L=  1,100)
       READ(N6'J4-2) (C(L), L=101,200)
       READ(N6'J4-1) (C(L), L=201,300)
   14  WRITE (4) C
       REWIND 4
       READ (4) (D1(L), L = 1,NC)
       REWIND 4
       DO 8 K = 1,NC
       F = D(K,M)
       IF (F) 4,15,4
   15  DO  2 J = 1,NC
       READ (NF(M)) (C(L) , L = 1,NC)
       GO TO 19
    4  F = 1.D0 / F
       DO 2 J = 1,NC
       IF (K-J) 16,17,16
   17  DO 21 L = 1,NC
   21  C(L) = 0.D0
       C(K) = 1.D0
       READ (NF(M))
       GO TO 22
   16  CONTINUE
       READ (NF(M)) (C(L), L = 1,NC)
       IF (N5) 23,22,22
   23  WRITE(3,11) J,(C(I), I = 1,NC)
   22  CONTINUE
       C(K) = C(K) * F
       G = C(K)
       DO 1 I = 1,NC
       IF(I-K) 3,1,3
    3  C(I) = C(I) - D(I,M) * G
    1  CONTINUE
   19  IF (J-K-1) 20,5,20
    5  DO 7 L = 1,NC
    7  D(L,N) = C(L)
   20  CONTINUE
```

```
      IF(N5) 9,10,10                                                    0380
    9 WRITE (3,11) J, (C(I), I= 1,NC)                                   0381
   11 FORMAT (I5/ (1X,1P10D12.5))                                       0382
   10 CONTINUE                                                          0383
      WRITE (NF(N)) (C(L), L = 1,NC)                                    0384
    2 CONTINUE                                                          0385
      REWIND 4                                                          0386
      REWIND 5                                                          0387
      L = M                                                             0388
      M = N                                                             0389
    8 N = L                                                             0390
      DO 30 J = 1,NC                                                    0391
      J4 = J * 4                                                        0392
      READ ( NF(M)) (C(L), L = 1,NC)                                    0393
      WRITE (N6' J4-3)(C(L), L =     1,100)                             0394
      WRITE (N6' J4-2)(C(L), L =   101,200)                             0395
      WRITE (N6' J4-1)(C(L), L =   201,300)                             0396
   30 CONTINUE                                                          0397
      CALL LINK (05)                                                    0398
      END                                                               0399
/*  TEB                                                               0400
   INCLUDE OVERLAY                                                      0401
// EXEC LNKEDT                                                         0402
```

TABLE 4 LISTING OF PROGRAM USED TO GENERATE TEST MATRIX

```
// OPTION CATAL
   PHASE BJPL$10,S
// EXEC FORTRAN
      COMMON N3, N4, N5, N6,JIM,NM,IT,IS,NJ,NP,NC,NX
C    N3  IS A WORK FILE ON DISC
C    N4 IS A TEMPORARY SEQUENTIAL FILE
C    N5  IS A SEQUENTIAL FILE FOR STORING MEMBER INFORMATION
C    N6 IS A DISC FILE IN WHICH MATRIX IS STORED
C    NM  IS THE NUMBER OF MEMBERS
C    IT  IS FILE TO WHICH PROGRAM WILL RETURN
C    IS   WILL CAUSE DIAGNOSTICS TO BE PRINTED
C    NP  IS PAGE NO
C    NC  IS NO OF COLUMNS IN BIG MATRIX
C    NX  IS NO OF COLUMNS IN COMPRESSED MATRIX
      DEFINE FILE  12(1600,200,U,I2)
      DOUBLE PRECISION  A(100,4)
      DO 10 K = 1,400
   10 A(K,1) = 0.00
      N3 = 13
      N6 = 12
      READ (1,1000) N5,IS,NX,(A(K,1) , K = 1,16)
 1000 FORMAT(2I2, I4, 16A4)
      WRITE (3,1000) N5, IS, NX, (A(K,1), K = 1,16)
      L4 = NX / 100 & 1
      DO 1 K = 1,NX
      DO 2 L = 1,NX
      LK = K * NX - NX & L
      KL = L * NX - NX & K
      A(L,1) = LK & 1
      IF (L-K) 3,6,2
    6 A(L,1) = A(L,1) * 10.
      GO TO 2
    3 A(L,1) = KL + 1
    2 CONTINUE
      K4 = 4*K - 4
      DO 4 L = 1,L4
    4 WRITE (N6' K4&L) (A(I,L) , I = 1,100)
      IF (N5) 5,5,1
    5 WRITE (3,1001) K,( A(I,1) , I = 1,NX)
    1 CONTINUE
      NM = MCTIME (0)
      WRITE (3,1001) NM
      CALL LINK (IS)
 1001 FORMAT(/I16/( 2X,10F10.1))
      END
/*  TEB
// EXEC ASSEMBLY
MCTIME  START   0
        USING   *,15
        GETIME BINARY
        LR      0,1
        BR      14
        END
/*  TEB
```

TABLE 5 LISTING OF PROGRAM USED TO CALCULATE RMS ERROR WHEN INVERTING A TEST MATRIX

```
// OPTION CATAL
// PHASE BJPL$05,S
// EXEC FORTRAN
      COMMON N3, N4, N5, N6,JIM,NM,IT,IS,NJ,NP,NC,NX
C    N3  IS A WORK FILE ON DISC
C    N4 IS A TEMPORARY SEQUENTIAL FILE
C    N5  IS A SEQUENTIAL FILE FOR STORING MEMBER INFORMATION
C    N6 IS A DISC FILE IN WHICH MATRIX IS STORED
C    NM  IS THE NUMBER OF MEMBERS
C    IT  IS FILE TO WHICH PROGRAM WILL RETURN
C    IS   WILL CAUSE DIAGNOSTICS TO BE PRINTED
C    NP  IS PAGE NO
C    NC  IS NO OF COLUMNS IN BIG MATRIX
C    NX  IS NO OF COLUMNS IN COMPRESSED MATRIX
      DEFINE FILE  12(1600,200,U,I2)
      DOUBLE PRECISION  A(100,4)
      NM = MCTIME(0) - NM
      NM =(NM + 30) / 60
      WRITE (3,1033) NM , NX
 1033 FORMAT ( I7, 25H MINUTES TO INVERT SIZE    , I5 / )
      SUM = 0.
      L4 = NX/100 + 1
      DO 1 K = 1,NX
      K4 = K*4 - 4
      DO 5 L = 1,L4
    5 READ( N6'K4 + L)( A(I,L), I = 1,NX)
      IF (N5) 15,15,14
   15 WRITE (3,103) K, (A(I,1), I = 1,NX)
  103 FORMAT (/I10/  (2X,1P10D12.4))
   14 CONTINUE
      DO 1 I = 1,NX
      E = 0.
      DO 2 J = 1,NX
      OIJ = NX*J - NX + I + 1
      IF (I-J) 4,6,2
    6  OIJ = OIJ * 10.
       GO TO 2
    4 OIJ = NX*I - NX + J +1
    2 E = E + A(J,1) * OIJ
      IF(I-K) 1,3,1
    3 E = E- 1.
    1 SUM = SUM + E * E
      E = SQRT (SUM /NX /NX ) * 100.
      WRITE (3,100) E
      CALL LINK (10)
  100 FORMAT ( 18H RMS ERROR, PC =      , F8.3)
      END
/*    TEB
// EXEC ASSEMBLY
MCTIME  START   0
        USING   *,15
        GETIME BINARY
        LR      0,1
```

```
      BR        14                                                    0521
      END                                                             0522
 TEB                                                                  0523
INCLUDE OVERLAY                                                       0524
EXEC LNKEDT                                                           0525
```

TABLE 6 LISTING OF THE SHORTEST PROGRAM THAT I COULD WRITE TO INVERT A MATRIX STORED ON DISC .

```
// JOB BJPL$
// OPTION CATAL
   PHASE BJPL$14,S
// EXEC FORTRAN
C    THIS IS SHORTEST PROGRAM THAT I COULD WRITE TO
C    INVERT MATRIX STORED ON DISC
       COMMON N3, N4, N5, N6,JIM,NM,IT,IS,NJ,NP,NC,NX
       DEFINE FILE 12(1600, 200, U , I2 )
       DOUBLE PRECISION T(100,3), D(100,3), Z(100,3), F,G
       L4 = (NX-1) / 100 + 1
       DO 1 L = 1,300
     1 Z(L,1) = 0.DO
       DO 21 K = 1,NX
       K4 = K*4 -4
       DO 2 M = 1,L4
     2 READ(N6' K4 + M) (D(L,M) , L = 1,100)
       F = 1.DO / D(K,1)
       Z(K,1) = 1.DO
       DO 3 M = 1,L4
     3 WRITE(N6' K4 + M) ( Z(L,M) , L= 1,100)
       Z(K,1) = 0.DO
       DO 21 J = 1,NX
       J4 = J*4 - 4
       DO 4 M = 1,L4
     4 READ (N6' J4 + M) (T(L,M), L = 1,100)
       T(K,1) = T(K,1) * F
       G = T(K,1)
       DO 22 I = 1,NX
       IF (I-K) 23,22,23
    23 T(I,1) = T(I,1) - D(I,1) * G
    22 CONTINUE
       DO 5 M = 1,L4
     5 WRITE( N6'J4 + M) (T(L,M), L = 1,100)
    21 CONTINUE
       CALL LINK (05)
       END
/*  TEB
   INCLUDE OVERLAY
// EXEC LNKEDT
/*          TEB

            TYPICAL DATA FOR RUNNING PROGRAM

// EXEC  BJPL$10
 1 4   60 X 60 MATRIX INVERSION BY PARTITIONING
 113   25 X 25 MATRIX INVERSION BY BY SEQUENTIAL FILE
 113   60 X 60 MATRIX INVERSION BY BY SEQUENTIAL FILE
/*
/&
```

COMPUTER REGISTRATION AT CHICO STATE COLLEGE

by

Neil C. McIntyre, Jr.

Faculty, Computer Sciences
Chico State College

COMPUTER REGISTRATION AT CHICO STATE COLLEGE

Since the inception of computerized registration at Chico State College in 1965, the College has received numerous inquiries from other colleges as to how our system works and how it might be modified and implemented for other schools and other hardware configurations. Since no general computer-oriented overview of our system has been undertaken previously, this paper is an attempt to describe the salient features of what we at Chico State College believe to be the only successful large-scale genuinely computerized registration system.

Like most California colleges, Chico State has recently experienced enormous growth. From an enrollment of 2,200 full-time equivalent students in 1961, the College has grown to 6,738 FTE this fall. Such rapid growth (about 15% for several years), with no abatement in sight for the near future, demanded that the College implement some form of automated registration.

Perhaps the best overview of the impact of computer registration at Chico State can be gleaned from a description of the process as executed under essentially a manual system and a description of the corresponding automated systems.

Prior to 1965, scheduling of students into the many sections of each course was effected by what we refer to as "arena" registration: on the Thursday, Friday, and Saturday preceding the first day of classes, the college's gymnasium

was equipped for mass student registration. For some time, record-keeping functions were assisted by unit-record data processing equipment. To register in the College, a student, having previously applied to and been admitted to the College or continuing from the previous semester in good standing, would be given a packet of punched cards which included a clearance card, study list, housing card, and the like. At the hour appointed for his class level and the initial letter of his last name, the student would proceed to the gymnasium. There he would wait, seemingly interminably, to be admitted to the arena. Inside, two major functions were performed--first, a student would select the classes he desired, proceed to the proper table, and, if the class he desired were not full, would be given an IBM card representing the class. If the student were exceptionally lucky, all his desired class sections would be open. If he were extremely unlucky, a class would close during the class-card gathering process and he would be forced to return his unwanted class cards, re-work his schedule with the help of an academic adviser, and start the class-card gathering process again. When he had all the desired class cards, he would proceed to the second phase: paying of fees. While the lines to pay fees were shorter and the time less than the waits to be admitted to the gymnasium and to receive class cards, the student was understandably grouchy after spending what was ordinarily about five hours in the registration process.

It can readily be seen that the process of registration by the arena method for 6,700 FTE will require more time and facilities than for 2,200 FTE: something had to be done, for our time and facilities were simply running out.

During the arena years, the class schedule was prepared entirely by hand, typed, photo-reduced, and printed. The secretary to the Vice-President for Academic Affairs would invariably spend the entire Christmas vacation period and the same amount of time during the summer manually preparing a list of all rooms and the classes to be taught therein during the upcoming semester. Inevitably, conflicts, empty rooms, and classes without rooms would appear which would frequently not be resolved until after classes began. As the College grew, physical plant utilization therefore dropped and confusion was the order of the day.

Students would meet with an academic adviser in their major subject area and, together, the student and his adviser would prepare a program planning sheet for the student's use at arena registration. By 1965, this process could not be completed conveniently on the Monday, Tuesday, and Wednesday preceding registration allotted to the advising procedure.

Moreover, all of the necessary but often irritating paraphenalia of student records, such as class lists, packet cards, grade lists, grade reports, and the like could not be conveniently prepared in the allotted time.

Chico State's computerized system revolves around three distinct but interdependent systems: schedule preparation,

student scheduling, and student record processing.

One of the major effects of computerized registration has been the expanded time schedule for the entire process. Using the arena system, the major functions were performed from just before Christmas until mid-February for the spring semester, and from early August to late September for the fall semester. Computerized registration has so drastically altered the scheme of things that registration processes are underway at all times during the year.

Chico State's registration process now begins with the computer center preparing, based on history and enrollment projections, a schedule request worksheet which is sent to the various academic department heads for their use in planning their course offerings for the upcoming semester. The schools and divisions of the College receive these worksheets shortly after the beginning of the preceding semester. Onto the worksheets the department heads write the requisite variable information, such as instructor number, class meeting times, building and room codes, and so forth. The completed worksheets are returned to the computer center for keypunching and verifying. The data are then submitted to the first in a series of major programs. This first program checks the schedule request against master lists of courses, rooms, instructors, valid times, etc., and produces listings of errors, omissions, and inconsistencies. The error listings are returned, together with a list of classes vying for the same room at the same

hour, to the department heads for thier corrections. The entire class schedule is stored on a disk pack once, and, as corrections, additions, and deletions are made to the schedule, additional error checks are performed. During the first week of November for the upcoming spring semester and during the third week in April for the fall semester, a computer program prints a copy of the tenative class schedule containing class number, department name, department course number, course title, number of units of credit, instructor names, and times. Building and room are not printed in the tenative schedule. The schedule is then sent to a print shop for photo-reduction and insertion in every copy of the next issue of the Wildcat, the free student newspaper.

The week following publication in the Wildcat is designated as "pre-registration" week. Each student meets with his adviser and prepares the successor to the program planning sheet. On the request form the student lists each class he desires, possbile alternates to it, the number of minimum and maximum units he will accept, states whether or not he will take any section of the course in order to get it, states any time restrictions he has, and similar information. The program planning sheets are then sent to the computer center for keypunching and verifying.

From the student request cards, a tabulation of the number of students requesting each course is prepared and

sent to academic department heads and college administrators. Based on demands for each course, departments may revise the schedule as required. During all phases of the schedule preparation, updated room lists, conflict lists, and lists of available space are frequently prepared, thus providing a great degree of flexibility and control over physical plant utilization. As an example, virtually no room conflicts now exist the day classes begin, whereas many of the classes had no room or met in a room occupied by another class before computer registration began.

When it is determined that the schedule is final (usually about four weeks before the beginning of the semester), computer registration is actually run.

Most registration programs for relatively large-scale use are "computer-assisted." Emphasis is principally on relieving facility and personnel problems of scheduling rather than on giving the student more choice and greater convenience in class selection. In this respect, our registration program is, I believe, unique, for it permits the student great latitude in course selection, time limitations, in fact, all of the advantages of ideal class scheduling with few disadvantages.

In the spring of 1965, two Chico State College students were commissioned to determine if a computer program could be written for the 1620 to enable registration by computer. The students determined that a program could be written by writing one! Written for, and still operating on (in slightly

modified form) a 20K card/disk 1620, the Chico State College registration program, as we call it, should actually be called a "scheduling" program. The registration program takes into account both student requests and the state of the schedule at the moment a student is processed. The process of scheduling is markedly similar to arena registration, but the computer does all of the work frustrating to students: in accordance with the student's requests, the program tries as many as 15,000 possible schedules before determining that it cannot find a workable schedule within the student- and college-applied restrictions. Where the student may spend upwards of three hours to find any workable schedule in the arena, the computer registers between 200 and 2000 students per hour on the 1620, depending upon the state of the schedule and how unrestrictive the student's requests are. When the program finds a schedule for a student which satisfies all requirements, a record of each class in which he is scheduled is punched, the disk records identifying the classes are updated to reflect the new enrollment, and the next student is processed. Three major characteristics of this program set it apart from most others seeking to provide the same sort of service: (1) the program allows great latitude and flexibility in any student's schedule--a student may apply sectioning, time, units, and course restrictions to prevent the program from giving him an unacceptable schedule; (2) when the program, as happens very frequently, is called upon to select a section from

those available, it attempts to schedule on the basis of class size, from least full to most full, thus balancing the size of the sections on a continuous basis. Only in the event that no section of a class will fit will the program attempt to delete the class and substitute an alternate if the student has listed one; and (3) the entire state of the schedule is held static during his processing, thus attempts at alternate schedules are unaffected by other students scheduling. The only condition causing a student to be unscheduled occurs when the program cannot satisfy the student's listed minimum units.

Irrespective of whether or not a student is successfully scheduled by the program, any errors the student and his adviser commit are logged and the program attempts to take corrective action. Among the many niceties in the program is the ability to include courses which have interdependent parts. For example, a Biology class may be divided into two hours of lecture, two hours of laboratory, and one hour of discussion each week. The student must register in laboratory and discussion sections which have the same instructor as the lecture section. The program insures that conditions of multiple-dependency such as the example are properly scheduled.

When all the students have been processed, the output cards are sorted alphabetically on student name and run through a program which prints, in class-schedule style, a list of the classes in which the student is scheduled on

continuous-form IBM cards. The separated cards are inserted in the proper student's registration packet together with his other registration materials. For the spring semester, the packets are then made available at the College Library for a period of several days. For the fall semester, the packets are mailed directly to the students.

Lists are prepared showing how many students were scheduled into each section, how many seats remain, and the percentage of students entering the process who were successfully registered.

The student now has the option of either accepting his entire schedule or rejecting it and participating in arena registration. It has been proposed and will soon be adopted that each student prepay materials and service fees and be required to accept the computer-generated schedule. At present, those who could not be registered by computer are required to register at a conventional arena. An arena-like arrangement is provided for spring semester for students to pay fees. Clearance cards from this fee-paying arena or from students who, during the summer, return their fees by mail, are matched against output from the computer registration run, class cards for the students who accepted their computer schedules are punched, and the remaining class cards for each course are punched, interpreted, and arranged for arena registration.

Arena registration was held for one day only this fall, 1967, for the first time in many years. Due to computer

assistance in room utilization, class card preparation, and class status reports, more than 4,000 students were processed in the arena from 8 AM to 8 PM this fall. Of those, nearly 2,500 were entering freshmen and transfer students who cannot currently participate in computer registration.

At the end of arena registration, class cards from both computer and arena registration are merged and class rolls are prepared. Updated rolls reflecting added and dropped students are prepared at regular intervals during the academic semesters. Chico State also has, in the form of the student record-keeping system mentioned previously, the usual computerized grade cards, grade rolls, permanent record labels, lists of the standing of each student, and many additional reports and services.

Due to the size limitations of the 1620 and the probability of a new, larger computer on the campus during the next few years, we have set several design maximums on all the systems. We have provided for a maximum of 100 sections of any one course; 2,400 different course types; 3,000 sections of classes; 750 instructors; 240 classrooms; 9,000 students entering computer registration; and 47,000 class cards. While the design maximums can be very much larger within the capability of our 1620, we do not feel that the allocation of the additional space on the system disk pack is desirable. For example, we presently sort, store, modify, and otherwise process virtually all of the

data within the system on one disk pack which also stores all of the programs to use that data. To expand the size maximums of the system would require us to revert to card resident programs and/or additional disk packs.

A new, more capable computer system, such as a System/360, will permit us to add additional flexibility to a request set for any given student. In addition to the obvious advantages of increased speed for all processing, a larger system would eliminate over 500 hours of sorting class and packet cards each semester. We would be able to generate an optimized class schedule based on history, enrollment projections, physical plant utilization, personnel idiosyncrasies, unique course requirements, and student desires, thus relieving the administration and department heads from much of the routine work now necessary for schedule preparation.

Among the many features we will implement with a newer system will be student request by course only, with programs to select and schedule the corresponding laboratory and activity courses required.

We have so often been asked about modifications and implementation for other machines and schools that I feel a few remarks might be useful.

The system we have designed is restricted to a college employing a semester-to-semester schedule and student planning basis. Modification for the 1620 for a college or other school desiring to plan college and student schedules more

than one semester in advance seems unlikely, for we have stretched the capability of the 1620 nearly to its breaking point. I am certain that long-range scheduling is possible and desirable for any school having, say, a large-scale 1401 or a 360 system having a disk. I have carefully examined other IBM and some non-IBM systems for simplicity of adaptation and implementation of our system and I can say certainly that the system as we use it could very easily be processed on a single-disk 12K 360 model 20. It should be obvious that a system with capabilities as great as the model 30 lends itself well to expansion of our system.

While the restrictions on size we have imposed upon ourselves may seem important, it is highly unlikely that any college much larger than Chico State would want to implement the system if the largest machine available to the school were a 1620. We have been asked if the system can be implemented on an 1130 disk system. It can indeed be implemented successfully, albeit on a smaller scale, due to the size limitations of the 1130 disk.

Chico State's system is based on a six-day week, 360 15-minute periods-per-week system of time. This arrangement is not fundamental to the system, however, and could be changed without difficulty: while our system is designed for a six-day week, Chico State is, at present, largely a five-day-a-week school.

By far the most frequent question we are asked is if any part of our system is dependent upon the characteristics

of the 1620. Let me assure you that, even at the time the system was designed, we considered the 1620 at best an interim machine and, as such, we have always insisted to all who have worked on the project that it remain free of machine dependencies. While re-programming would, of course, be necessary for implementation on other hardware, conversion or modification for a new computer will not be at all difficult.

Significant in the development of the entire system are two facts: (1) many ideas were conceived and all programs were written entirely by students at the College; and (2) when the idea for the registration program was born, IBM Systems Engineers were asked if it could be done on our 1620. Their reply: not even on a 60K, four-disk 1620 model II. Chico State College is indeed fortunate to have students rebellious enough to doubt all that they are told is and is not possible.

STUDENT INFORMATION SYSTEM

AT

CHRISTIAN BROTHERS COLLEGE

Presented at the

Winter Meeting of COMMON

December 8, 1967

by

Brother Jerome David Wegener
Registrar and Dean of Admissions
Christian Brothers College
Memphis, Tennessee

STUDENT INFORMATION SYSTEM AT CHRISTIAN BROTHERS COLLEGE USING AN IBM 1130 COMPUTER

In 1963, a system of records and grade reports adapted to the 1620 was initiated. This was a minimum configuration consisting of a 40K memory, a 407 for print-out, an 082 sorter, and several key punches used also for interpreting and reproducing. This was basically a unit record operation with the computer used for calculations. There was a great deal of hand work involved in every operation.

This summer, an 1130 with disk, 8K word memory and an 1132 printer was installed. The immediate problem, then, was to reprogram the system making it adaptable to the 1130. This new machine had many advantages with its on-line printer and its disk storage for direct access to student information. It should be noted that IBM has prepared a Student Information System for the 1130 available from the COMMON library (3.0.003). For several reasons, it was decided not to use this. One important reason was that there was no provision made for storing the student's address. Since there is a great deal of correspondence between school and student, it was essential that this be included. There were a number of other variables in the old system that had not been included in the IBM Student Information System. Because of the shortness of time, it was desirable to keep as many of the features of the former method as possible. Another important reason for not adopting it was the fact that it

had been written in Assembly language. While it is probable that it could have been adapted, it was felt that there would be many problems and difficulties involved in trying to revise a large assembly program. Therefore, it was decided to start from scratch and to write a system in Fortran incorporating all the data forms of the previous system. This probably runs a bit slower, but the ease in reading, debugging, and revising seems to make it well worth the time.

The first problem was to lay out the student file. It had been more or less arbitrarily decided to attempt to store two students per sector. However, most attempts at planning it ended with more than 160 words required. This difficulty was solved by adapting several of the subroutines from the Commercial Subrouting Package (1130-SE-25X). These are similar to the Forcom routine for the 1620. One basic advantage that these new ones have is that, with two or three exceptions, they are written in FORTRAN so that they can be modified with a minimum of effort. They are called in the same manner as are FORTRAN subroutines. Of primary interest is the Pack/Unpack subroutine. This takes an array in A1 format (one character per word) and converts it to A2 format (two characters per word), thus halving the amount of storage required. A GET subroutine converts from the A-format to decimal form for arithmetic operations, and a PUT subroutine accomplishes the reverse conversion.

The student file finally decided upon was set to contain 320 characters. When operating on the file, these characters would be in A1 format, and

then packed to 160 A2 format characters for the purpose of storage. Thus two students occupy one sector on the disk. As can be seen in Figure 1, this was more than ample for our needs, and it had space for additional information that might need to be added as time went on.

The system works in the following manner. As was mentioned above, as much as possible of the old system was kept intact. From the application blank, the student's name is punched and he is assigned a student number. The student number is a five-digit number assigned in alphabetical order. In originally setting up the student file, they were read into the computer without regard to order. The computer then sorted them and so assigned a file number to each student. At the same time, it stores the student numbers in a separate file in the same order. Whenever a particular file is needed, the student number is read in. The computer then does a binary search using function subroutine RSEEK to locate that particular student's file. His file is read into core, unpacked, and the necessary information postioned, usually by means of an EQUIVALENCE or a PUT statement. The file is then packed and written back onto the disk.

At the time of registration, the students pick up their own class cards after their schedule has been approved by their adviser. Positioning these behind the student's name card, they are ready to be read directly into the computer. The computer reads in the class cards for each student, alphabetizes them, and stores them in the student's file in the manner described.

This has eliminated the long process of gang-punching all of the class cards, and then sorting them into order first by course, and then by student. The next program reads all of the classes from all of the files, and sorts the students into order according to their classes. Class lists are then ready to be printed and after that, the student information sheets. The latter are distributed in toto to the Dean, Guidance Office and other administrative offices, to the dormitory prefects according to residence hall, and to the department heads according to the students' majors.

Adds and drops are handled very easily. The very tedious job of pulling drops and inserting adds by hand has been eliminated. It is now necessary to simply place the add cards after the student's name card, followed by the drop cards (coded with a 1 in column sever) and read them into the computer which then rearranges the student's file. The running of report cards has also been simplified. It is no longer necessary to gang punch the marks into each of the grade cards. The cards are simply sorted by grade and then read into the computer, switch settings indicating the type of mark that should be stored. Honor rolls, failure lists and grade statistics are produced very easily, also. Permanent record labels are printed at the semester.

New students can be added very easily once a student number has been assigned. The program finds the proper position and moves down one

notch all those files that will be behind him, much as a Disk Utility Program does. Likewise, a student's file can be deleted with those following moved up one. Bills for each student are run off for the Business Office. The Student Directory, Teachers' Schedule and various types of statistics have been programmed also. Since the memory is quite small, each of these different operations requires a separate program, and in some cases, two or three linked together. This is, however, no great difficulty, since once they are compiled and stored on the disk, a simple XEQ command calls them into action.

Improvements will be made when the two additional disks that are on order arrive. At that time, there will be more than enough storage to keep the student's entire four year course of studies accessible by the computer. The 1132 printer is quite slow, but the operations will be greatly speeded up when it is replaced by the 1403 printer.

As presently set up, the system will handle 1200 students, 100 instructors and 300 courses. Figure 2 shows the layout of the cards used as input. While no claims are made about the completeness or efficiency of the programs, the system does work well. With proper modification, it could probably do the job at other small colleges as well.

1 A	33 L	65 R	97 A	129 8	161 1	193	225 9	257 4	289
2 D	34 P	66 I	98 N	130 1	162 B	194 2	226 8	258 1	290
3 O	35 H	67 N	99	131 0	163 0	195 3	227 F	259 0	291
4 L	36	68 G	100 H	132 2	164 2	196 1	228	260 1	292
5 P	37	69 F	101 A	133 3	165 0	197 E	229	261 7	293
6 H	38	70 I	102 L	134 4	166 8	198	230	262 0	294
7	39	71 E	103 L	135 9	167	199 4	231	263 1	295
8 W	40	72 L	104	136 1	168	200 2	232	264 1	296
9 I	41	73 D	105	137 9	169	201 3	233	265 0	297
10 L	42	74	106 2	138 1	170	202 D	234	266 1	298
11 L	43 5	75 P	107 7	139 2	171	203	235	267 6	299
12 I	44 2	76 A	108 2	140 0	172	204 4	236	268	300
13 A	45 4	77	109 -	141 6	173	205 5	237	269	301
14 M	46	78	110 1	142 0	174	206 6	238	270	302
15	47 C	79	111 5	143 8	175	207 E	239	271	303
16 F	48 H	80	112 1	144 1	176	208	240	272	304
17	49 E	81 1	113 6	145 1	177	209 6	241	273	305
18	50 Y	82 9	114	146 1	178	210 5	242	274	306
19	51 N	83 0	115 1	147 1	179	211 1	243	275	307
20	52 E	84 6	116 7	148 1	180	212 D	244	276	308
21	53 Y	85 4	117 4	149 0	181	213	245	277	309
22 M	54	86 3	118 8	150 0	182	214 7	246	278	310
23 R	55 R	87 6	119 5	151 8	183	215 0	247	279	311
24	56 D	88 4	120	152 0	184	216 2	248	280	312
25 W	57	89	121	153 0	185	217 C	249	281	313
26 M	58	90 M	122	154 1	186	218	250	282	314
27	59	91 A	123	155 0	187	219 7	251	283	315
28 F	60	92 U	124	156 0	188	220 6	252	284	316
29	61	93 R	125	157 2	189 0	221 8	253	285	317
30 A	62	94 E	126 1	158 0	190 3	222 F	254	286	318
31 D	63 S	95 L	127 7	159 3	191 6	223	255 0	287	319
32 O	64 P	96 I	128 8	160 8	192 C	224 7	256 9	288	320

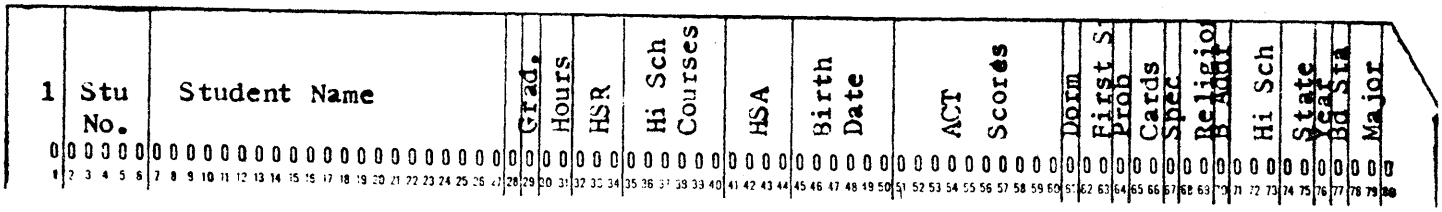
Figure I-a

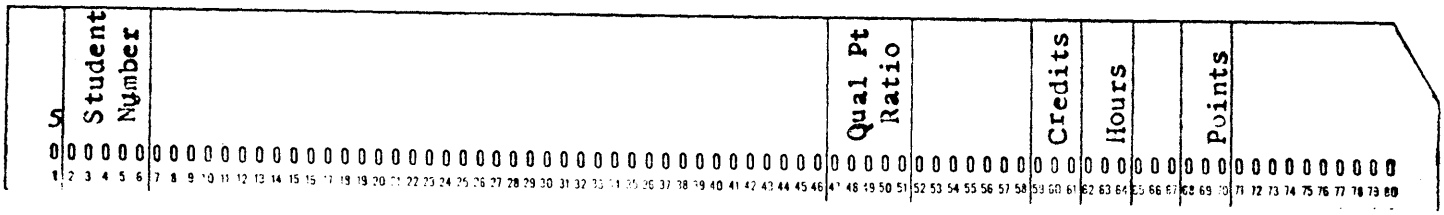
DISK STUDENT RECORD FORMAT

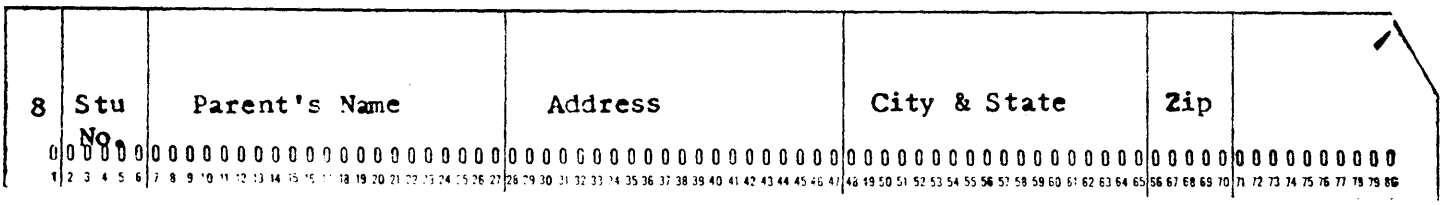
Word	Description
1-21	Student Name
22-42	Parents' Name
43-62	Address
63-80	City and State
81-85	Zip Code
86-105	Local address
106-113	Local telephone number
114	Graduation code
115-116	Hours being taken
117-119	High School rank
120-121	High school courses
122-129	High school average
130-135	Date of birth
136-145	ACT scores
146	Dormitory code
147-148	First time status
149	Probation code
150-151	Number of courses being taken
152	Special student code
153-154	Religious preference code
155	Billing address code
156-158	High school code
159-160	State code
161	Year in school
162	Boarding status
163-164	Major code
165-166	Action code
167-175	Blank
176-179	Cumulative quality point ratio
180-182	Cumulative hours
183-185	Cumulative credits
186-188	Cumulative quality points
189-248	Course numbers and grades In groups of five: 1-3 course number 4 grade in course 5 former grade if repeating
249-251	Rank in class
252-254	Number in class
255-258	Semester quality point ratio
259-261	Semester hours
262-264	Semester credits
265-267	Semester quality points

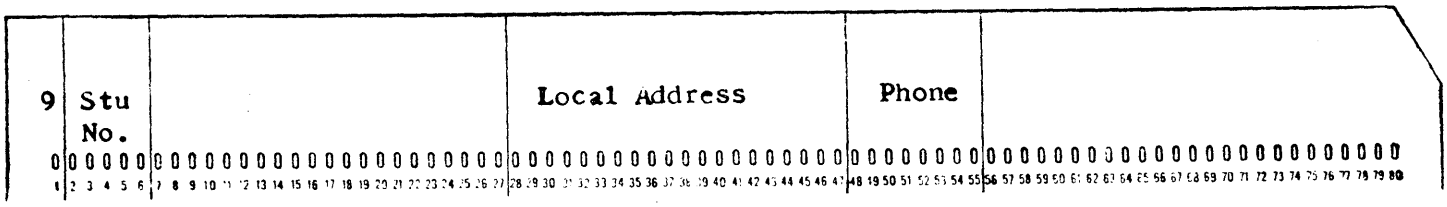
Figure 1-b

CARD INPUT FORMATS









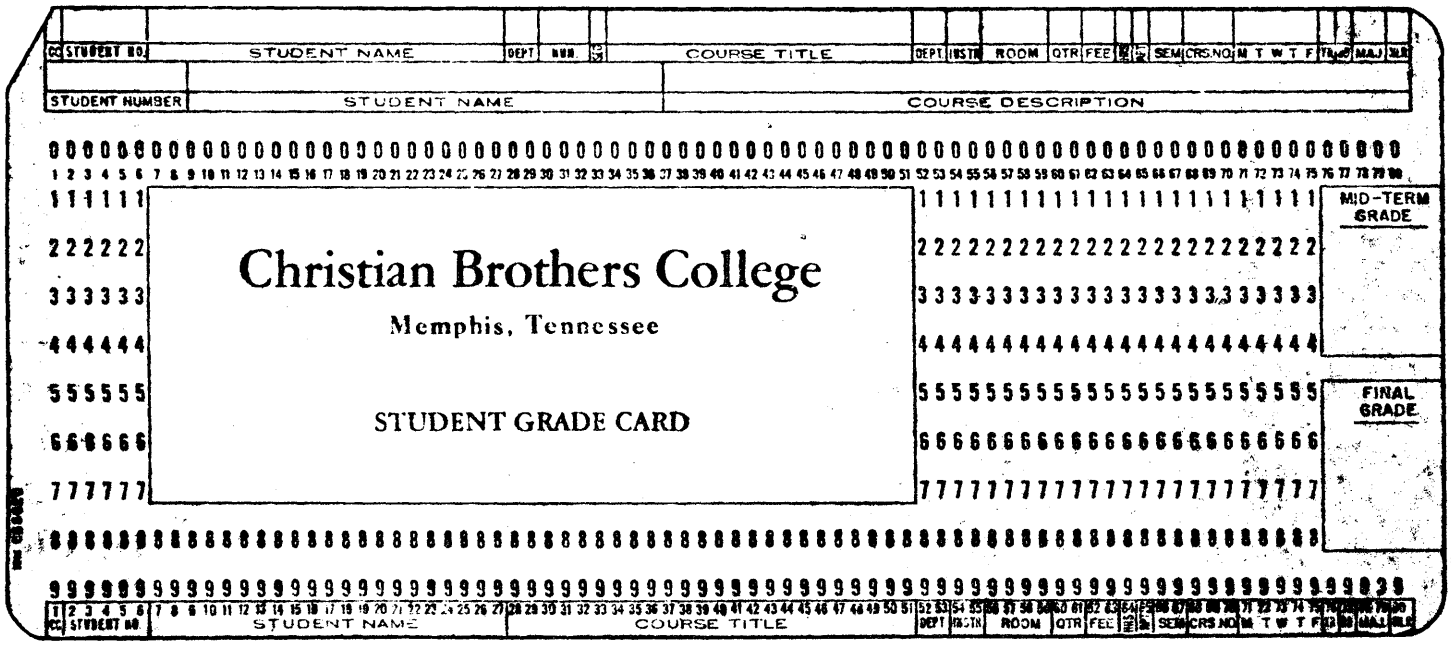


Figure 2

Appendix A. Programs used to set up and maintain student file

RDNAM	Read in name cards to initialize students
RDATA	Read in students' data
RDSKD	Read in students' schedules, any order
CTSEC	Count setions and prepare class lists
PRLST	Print class lists
INFO	Print student information
ADDCS	Adds and Drops
COREC	To correct student information
ADNAM	To put a new student in the file
BILLS	To prepare bills
BILSM	Billing summary
PRTOT	To print totals in each section
ENROL	Enrollment statistics
DRTRY	Print student directory

Appendix B. Programs used at report time.

PCHGR	To punch grade cards and print grade report forms
CLGRS	To clear grade area and set Orientation mark
RDGRS	To read in grades
CKGRS	Check to see if all grades are present
CORGR	To correct missing grades
GRSUM	To prepare grade totals
GRSM2	To prepare grade totals (second half)
PRGSM	To print grade totals and statistics
CTACN	To separate and store actions
PRDL	To print Dean's list
FALUR	To prepare failure list
CLSTD	To determine class standing

Appendix C. Utility programs

RDCRS	Read in master course cards
RDTCH	Read in teachers' names
CTTCH	Prepare teachers' schedule
PRTSK	To print teachers' schedule
RDABM	Read in abbreviated majors
RDACN	Read in actions
RDEPT	To read in department names
RDPAB	To read in department abbreviations (4 letters)
RDPAA	To read in department abbreviations abbreviated (2 letters)
RDFIL	To read in entire student file from four 80A1 cards
RDMKS	To read in and store the 12 possible marks
PCHCD	To punch class cards
PRCRS	Print list of all courses
PRFIL	Print out student files
PCHFL	To punch entire student file
STPRB	To set probation code to zero

Appendix D. Mark Conversion

The twelve possible marks are stored in a file in a specified order. In the student file they are given a code letter as follows:

Mark Number	Mark	Mark Code
1	A	A
2	B	B
3	C	C
4	D	D
5	F	E
6	S	F
7	U	G
8	I	H
9	WP	I
10	WF	J
11	Ex	K
12	Au	L

The mark number is read in. The following steps set up the mark code:

```
      IF(MKNUM - 10) 1,2,2
1     MCODE + -16320 + 256*MKNUM
      GO TO 3
2     MCODE = -14528 + 256*MKNUM
3     CONTINUE
```

The reverse process is used to determine the mark number from the mark code when printing reports.

Appendix E. SUBROUTINES

GET(RLVAR,JFLD,I,J,ADJST)	To get a real variable from an array.
IGET(INTVR,JFLD,I,J)	To get an integer variable from an array.
PUT(RLVAR,JFLD,I,J)	To store a real variable in an array.
IPUT(INTVR,JFLD,I,J)	To store an integer variable in an array.
RSEEK(RLVAR,RARAY,N)	To locate position of a real variable in an ordered real array.
ISEEK(INTVA,IARAY,N)	To locate the position of an integer variable in an ordered integer array.
SORT(RARAY,N,RMODE)	To order a real array.
ISORT(IARAY,N,MODE)	To order an integer array.
PACK(LIST,I,J,LPKD,K)	To convert A1 format to A2.
UNPAC(LPKD,I,J,LIST,K)	To convert A2 format to A1.

1.	Name of Prime Committee:	Installation Management Project
2.	Subject:	"Use of Psychological Tests In Selecting Programmers"
3.	Speaker's Name:	David B. Mayer, (speaker) co-author: A. W. Stalnaker
4.	Company Speaker Represents:	International Business Machines and Georgia Institute of Technology
5.	Mailing address and Phone No.:	IBM Thomas J. Watson Research Center P. O. Box 218 Yorktown Heights, New York 10598 (914) 945-1708
6.	Day and Time of Speech:	Tuesday, December 12, 1967 8:30 - 10:00 a.m.
7.	Number of Pages of Text and Number of Pages of Graphics Included:	21 Pages of Text 3 Pages of Graphics

The Use of Psychological Tests In Selecting Computer Programmers

David B. Mayer, IBM - Watson Research Center
Yorktown Heights, New Yotk

Ashford W. Stalnaker - Georgia Institute of Technology
Atlanta, Georgia

A presentation for the COMMON Organization, San Francisco, December 12, 1967. Adapted from the tutorial entitled "Computer Personnel Research - Issues and Progress in the 60's" given at the Fifth Annual Conference on Computer Personnel Research, University of Maryland, June 26, 1967, sponsored by ACM-SIG/CPR.

The first session of the Conference was devoted to a review of SIG/CPR's progress in research since 1962 and some thoughts on problems which still face us. This review was in the form of a dialogue between a computer center manager, David B. Mayer, and a management scientist, Professor Ashford W. Stalnaker.

Mayer: Good morning ladies and gentlemen.

I represent both IBM and the ACM Special Interest Group on Computer Personnel Research. Originally, this group was known as CPRG (Computer Personnel Research Group); founded back in 1962 as a result of some discussions at the RAND Corporation. I guess one might say that psychologist Robert Reinstedt was a little lonesome for the company of some other researchers. He had been investigating the problem of programmer selection and discovered that there was indeed very little research on this subject -- except in the hands of a few colleagues in his general professional area, namely, Dallis Perry at the Systems Development Corporation, Professors Raymond Berger and James Rigney at USC, Jim Tupac at RAND, and Dr. Sherwood Peres who was at the Sandia Corporation. He called these people together and they laid out a charter for CPRG in September 1962 at the American Psychological Association meeting. The group decided upon the final design of a test battery to be administered across the country to as many installations and experienced programmers as they possibly could find. The test battery consisted of the Programmer's Aptitude Test, (better known as the PAT) by Hughes and McNamara of IBM; the Strong Vocational Interest Blank; and a special trial test named the Test of Sequential Instructions. There was some personal background material covering education experience, and so forth. The results of this test battery will be included in the body of my talk, but at this point it suffices to note that we completed it successfully, and we calculated correlations on the data, obtained much information, and initiated the kind of research that CPRG performs.

The initial researchers went on from that point to develop a Programmer's Appraisal Instrument, which is an evaluation device. This was the result of Dr. Sid Fine's work with Robert Dickmann (now SIG/CPR Chairman) at Johns Hopkins University. It was tested there at the Applied Physics Laboratory, and at 24 other installations.

The next step in our research was aimed at advancing the state-of-the-art of interest tests. The Strong Vocational Interest Blank had been revised (this had nothing to do with CPRG) and, after additional testing, Dallis Perry developed a key for programming as a separate occupation.

In 1966 ACM approached us -- they thought that we were doing so well that we should join the computing fraternity, since originally CPRG was composed primarily of psychologists with only a sprinkling of computer managers. We agreed that joining forces with ACM allowed CPRG's research to be enhanced through broader contacts and outlets. Hence, in this survey paper today, I would like to detail some of that research history, and some of the existing issues as we see them at this time.

First, permit me to tell you a bit about the use of some of these tests in selecting computer personnel.

Figure 1 summarizes the results of the Dickmann survey of 1966 which was reported at the Fourth Annual Conference (1). This figure indicates that 483 firms in the United States and another 98 in Canada participated in the survey. In the US 68 percent used tests in some form or another for selection. This corresponds very closely to 72 percent in Canada. The number of programmer-analysts actually employed by these organizations was

over 23 thousand in the United States, with another thousand in Canada. The number of people who are needed in the forthcoming year as of the time of this survey was another 25 percent.

Figure 1 also shows the composition of the sample by industry groups.

The kinds of programming being performed were basically of four major types - BUSINESS, SCIENTIFIC, SOFTWARE or MILITARY, or combinations of these (Figure 2). Of them, the largest percentages were in business computations with scientific applications coming in somewhat lower. Military and software programming were small by comparison.

What kind of programmers are there? We classified them at the time the survey was designed into four major categories: a programmer who was essentially a junior or trainee; the second one was called the experienced programmer; a third level we called the system analyst trainee; and the fourth one was termed the experienced systems analyst. Figure 3 answers the question as to what kind of education is demanded by the various institutions or organizations in their hiring practices. In the United States it tended toward having some college training or a degree - over 50 percent of the US sample, especially for the programmer-trainee. This tendency is a little more emphasized as you move up the experience ladder. Canada's educational requirements were somewhat lower, possibly because they do not have as large a college population from which to recruit. In Canada, 65 percent of the programmer trainees had only a high school education and the remainder had some college or above. If you consider the experienced Canadian systems analysts, you will find that 28 percent had a high school education or better. However, 37 percent were not reported, so we can only generalize about the true proportions in this situation. Canada, therefore, is drawing on its resources of personnel in accordance with what they have available.

Tests were used in many of these organizations, but they were used differently depending on whether the firm required much education, or little. Or to put it in reverse, possibly tests were NOT used in many cases, and hence the educational requirement was increased to compensate. Taking one category as an example, the systems analyst trainee, almost 50 percent were required to be college graduates if tests were not used; if a test were used, only 39 percent were required to have college degrees. This pattern repeats itself throughout Figure 4. We will not dwell on this except to note that tests are used in many cases in conjunction with educational requirements, but in differing degrees.

What types of tests are used in these two countries? The tests reported used were broken down into four major classifications as shown in Figures 5 and 6. The first major classification is the general intelligence test with the most commonly used being the Wonderlic Personnel Test -- 60 organizations in the US and 7 in Canada. 13 of these organizations had undertaken validation studies of this test. Whether the validation studies consisted of actual on-the-job performance validation or training validation was not indicated in the survey. The other two principal tests used are general intelligence tests.

The second type is the aptitude test, which is being used as if it isolates programming as a separate and special aptitude. The IBM Programmer

Aptitude Test, PAT, is by far the most commonly used test in both countries: over 282 organizations in the United States -- approximately 83 percent, and 67 in Canada. Of the remaining aptitude tests, the National Cash Register Test, the Science Research Associates Test Battery and several others were used, but nowhere nearly as widely as the PAT. The Federal Service Entrance Examination is not a true aptitude test - it is a test, I believe, that is given to almost any prospective federal service employee, for many different positions.

The two other types found in the survey were the personality tests and the interest tests. Very few of them are used. Personality tests were being used by only 10 or 15 organizations and very sparingly at that. For example, the Thurstone Temperament Schedule and the Activity Vector Analysis were each used by only three organizations; several others were used in varying degrees.

For the interest tests, the Kuder Preference Record was cited by only two organizations. Interestingly enough to me, neither Strong Vocational Interest Blank (SVIB, original or revised version) was mentioned by any organization. Yet it is one of the few for which there exists a key for programming. It is hoped that SIG/CPR will have some educational effect by bringing the value of SVIB to the attention of those responsible for personnel selection.

Among the other tests is the 1401 Autocoder exam, which was developed by Computer Usage Corporation. This test is a form of the Logical Analysis Device developed by Langmuir at the Psychological Corporation of America. This latter test, known as the LAD, was not cited at all, however.

Let us take a look at the use of the PAT for a moment. (See Figure 7). 282 organizations use it in the United States. 128 of them use it in combination with some other test. 154 use it alone. As for position levels at which it is used, for the programmer trainee - 278 organizations gave the PAT; for the experienced programmer ---and this is always a delicate subject for a computer manager who is interviewing candidates--- there were 138 organizations; for systems analyst trainees - 142; and for experienced systems analysts - only 87. Of these, 71 organizations had performed validation studies - that is, how the performance of the programmer compared to his score on the PAT. 22 organizations, or a little less than 10 percent, actually discontinued the use of the PAT for various reasons.

This completes our summary of the Dickmann survey, and I think it would be well worth while to go into some of these tests and describe them and relate them to a sample test which you will take in a few minutes. Let us consider some of the several tests that have been used in various CPRG studies --- a sample of some of these were included in the small test battery which you just completed. In the original CPRG national survey (6), three cognitive devices were included. These were the PAT, the Strong Vocational Interest Blank, and the Test of Sequential Instructions. The first of these, as was indicated by the Dickmann survey, is by far the most popular selection instrument in both the US and Canada. The results of the first CPRG study indicated positive correlations between

the PAT score and actual performance only in a small number of cases. In the overall summary of the report, no correlation was found between the PAT and supervisory rankings of performance. The PAT has also been used in two studies subsequent to the CPRG national survey. The first was by Biamonte (11) at NYU working with a group of non-credit programming students, and the second by Gotterer and Stalnaker (12) at Georgia Tech with several groups of undergraduates enrolled in a computer course which had as a major component programming and systems analysis. In the latter case, as was true in the national survey, no correlation was found between PAT scores and performance in a training situation. We emphasize that the work at Georgia Tech was strictly in a training situation and in no way relates to subsequent performance in an actual programming assignment.

Does such a thing as a programming aptitude really exist? Well, I have been trying to find that out from CPRG for several years. You know, after over a thousand interviews -- which I'm told is the worst way to find out whether they are a programmer or have potential -- and approximately 200 people hired over my signature, I would say that I can detect what one might call an X factor; it's probably called aptitude. I do not know of what it consists; all I can say is that a particular candidate sitting before me seems to possess "it" and will succeed in the programming art.

CPRG seriously questions that there is, as far as the training situation is concerned, any indication of a specific programming aptitude. Our interest was generated by the repeated occasions which were observed at Georgia Institute of Technology, wherein a truly marginal student -- a student who was only barely able to maintain satisfactory status in school -- was able to succeed in developing a rather sophisticated programming skill and also a sophisticated approach to systems analysis.

Let us now look at some of the CPRG results with regard to the PAT (6). As I mentioned, only in a limited number of cases was there a significant correlation between PAT results and ranked performance. However, I feel that one of the points that we should pay special attention to is the sort of cross-overs or the reversals we get in terms of the PAT. We will notice among the business programmers (Figure 13) who are graded in the upper half with regard to their performance, 42 percent of them scored 44 or below the PAT. On the other hand, among those who were rated in the lower half in regard to their performances, 49 percent of these scored 69 or above on the PAT. We might note too that the relationship in this case could be curvilinear. In the scientific group, Figure 14, the relationship is approximately linear and the degree of the reversals not as large as in the case of the business group. Here we note that 31 percent of those who are rated in the upper half with regard to the performance scored 44 or below, and 34 percent who scored in the lower half in terms of their performance scored 69 or above. In the sample there were 534 programmers; 301 of them were scientific programmers and the remainder were in business programming. There were 25 different installations and companies.

(Administer and discuss the DTSI here)

A second component of the national study (6) was the Test of Sequential Instructions. The author of the TSI states that it was designed to

show that any test that in some way measures a form of logical reasoning will in some sense indicate the level of performance in programming jobs and will also indicate in some sense the intelligence of the individual. This hypothesis is borne out by the results of the national study in which the correlations between the TSI and the PAT were in many cases quite significant.

In my scientific programming group I obtained a correlation coefficient of .70 between the PAT test results and my supervisor's rankings. But interestingly enough, on the TSI, the correlation coefficient was .71. So I naturally asked myself, what was I doing that was right? Could I interchange the TSI with the PAT as part of selection procedure for programmers?

The DTSI, to me, tests the ability to do multiple tasks simultaneously. To perform well on that test you are holding one task in the back of your mind while another task is being performed in the foreground. Then, we build up to 3, 4 and 5, in fact, in the real TSI, I think we have 7 tasks running simultaneously. The PAT, to me, has no such attribute. The PAT does test other components which are required in programming --- numerical capability, spatial relationships, and such. I suspect the two tests supplementary rather than interchangeable.

The third component of the CPRG national the Strong Vocational Inventory Blank (SVIB). The SVIB is a test that has been in use for a great number of years, primarily though, in the vocational counseling area. The purpose of the SVIB is to elicit information regarding the interests of the testee. The interests are then compared to the interests of people who have been successful in several occupational groups, the hypothesis being that if a person has interests similar to those successful in certain occupations, there may be some motivation to enter this occupational area. It should be noted that the SVIB is not claimed by its author to predict performance on the job; instead, it only elicits information regarding interests.

In Figure 15 we compare the interests of computer personnel to the public in general in regard to certain answers to questions on the SVIB. The SVIB contains 400 items. Notice specifically the item on progressive people: 41 percent of computer personnel like this type of person -- among the general public 85 percent of these people like people with this outlook. On the other hand, there is a great flocking among computer personnel to conservative people. 84 percent prefer these, while among men in general, only 56 percent. Another example was thrifty people: 45 percent of computer personnel stated they liked this attribute. They are much better liked by people in general -- 74 percent.

Can we look at individual items in the SVIB to see if these indicate anything in regard to the general interest pattern that should be the pattern of a successful programmer. The answer to this is a decided NO. We mentioned two categories -- progressive and conservative people. The results shown by the SVIB are reversed in work by Biamonte (7) at NYU in which he specifically considered attitudes. He shows there were negative correlations between training success and such attributes as dogmatism, conservatism and authoritarianism, a finding which would indicate the reverse of your conjecture. The point is that the SVIB questions when taken out of context have no meaning. One must analyze the complete 400 questions in order to elicit anything regarding the total interest pattern of the individual.

The normal scoring of the SVIB is quite complex because it has to be scored for all occupations. It is possible, I might mention, to hand-score for a particular occupational group. For instance, since the interest of this group is the computer programmer, it could be that the keys could be obtained and thus we could hand-score for this specific occupation. However, I would like to warn against this; the SVIB is meaningful only in terms of its total content. When we take an occupation or a question out of context, the results are questionable to say the least.

I have mentioned the existence of the programmer scoring key for the revised SVIB. The series of questions that you have in your sample battery are from the old SVIB which was the one used in the original CPRG national study. Subsequently, Dallis Perry (8) undertook another national study in which he used the revised SVIB. He also worked with a larger sample than that included in the original CPRG study. Based upon this work, Perry has developed the programmer scoring key which is now available to all users of the SVIB.

I think now we should move onto the topic of programmer evaluation -- how can we determine if a programmer is actually effective on the job. The first research in this area was reported by CPRG. This was the Programmer appraisal Instrument (4) developed at the Applied Physics Laboratory, under the direction of Bob Dickmann and Sid Fine. This is a multi-dimensional instrument, which in many ways appears to be more concerned with what might be regarded as a professional programmer rather than the operating programmer - at least I think this is sort of a summary of remarks made in the evaluation of the PAI. There is a considerable emphasis on professional activities and this, in some cases, has led to resistance. It is composed of four specific areas: professional preparation and activity, programming competence, dealing with people and adapting to the job. This instrument was validated by Bob Dickmann but is not believed to be widely used at this time.

The PAI asks a number of items to be scored numerically, such as: how many societies does he belong to? and how old is he? does he give some on-the-job training -- a lot, or not at all? A supervisor finds that the items do not really cover the subject of <u>programming</u>, or programming <u>capability</u>. Supervisors generally shy away from question-and-answer procedures for appraisal of programmers. Practically none have actually put this PAI or equivalent ratings into effect. Progressive as I am - I haven't either. My project leaders were very resistant to it. They prefer to use subjective techniques, such as their impression of the programmer's output; sometimes they actually read his programs. But a formal document of this type they resist. In evaluating a coder for upgrading to programmer, a better procedure would be to temporarily take a coder out of his current category and put him into direct programming tasks and to observe his programming capability rather than evaluating him through a questionnaire. Additionally, I would like to have some kind of test which will actually show his level of competence. A test that is concerned with programming ability to indicate readiness for the move from coder or programmer to senior programmer should be implemented. There is one further thing that must be done, and that is, as Dr. Paul Herwitz (13) of IBM has recently stated, the only way a supervisor can tell what a

programmer is doing is by being knowledgeable about the code that he has written. In other words, he must read the program, and very few supervisors do. That would be the first step I would say towards proper evaluation.

The second step would be to give a proficiency test, an objective one. In terms of such an objective test, Berger (10) at USC has developed a test which is called The Basic Programming Knowledge Test (BPKT). This test was developed and validated with a group of naval training programmers and also with some outside agencies such as RAND, SDC, etc. Specifically, the test is designed to evaluate six different abilities: first, logic estimation and analysis; second, flow diagramming; third, programming constraints; fourth, coding operations; fifth, program testing and checking; and sixth, documentation. Not only does the test evaluate the person's performance in these areas, but it is also designed to elicit information regarding his basic knowledge of the areas. Similar tests for different types of programmers at different levels would be very effective evaluators. The problem, of course, is the resistance the programmers show particularly as it applies to experienced ones who are very much in demand.

We have cited the PAT, we have cited the Strong Vocational Interest Blank, we have cited the TSI.

Now then, the PAT is an aptitude test, presumably, and the SVIB is an interests test; the TSI presumably tests a kind of logical capability. If I use all these tests and get good scores on all of them, does it mean I selected a good programmer? The answer at the moment is - no. I don't think it means that I have selected a good programmer - but it may well increase the probability that I have selected one. We have not been able to show at this point, with the exception of the recurring interest pattern of programmer personnel, any strong indication of substantially increasing the probability of correctly selecting a programmer by the use of this battery.

Very probably if you would use all of the tests to select an individual, you can obtain a person who has a high probability of successfully completing your training program. Whether this individual is going to like programming or will possess the motivation that will allow him to take the successful training onto the job site is a question that is not yet answered.

Well, then I think we come to a small denouement, and it is that if we look at the past five years of computer personnel research, the major effort has been in the development of two sets of testing predictors: (Figure 16) testing for training success, and testing for job performance. We have said that we have good predictors for the training phase and questionable ones for the working phase. Temperament and motivation tests are frequently considered as good predictors of training success and job performance, but we have no research using them. Finally, of the "work sample" tests, the only proficiency test I know of is Berger's Basic Programmer Knowledge Test. I understand that he will publish a version of it in the public domain, by permission of the Navy. For evaluation procedure, at the moment this consists of the Programmer Appraisal Instrument (PAI) which we developed but have not used. This I think sums up the situation at the moment. A more concise summary of our current knowledge is that the more intelligent person you can find, the better programmer you can probably get... which makes all of us here today good programmers!

We might mention this point, that a test does exist that claims to be in this general area. This is the DPMA Certification program. However, I think we should note that it is quite questionable that there exists a relationship between this test and what we might identify as any level of programming skill. The DPMA test is primarily a knowledge test, but not at a very sophisticated level. It is my understanding that it is general knowledge -- at least that is my impression from the groups that were formed to study for it.

To summarize: We have called for several new procedures which should be investigated in the years to come. Despite the fact that I have been told time and time again by my psychologist friends that my or other people's interviewing technique cannot be a really effective device, it is still the one that I, as a computer manager, use in at least 50 percent of my evaluation of the candidate. Hopefully, if these new procedures can be developed, both I and my fellow managers will be better able to select, train, evaluate, and reward our computer personnel. Thank you everyone for your kind attention today.

BIBLIOGRAPHY

1. Dickmann, R. A., "A Survey of Computer Personnel Selection Methodology," Proceedings of the Fourth Annual Computer Personnel Research Conference pp. 15-27.

2. Lothridge, Charles, "Levels of Classifying Data-Processing Personnel," Proceedings of the Second Annual Conference Computer Personnel Research Group, pp. 13-28.

3. Berger, R.M. and Wilson, R. C., "The Development of Programmer Evaluation Measures," Proceedings of the Third Annual Computer Personnel Research Conference, pp. 6-17.

4. Dickmann, R.A., "A Programmer Appraisal Instrument," Proceedings of the Second Annual Conference Computer Personnel Research Group, pp. 45-64.

5. Bairdain, E.F., "Research Studies of Programmers and Programming," IBM Corporation, 1964 pp. 78, 136, 62.

6. Reinstedt, R.N. et al., Computer Personnel Research Group Programmer Performance Prediction Study (RM-4033-PR), The RAND Corporation, Santa Monica, California, March 1964.

7. Biamonte, A.J., "A Study of the Effect of Attitudes on the Learning of Computer Programming," Proceedings of the Third Annual Computer Personnel Research Conference, pp. 68-74.

8. Perry, D.K. and Cannon, W.M, "A Vocational Interest Scale for Computer Programmers - Final Report," Proceedings of the Fourth Annual Computer Personnel Research Conference, pp. 61-82.

9. Stalnaker, A.W., "The Watson-Glaser Critical Thinking Appraisal as a Predictor of Programming Performance," Proceedings of the Third Annual Computer Personnel Research Conference, pp. 75-78.

10. Berger, R.M. and Wilson, R.C., "Correlates of Programmer Proficiency," Proceedings of the Fourth Annual Computer Personnel Research Conference, pp. 83-95.

11. Biamonte, A.J., "Predicting Success in Programmer Training, Proceedings of the Second Annual Conference, Computer Personnel Research Group, pp. 9-12

12. Gotterer, M and Stalnaker, A.W., "Predicting Programmer Performance Among Non-preselected Trainee Groups," Proceedings of the Second Annual Conference Computer Personnel Research Group, pp. 29-44

13. Herwitz, P.S., "Programmer Evaluation" GUIDE Organization Talk, Management and Administration Committee, 1966 (IBM, Armonk, N. Y.), 12 pp.

14. Rigney, J.W. and Berger, R.M., "Computer Personnel Selection and Criterion Development: II. Description and Classification of Computer Programmer and Analyst Jobs," Tech. Rpt. 37, Proj. NR 153-093, Dept. of Psych., U. of Southern Calif., Dec. 1963, pp. 26 ff.

	United States	Canada
Organizations Participating	483	98
Programmer/Analysts Involved	23,636	1,083
Approximate Number Hired Each Year	5,317 (25%)	177 (20%)

	United States		Canada	
	Number	Percent	Number	Percent
AIRCRAFT INDUSTRY (Industrial, Aerospace)	47	10	3	3
ELECTRONIC INDUSTRY (Industrial, Electrical-Electronic)	35	7	2	2
OTHER INDUSTRY (Petroleum, Metal, Automotive, etc.)	120	25	34	35
FINANCE (Banks, Insurance Companies, etc.)	81	17	21	22
RESEARCH (Non-profits, University Labs, etc.)	90	19	10	10
GOVERNMENT (Federal, State, and City Civil Service)	50	10	8	8
UTILITY AND OTHER NON-MANUFACTURING CONCERNS	60	12	20	20
	483	100	98	100

From Dickmann, Ref. 1

FIGURE 1: TYPE OF ORGANIZATIONS IN 1966 CPRG SURVEY

	United States	Canada
Business	186	58
Business and Scientific	84	11
Business and Scientific and Software	72	6
Scientific	44	6
Scientific and Software	34	1
Business and Software	33	1
Business and Scientific and Software and Military	12	0
Software	6	0
Military	4	0
Scientific and Software and Military	3	0
Business and Military	2	0
Business and Software and Military	1	0
Other	2	1

From Dickmann, Ref. 1

FIGURE 2: PROGRAMMING STAFF APPLICATIONS

	Programmer Trainee		Experienced Programmer		System Analyst Trainee		Experienced Systems Analyst	
	U.S.	Canada	U.S.	Canada	U.S.	Canada	U.S.	Canada
None Specified	9	14	9	10	7	4	8	7
High School	27	65	19	43	13	32	11	28
Some College	25	3	23	6	12	16	14	10
College Graduate	34	13	35	8	43	9	40	11
Graduate Degree	1	2	1	2	2	7	5	7
Not Reported	4	3	13	31	23	32	22	37
	100%	100%	100%	100%	100%	100%	100%	100%

From Dickmann, Ref. 1

FIGURE 3: EDUCATIONAL REQUIREMENTS

	Programmer Trainee		Experienced Programmer		Systems Analyst Trainee		Experienced Systems Analyst	
	Non-Test	Test	Non-Test	Test	Non-Test	Test	Non-Test	Test
None Specified	6	10	6	10	5	8	7	8
High School	16	32	6	25	4	17	3	15
Some College	27	24	19	25	10	14	7	17
College Graduate	37	32	53	27	50	39	48	36
Graduate Degree	3	1	4	0	4	1	10	3
Not Reported	11	1	12	13	27	21	25	21
	100%	100%	100%	100%	100%	100%	100%	100%

From Dickmann, Ref. 1

FIGURE 4: COMPARISON OF EDUCATIONAL REQUIREMENTS FOR ORGANIZATIONS USING TESTS IN SELECTION VERSUS ORGANIZATIONS NOT USING TESTS (United States Sample)

TEST NAME	FREQUENCY OF USE* United States	Canada	VALIDATION Studies (Total)
GENERAL INTELLIGENCE TESTS			
Wonderlic Personnel Test	60	7	18
Thurstone Test of Mental Alertness	12	1	4
Otis Tests (Unspecified)	11	0	5
School and College Ability Tests (SCAT)	5	0	2
Wesman Personnel Selection Test	3	0	0
Ship Destination Test	3	0	0
Lowry Lucier Reasoning Test Combination	3	0	2
Concept Mastery Test	2	0	1
Henmon Nelson Tests of Mental Ability	2	0	2
Schubert General Ability Battery	2	0	0
APTITUDE TESTS AND BATTERIES			
IBM Programmer Aptitude Test	282	67	83
National Cash Register Programming Aptitude Test (E51)	9	4	6
Federal Service Entrance Exam	13	0	3
SRA Computer Programmer Aptitude Battery (Burroughs Corp.)	5	3	1
Employee Aptitude Survey	7	0	3
Differential Aptitude Tests	6	0	4
Watson Glaser Critical Thinking Appraisal	5	1	3
Short Employment Tests	5	0	1
Test of Sequential Instructions	2	0	1
Minnesota Clerical Test	2	0	1
Guilford Zimmerman Aptitude Survey	2	0	1
Bennett Mechanical Comprehension Test	2	0	2

*For Tests Used Two or More Times

From Dickmann, Ref. 1

FIGURE 5: TESTS USED FOR INTERVIEWING PROGRAMMER CANDIDATES: 1

TEST NAME	FREQUENCY OF USE* United States	Canada	VALIDATION Studies (Total)
PERSONALITY TESTS			
Thurstone Temperament Schedule	3	1	1
Activity Vector Analysis	3	0	3
Rohrer-Hibler-Replogle Personality Test	2	0	0
Humm-Wadsworth Temperament Scale	2	0	1
Edwards Personal Preference Schedule	2	0	0
Cleaner Self Description	2	0	0
Adaptability Test	2	0	0
Guilford Martin Inventory of Factors	1	0	1
Guilford Martin Temperament Profile Chart	1	0	1
INTEREST TESTS			
Kuder Preference Record	2	0	2
OTHER TESTS			
Manhattan Symbol (MAZE)	4	0	2
1401 Autocoder Exam	3	0	0
GCT	2	0	1
LOMA	2	0	1
Personagraph	2	0	0

*For Tests Used Two or More Times

From Dickmann, Ref. 1

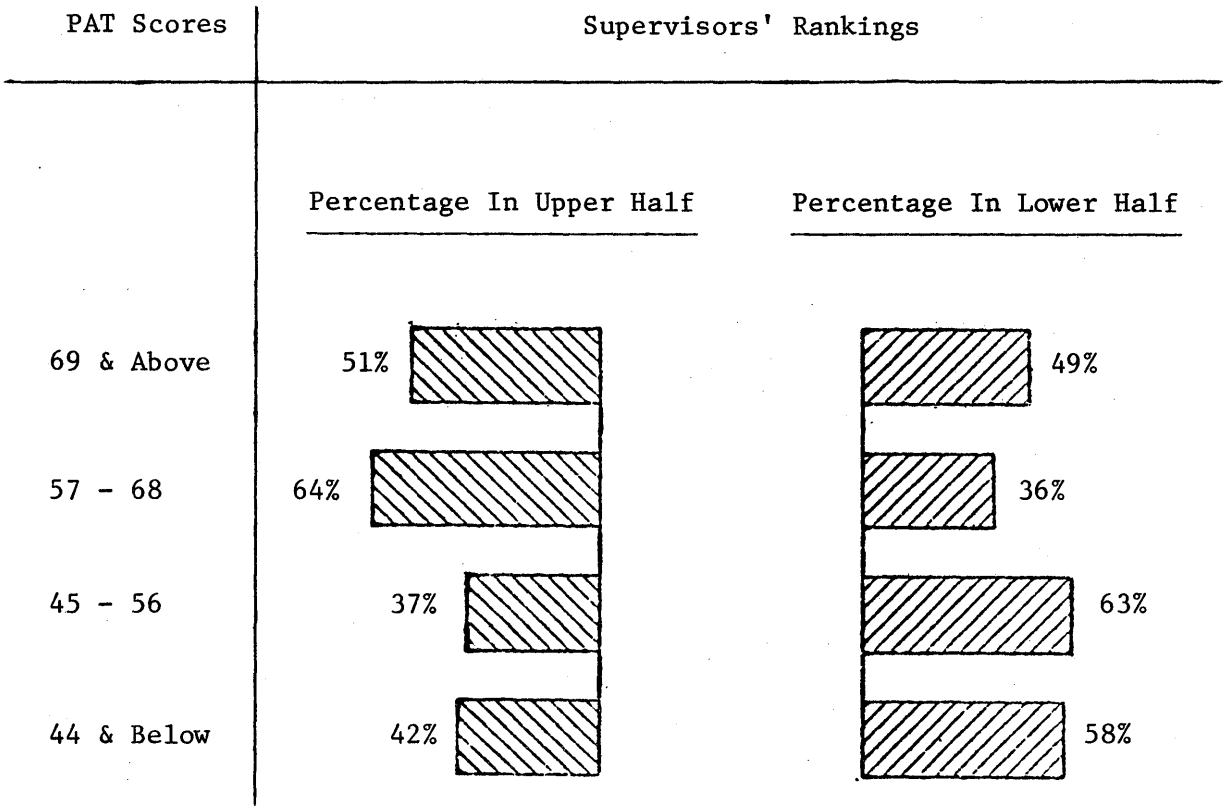
FIGURE 6: TESTS USED FOR INTERVIEWING PROGRAMMER CANDIDATES: II

	UNITED STATES	CANADA
NUMBER OF ORGANIZATIONS USING:	282	67
IN COMBINATION WITH OTHER TESTS:	128	18
ALONE:	154	49
PERCENT OF TOTAL SAMPLE:	58%	68%
PERCENT OF THOSE USING TESTS:	85%	93%
LEVELS OF TEST USE:		
PROGRAMMER TRAINEES:	278	67
EXPERIENCED PROGRAMMERS:	138	27
SYSTEMS ANALYST TRAINEES:	142	32
EXPERIENCED SYSTEMS ANALYSTS:	87	14
VALIDATION STUDIES:	71	12
DISCONTINUATION:	22	0

From Dickmann, Ref. 1

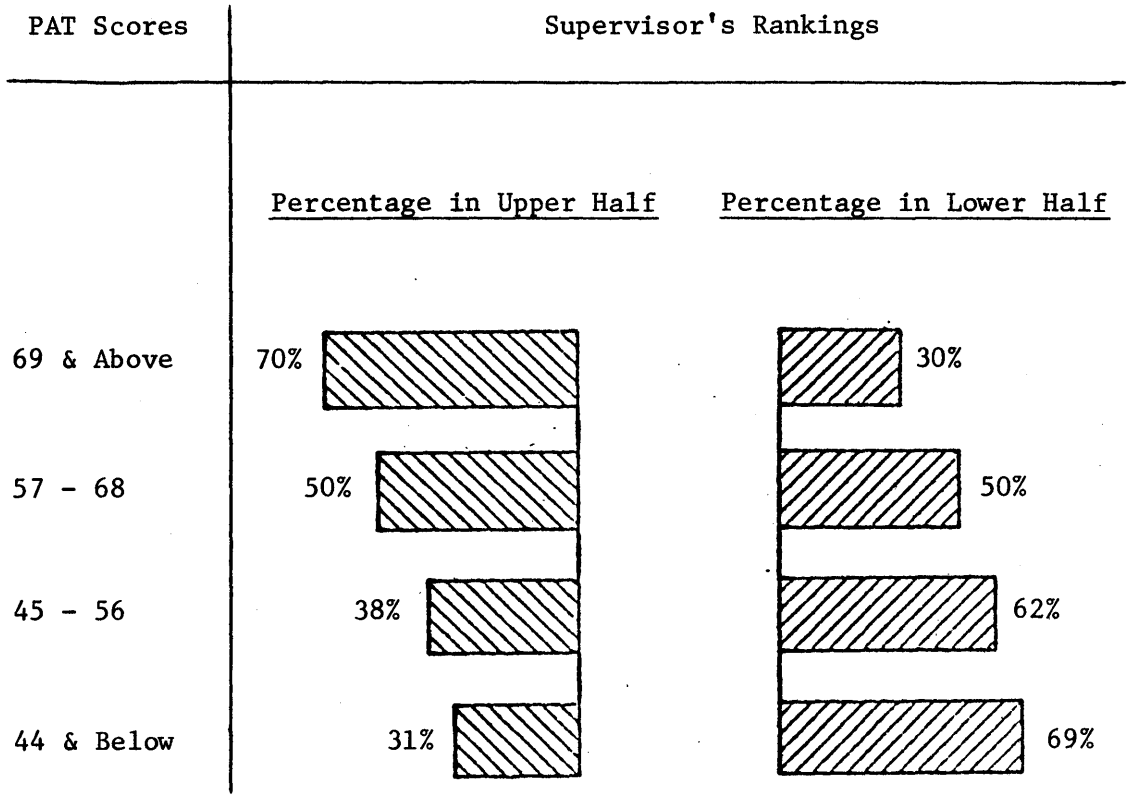
FIGURE 7: 1966 CPRG SURVEY

IBM PROGRAMMER APTITUDE TEST



From Reinstedt, Ref. 6

FIGURE 13: RELATIONSHIP BETWEEN PAT SCORES AND RANKINGS -- BUSINESS GROUP



From Reinstedt, Ref. 6

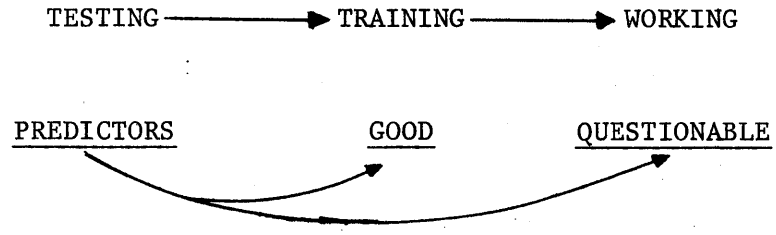
FIGURE 14: RELATIONSHIP BETWEEN PAT SCORES AND RANKINGS --SCIENTIFIC GROUP

	PROGRAMMERS			GENERAL POPULATION		
	Like	Indifferent	Dislike	Like	Indifferent	Dislike
Aviator	67%	21%	12%	30%	36%	34%
Mathematics	90	8	3	69	20	11
Solving Mechanical Puzzles	63	29	9	39	34	27
Giving First Aid	22	51	27	40	42	18
Progressive People	41	42	17	85	11	4
Conservative People	84	15	1	56	35	9
Energetic People	14	48	38	89	9	2
Thrifty People	45	44	11	74	22	4
Sell Machines	8	27	65	26	32	42

A	B	Prefer A	Indifferent	Prefer B	Prefer A	Indifferent	Prefer B
Few Details -	Many Details	18	27	55	36	28	36
Technical -	Supervision	65	16	19	34	19	47

FIGURE 15: STRONG VOCATIONAL INTEREST BLANK

INTERESTS OF PROGRAMMERS AND GENERAL POPULATION BY PERCENTAGES



PROFICIENCY

Programmer's Aptitude Test (PAT)

Buchannon Logical Test (BLT)

Strong Vocational Interest Blank (SVIB)

Temperament Tests (None)*

Watson Glaser Critical Thinking Analysis

*Motivation, primarily

BPKT - Basic Programmer's Knowledge Test

EVALUATORS

(PAI) Programmer's Appraisal Instrument

FIGURE 16: 5 YEARS OF COMPUTER PERSONNEL RESEARCH

ISSUES - '67

IN COMPUTER PERSONNEL RESEARCH

- EFFECTIVE EVALUATION PROCEDURES
- STRATIFICATION OF SKILLS
- NEW OBSERVATIONAL PROCEDURES
- ROLE OF CREATIVITY IN PROGRAMMING

FIGURE 17: ISSUES IN PERSONNEL RESEARCH - 1967

	PROGRAMMERS			AUDIENCE		
	Like	Indifferent	Dislike	Like	Indifferent	Dislike
Aviator	67%	21%	12%	85%	15%	0
Mathematics	90	8	3	99	0	1
Puzzles	63	29	9	25	65	10
First Aid	22	51	27	5	55	40
Progressives	41	42	17	70	27	3
Conservatives	84	15	1	5	80	15
Energetic People	14	48	38	75	24	1
Thrifty People	45	44	11	10	75	15
Selling Machines	8	27	65	5	85	10
Like Details:	Few	Indiff.	Many	Few	Indiff.	Many
	18	27	55	15	45	40
	Technical	Indiff.	Supervise	Technical	Indiff.	Supervise
Technical (vs. Supervision)	65	16	19	45	10	45

FIGURE 18: A COMPARISON OF THE RESULTS ON SAMPLE QUESTIONS TAKEN FROM THE SVIB, AS INDICATED BY THE AUDIENCE POLL AT COMMON, DEC. 12, 1967. MOST OF THE AUDIENCE (WHICH WAS 75 PEOPLE) SAID THEY WERE PROGRAMMING OR OTHER SUPERVISION, BUT A VERY LARGE PROPORTION ALSO PROGRAMMED WHILE SUPERVISING. THESE RESULTS ARE DISPLAYED ONLY FOR INTEREST: NO CONCLUSIONS FROM THE SAMPLE OR INDIVIDUAL QUESTIONS MAY BE DRAWN.

DBM/12/67

Team	Game # 1	2	3	Total 3 Games
Abel's	390	420	476	1286
Baker's	419	501	427	1347
Charley's	289	394	325	1006

In this exercise
1A 1B 1C

each word has a code
1D 1E 2A 2B 2C

letter and number
2D 2E 3A

combination beneath it.
3B 3C 3E

For example, the word "has" in the previous sentence has
3E 4A 4B 4C 4D 4E 5A 5B 5C 5D

the code 2A beneath it. In the first sentence of this
5E 6A 6B 6C 6D 6E 7A 7B 7C 7D 7E

paragraph, circle the code combination of the first occurrence
8A 8B 8C 8D 8E 9A 9B 9C 9D

of the word "code"; you should have circled 2C,
9E 10A 10B 10C 10D 10E 11A 11B 11C

thus: code.
11D (2C)

Now; beginning with the next sentence, circle the
12A 12B 12C 12D 12E 13A 13B 13C

code letters of all words beginning with the letter "w."
13D 13E 14A 14B 14C 14D 14E 15A 15B 15C

Also, each time a sentence begins with any vowel except
15D 16A 16B 16C 16D 16E 17A 17B 17C 17D

"a", circle the codeletter of its first word. Meanwhile, if the
17E 18A 18B 18C 18D 18E 19A 19B 19C 19D 19E

number 15 is greater than the number 25, circle the code
20A 20B 20C 20D 20E 21A 21B 21C 21D 21E 22A

letters of the third word in the previous sentence;
22B 22C 22D 22E 23A 23B 23C 23D 23E

otherwise, use the third word in this sentence. You may
24A 24B 24C 24D 24E 25A 25B 25C 25D 25E

halt your search for sentences beginning with "u", as well
26A 26B 26C 26D 26E 27A 27B 27C 27D 27E

as "a."
28A 28B

Team	Game # 1	2	3	Total 3 Games
Abel's	390	420	476	1286
Baker's	419	501	427	1347
Charley's	289	394	325	1006

Observe 30A the 30B bowling 30C scores 30D of 30E the 31A three-man 31B 31C teams 31D in 31E the 32A box 32B at 32C the 32D top 32E of 33A the 33B page. 33C

If 33D Baker's 33E team's 34A third 34B game 34C was 34D his 34E second 35A highest, 35B then 35C circle 35D the 35E code 36A under 36B the 36C second 36D occurrence 36E of 37A the 37B word 37C "the" 37D in 37E this 38A paragraph; 38B otherwise 38C circle 38D code 38E "36C." 39A

You 39B may 39C now 39D halt 39E looking 40A for 40B words 40C beginning 40D with 40E the 41A letter 41B "w" 41C at 41D the 41E end 42A of 42B this 42C sentence. 42D And 43A you 43B may 43C also 43D halt 43E looking 44A for 44B all 44C those 44D vowels, 44E beginning 45A with 45B the 45C next 45D sentence. 45E And 46A for 46B your 46C last 46D tasks, 46E if 47A Abel's 47B team 47C score 47D total 47E was 48A higher 48B than 48C Baker's 48D total 48E then 49A circle 49B codes 49C "40A" 49D and 49E "42A." 50A Otherwise, 50B circle 50C "40B" 50D and 50E "42B" 51A unless 51B Charley's 51C second 51D lowest 51E game 52A was 52B less 52C than 52D Abel's 52E lowest 53A game, 53B in 53C which 53D case 53E circle 54A the 54B code 54C underneath 54D all 54E words 55A with 55B an 55C "n," 55D in 55E this 56A sentence. 56B

1. 2C
2. 17A
3. 18E
4. 19A
5. 19C
6. 23A
7. 24E
8. 27B
9. 27E
10. 30A
11. 33D
12. 34D
13. 36C
14. 37C
15. 40C
16. 41C
17. 50E
18. 51B
19. 51D
20. 52D
21. 53C
22. 54D
23. 55C
24. 55D
25. 55E
26. 56B

Session No. T.2.2

Session Title: 1130 as a Terminal for S/360

Chairman: L. D. YARBROUGH

Speaker: Miss Kaye Gabbert

Topic:: Multiprocessing and RJE with 1130 - S/360

Summary: Several programs are available (types I, II, III) for using the 1130 as a powerful terminal, supporting RJE and graphics, to a S/360 (/30 or larger) NO IBM HANDOUT

Speaker:

Topic:

Summary;

Speaker:

Topic:

Summary:

NO. OF ATTENDEES: NOT RETURNED TO DAD

Dad

Kaye Gabbert

1130 As A Terminal For S/360 Using The Synchronous Communications Adapter

TYPES OF COMMUNICATIONS:

- SYNCHRONOUS TRANSMIT-RECEIVE (STR)
- BINARY SYNCHRONOUS COMMUNICATION (BSC)

1130 TYPE I PROGRAMS

STR
- SCAT 1

BSC
- SCAT 2
- SCAT 3
- REMOTE JOB ENTRY - WORK STATION PROGRAM
- 1130/2250 GRAPHIC SUBSYSTEMS

o SCAT1 STR

* AVAILABLE : NOW
* INTERRUPT SERVICE SUBROUTINE (ISS)
* POINT-TO-POINT COMMUNICATION
* SWITCHED OR LEASED NETWORK
* 4-OF-8 TRANSMISSION CODE (MAX. 64 DATA CHARACTERS)
* SIZE : 1078

SCAT 1 STR 0

SUPPORTED SYSTEMS:

* IBM S/360,
  MODEL 30, 40, 44, 50, 65, 67, 75
  WITH THE IBM 2701
  DATA ADAPTER UNIT
  EQUIPPED WITH
  SYNCHRONOUS DATA
  ADAPTER - TYPE I
* IBM S/360, MODEL 20
  EQUIPPED WITH THE
  COMMUNICATIONS
  ADAPTER (#2073)

SCAT 1 STR

SOFTWARE SYSTEMS:

* S/360 BOS/BPS
TYPE I PROGRAM
AVAILABLE: NOW

* S/360 MOD 20 CIOCS
TYPE I PROGRAM
AVAILABLE: NOW

* S/360 DOS/OS STRAM
TYPE II PROGRAM
AVAILABLE: NOW

SCAT1 STR

CALLED BY AN ASSEMBLY LANGUAGE PROGRAM

```
LIBF   SCAT1
DC     (CONTROL PARA.)
DC     (I/O AREA ADDR.)
DC     (ERROR ROUTINE
           ADDRESS)
```

IDEAL STR

STR COMMUNICATION IN IBM FORTRAN

- BCD, EBCDIC, BINARY I/O

TYPE III PROGRAM
AVAILABLE: NOW
FILE # 1130-03.8.004

FOR THE IBM 1130
FORTRAN USER

- SCAT 2 BSC

* AVAILABLE: NOW
* INTERRUPT SERVICE SUBROUTINE (ISS)
* POINT-TO-POINT
* SWITCHED OR LEASED
- NETWORK

* TRANSMISSION CODE
  - NORMAL EBCDIC
  - FULL-TRANSPARENT
* SIZE: 1066

-

SCAT 3 BSC

* AVAILABLE: NOW
* INTERRUPT SERVICE SUBROUTINE (ISS)
* MULTI-POINT TRIBUTARY STATION
* LEASED NETWORK
* TRANSMISSION CODE
  - NORMAL EBCDIC
  - FULL-TRANSPARENT
* SIZE: 1256

SCAT2-SCAT3 BSC

SUPPORTED SYSTEMS

* IBM S/360,
  MODEL 30, 40, 50, 65, 75
  - WITH IBM 2701 DATA ADAPTER UNIT
    WITH SDA-II
  - WITH IBM 2703 TRANSMISSION CONTROL UNIT
    WITH SYNCHRONOUS BASE I

SCAT2-SCAT3 BSC

SOFTWARE SYSTEMS

* S/360 DOS/BTAM
AVAILABLE: NOW

* S/360 OS/BTAM
AVAILABLE: APRIL 30, '68

SCAT2-SCAT3 BSC

* CALLED BY AN ASSEMBLY LANGUAGE PROGRAM

* TRANSMIT & RECEIVE DATA — PROGRAM SPECIFIED I/O AREAS

* OVERLAPPED OPERATION WITH OTHER I/O DEVICES

WHY USE BSC ?

* FULL-TRANSPARENT TEXT
  UNRESTRICTED CODING
  OF DATA WITHIN
  MESSAGES

* 'GROUND RULES'
  WELL-DEFINED

* ERROR CHECKING
  CRC-16

* MESSAGE HEADINGS

* MULTI-POINT

1130

Remote Job Entry Work Station Program

1130 RJE BSC

* CENTRAL STATION = S/360 OS/RJE

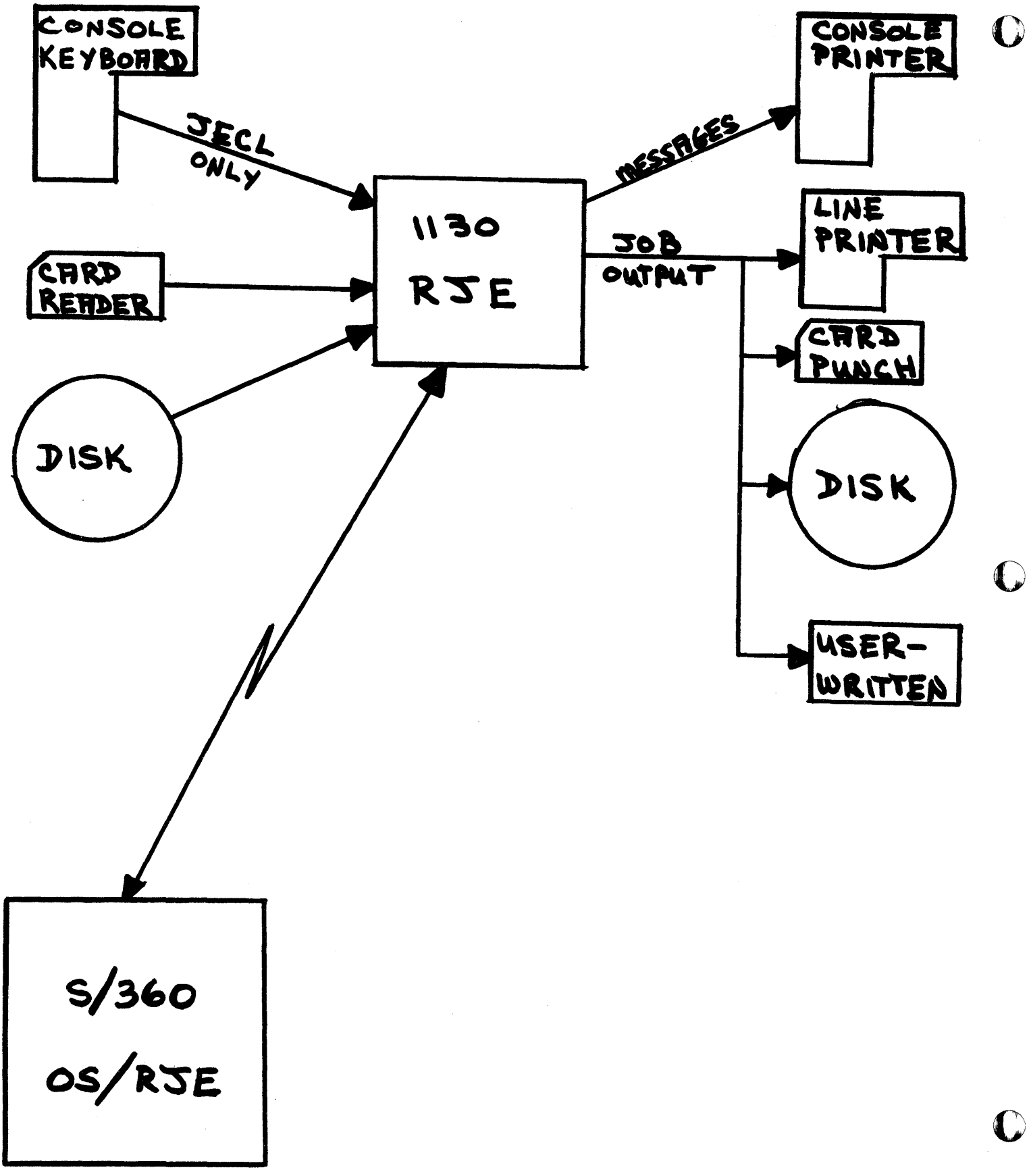
* AVAILABLE:
  - POINT-TO-POINT
    JULY 31, 1968
  - MULTI-POINT
    OCTOBER 31, 1968

* 1130 DISK MONITOR VERSION II
  - SCAT2
  - SCAT3

1130 RJE BSC

FUNCTIONS

- INPUT OF JOB ENTRIES & COMMANDS THRU AN ATTACHED INPUT DEVICE
- DATA TRANSMISSION TO & FROM CENTRAL
- WRITING OF JOB OUTPUT & MESSAGES TO OUTPUT DEVICES


CONSOLE KEYBOARD
JECL ONLY
CARD READER
DISK
1130 RJE
MESSAGES
JOB OUTPUT
CONSOLE PRINTER
LINE PRINTER
CARD PUNCH
DISK
USER-WRITTEN
S/360 OS/RJE

1130 RJE BSC

SECL FOR 1130 WORK STATION

SAME AS S/360 WORK STATION

EXCEPT FOR

.. DATA DMS, C

.. DATA DMS, D, XAAA, BBBB

SECTOR ADDRESS

BLOCK COUNT (OPTIONAL)

1130 RJE BSC

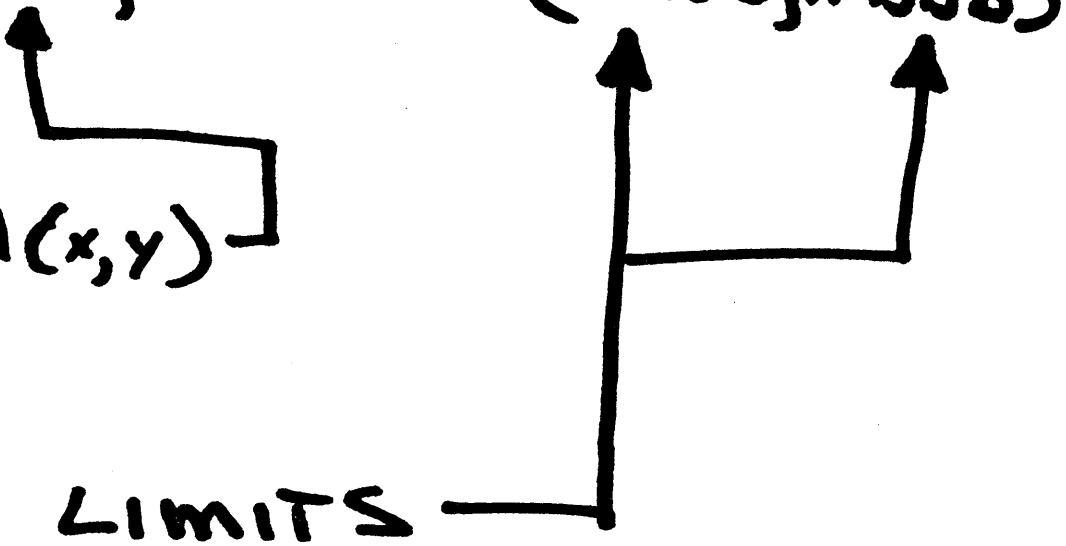
WHEN INPUT IS FROM DISK

- PREVIOUSLY STORED
- 80 CHAR. RECORDS
- 8-BIT PACKED CODE
- 8 RECORDS/SECTOR

1130 RJE BSC

GENERATION

```
// XEQ RJEφφ
LINE= X , UEXIT= (xaaaa,xbbbb)
```



P, D, OR M(x,y)

SECTOR LIMITS FOR STORING DATA DIRECTED TO THE USER-EXIT

1130 RJE

BSCO

EXECUTION

```
// XEQ RJE
.. RJSTART
[  JECL STATEMENTS
   & OS/360 JOBS  ]
.. RJEND
```

- 1130 RJE BSC
- ENTER JOBS INTO JOB STREAM
- EQUIPMENT CENTRALIZATION
- COMPUTING POWER ON A DEMAND BASIS RATHER THAN REGULAR
- SHARING OF A COMMON BODY OF INFORMATION

1130 RJE BSC

STRONG SYSTEM DISCIPLINE

- JOBS REQUIRING SET-UP MAY CAUSE SYSTEM JOB FLOW TO BE UPSET

- JOBS REQUIRING A LOT OF CORE CAN CAUSE A SYSTEM PROBLEM SINCE PROCESSING IS DELAYED UNTIL CORE IS AVAILABLE

1130/2250 GRAPHIC SUBSYSTEMS

1130/2250

* AVAILABLE: FEB 15, 1969

* AREAS OF SUPPORT

  - PROCESSOR-TO-PROCESSOR DATA TRANSMISSION *
  - DATA COMPATIBILITY SERVICES *
  - SATELLITE GRAPHIC JOB PROCESSOR

* 2250 NOT REQUIRED

1130/2250 NETWORK

* S/360 MODEL 40, 50, 65, 75 WITH 2701 OR 2703

* 1130 - 8K OR LARGER
  - WITH SCA AND DISK

* POINT-TO-POINT
  SWITCHED OR LEASED
  - IF SWITCHED, USER REQUIRED TO USE OWN PROCEDURES TO ESTABLISH CONNECTION

1130/2250

PROCESSOR-TO-PROCESSOR DATA TRANSMISSION

- FOR THE IBM 1130 FORTRAN USER
- SIMILAR DATA TRANSMISSION SUBROUTINES FOR 1130 AND S/360
- NO DETAILED KNOWLEDGE OF DATA LINK PROCEDURES IS NEEDED

1130/2250

TRANSMISSION SUBROUTINES

GTNIT -- INITIALIZE FOR COMMUNICATION

GTRED -- READ DATA

GTWRT -- WRITE DATA

GTCLT -- CONTROL TEST

GTEND -- END COMMUNICATION

1130/2250

DATA COMPATIBILITY SERVICES

- SIX DATA CONVERSION SUBROUTINES TO AND FROM 1130 AND S/360 DATA FORMAT
- ALL CONVERSION IS PERFORMED IN S/360

1130/2250

SATELLITE GRAPHIC JOB PROCESSOR

- EXTENSION OF GRAPHIC JOB PROCESSOR
- S/360 JOB CONTROL FROM 2250 ATTACHED TO A REMOTE 1130
- DEFINE A S/360 PROGRAM TO RUN IN CONJUNCTION WITH AN 1130 PROGRAM

1130/2250

- DEFINE AND INITIATE JOBS TO RUN INDEPENDENTLY IN S/360
- REQUESTS JOB CONTROL INFORMATION FROM A USER THROUGH A SERIES OF DISPLAYS
- UP TO 14 REMOTE 1130/2250 SYSTEMS ATTACHED TO A S/360

T. 2. 3(a)

Public Service Company of Colorado Uses the 1800 Data Acquisition and Control System in Gas Load Control

By Earl E. McLaughlin Jr.
Public Service Company
of Colorado

Presented Before the
1800 "Common" Conference
San Francisco, California
December - 1967

ABSTRACT

The Public Service Company of Colorado, a Gas and Electric utility, has installed the 1800 Data Acquisition and Control System in its Gas Load Control Center.

The Gas Load Control Center has the responsibility of regulating and maintaining system pressures, contracted gas purchases, and gas storage facilities in the Company's service area. The computer system is monitoring variables from the field, logging hourly flow and pressure data, and controlling system pressures.

This paper will explain the necessity and benefits of a real time computer system in the dispatching of natural gas.

CONTENTS

Service Area	1.0
Gas Supply	1.1
Purchase Contracts	1.2
Gas Storage	1.3
Industrial Customers	1.4
Pressure Systems	1.5
Weather	1.6
Gas Load Control Center	1.7
Computer Configuration	2.0
Telemeter Input	2.1
Data Checking	2.2
Data Processing	2.3
Operator Inquiry and Entry	2.4
Gas Load Forecasting	2.5
Data Logging	2.6
Low Pressure Control	2.7
Physical Planning	3.0
Future Planning	4.0
Computer Output	5.0

1.0 SERVICE AREA

Public Service Company of Colorado and its subsidiaries, Western Slope Gas Company, Cheyenne Light, Fuel and Power Company, and Pueblo Gas and Fuel Company serve natural gas to approximately 400,000 customers in Colorado and Southern Wyoming. Natural gas is distributed in the Company's service areas through extensive transmission and distribution systems.

A general system map is shown in Figure 1. This graphically illustrates the great distances and rough terrain that must be encountered in order to serve our customers. An example of this is the Western Slope Gas Company, Southern Division, transmission line that is 449 miles in length and crosses the Continental Divide five times.

1.1 SUPPLY

Gas utilized by Public Service Company and Western Slope Gas Company, Eastern Division, is purchased directly from Colorado Interstate Gas Company. Gas for this area is transported by Colorado Interstate from the Texas, Kansas, Colorado and Oklahoma gas fields and from San Juan gas fields in the Four Corners area by connection with another pipeline company. This represents an investment of over $202,000,000 by Colorado Interstate to serve its customers. Public Service and Western Slope represent over two-thirds of Colorado Interstate's business and have a direct relationship in establishing purchase contracts and rate structures in this area.

Gas utilized by Western Slope Gas, Western Division, is produced from gas fields in Western Colorado.

The Western Slope Gas Company, Southern Division, produces the gas it needs from the San Juan gas field in Southern Colorado.

1.2 PURCHASE CONTRACTS

Gas purchased by Public Service Company, Cheyenne, Pueblo, and Western Slope from Colorado Interstate is purchased under a two-part rate, consisting of a Contract Demand charge and a Commodity charge. The Commodity charge is the unit price on the actual volume of gas delivered by Colorado Interstate and the Demand Charge is the charge for the maximum volume of gas Colorado Interstate is obligated to deliver on the peak day.

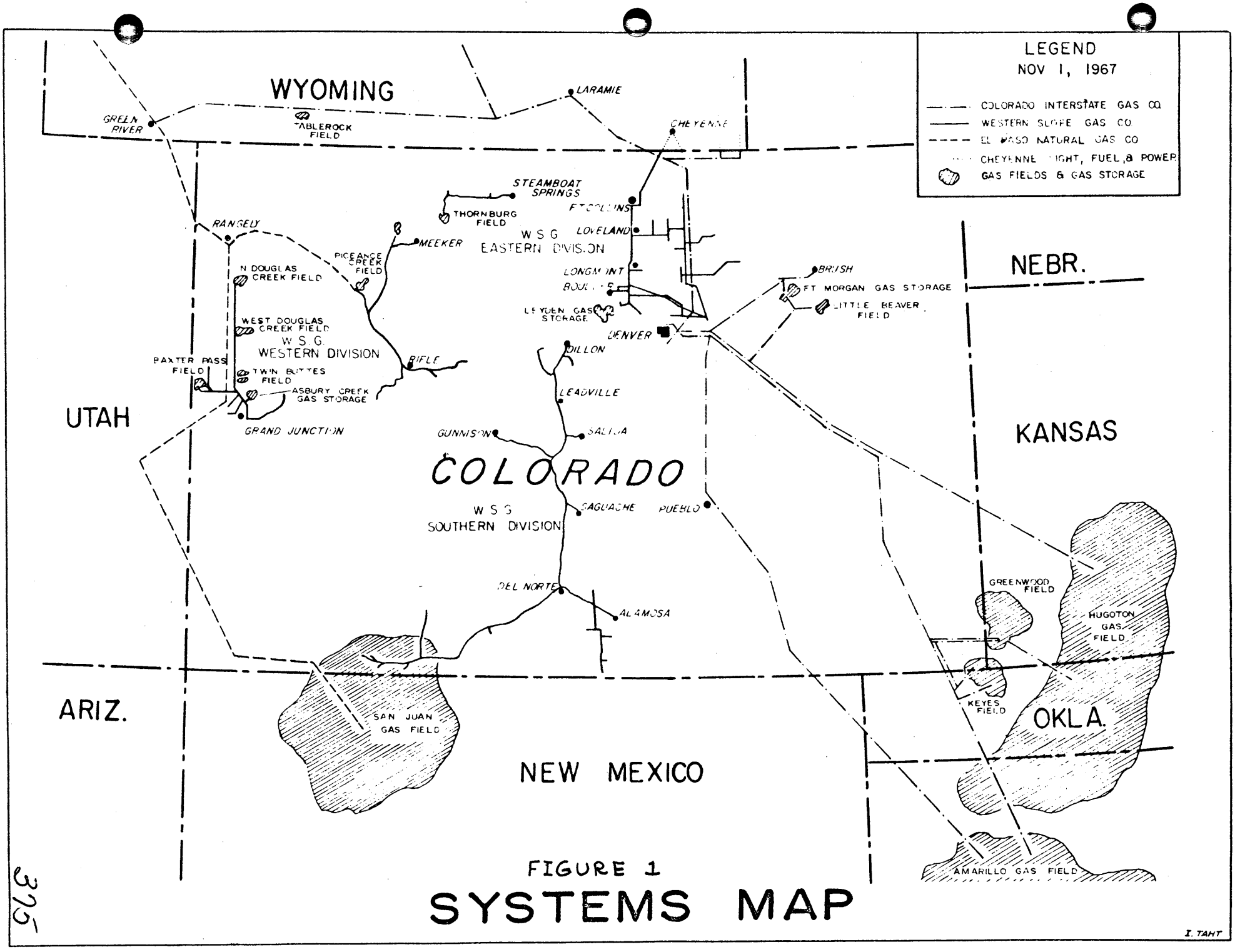


FIGURE 1

SYSTEMS MAP

I. TAHT

11-6710-5

325

Public Service and Western Slope are required to pay the Demand Charge on 100% of the Contract Demand each day and the Commodity Charge for the gas actually purchased. Penalties are charged the Company in the event that the Contract Demand limit is exceeded. As example of a substantial penalty, in 1963, Public Service and Western Slope paid in excess of $785,000 for gas that was purchased over the contract limits.

The justification of such a rate structure is based on the wide range of load conditions that occur in the Colorado area. On July 16, 1967, the mean temperature was 65 degrees F. with a 24-hour gas load of 107,000,000 cubic feet. January 7, 1967 produced a load of 565,000,000 cubic feet with a mean temperature of 16 degrees F. The mean temperature in Denver on January 11, 1963, was -17 degrees F.

It is apparent that pipeline capacity required to meet peak winter needs becomes idle during warm weather.

The two-part rate structure is needed to protect the pipeline's large plant investment that was made to serve areas during the peak load, while the load factor is at a minimum due to conditions such as warm weather.

1.3 STORAGE

It would be an advantage to Public Service Company of Colorado and Western Slope Gas Company to offset the varying load conditions by increasing the purchases from Colorado Interstate during off peak periods and decreasing the purchases on the peak day. This can be accomplished to some extent by storing gas during warm weather and withdrawing the same gas from storage when needed in cold weather.

Public Service Company has converted an abandoned coal mine near Denver, Colorado, to an underground natural gas storage reservoir. This project, called Leyden Storage, is capable of storing 2,500,000,000 cubic feet of gas of which 1,250,000,000 is considered usable for peak shaving at a cavern pressure of 250 psig. Leyden is connected to both Public Service and Western Slope service areas by a network of intermediate pressure systems (50-150 psi). An agreement also was made with Colorado Interstate for storage gas from its storage facilities near Fort Morgan, Colorado, to serve the Denver area. Both of these storage facilities are used during the heating season to control the load curve of Public Service Company of Colorado and Western Slope Gas companies.

A typical 24-hour winter load curve is shown in figure 2. For this particular day, all industrial customers were curtailed and 107,046,000 cubic feet of gas was withdrawn from Leyden storage to maintain the contracted limits. An average hour load is shown by the line at 15.6 MMCF.

1.4 INDUSTRIAL CUSTOMERS

Industrial customers in our service area are under contract to purchase interruptible gas. Interruptible Gas is gas that is available in excess of the amount required to serve the firm customer while still under the Contract Demand. The interruptible gas is sold to the industrial customer at a rate that is less than the amount paid by the firm customer. This rate class justifies the installation of a standby fuel system, such as, propane or oil.

When the predicted load in a service area is estimated to be greater than 100% of the contracted demand and above the amount that can be supplied from storage, the industrial customer is given notice that he must curtail the use of natural gas and switch to standby fuel. When the daily load decreases to less than 100% of contract, the industrial customer is notified that he may resume the use of natural gas.

There are over 450 industrial customers in the Company's service area and every attempt is made to keep curtailment at a minimum to optimize purchases and sales. The industrial load is approximately 20% of the contract on a cold day.

The load curve that is shown in Figure 2 is with all of the industrial customers curtailed.

1.5 PRESSURE SYSTEMS

In general, there are three types of pressure systems--High, Intermediate, and Low. The high pressure system is used for the transmission of gas over long distances. The operating range is from 200 psig to 1,000 psig. This type of system is usually a function of the transmission or pipeline company, such as Western Slope Gas Company. The high pressure system terminates at a city's town border station.

A town border station is the point where the distribution company purchases gas from the transmission company for its system. The Denver area has six major purchase stations.

Intermediate pressure systems operate in the range of about 20 to 150 psig. Their function is to carry the gas through the market area and supply all of the city's district regulator stations. In the Denver

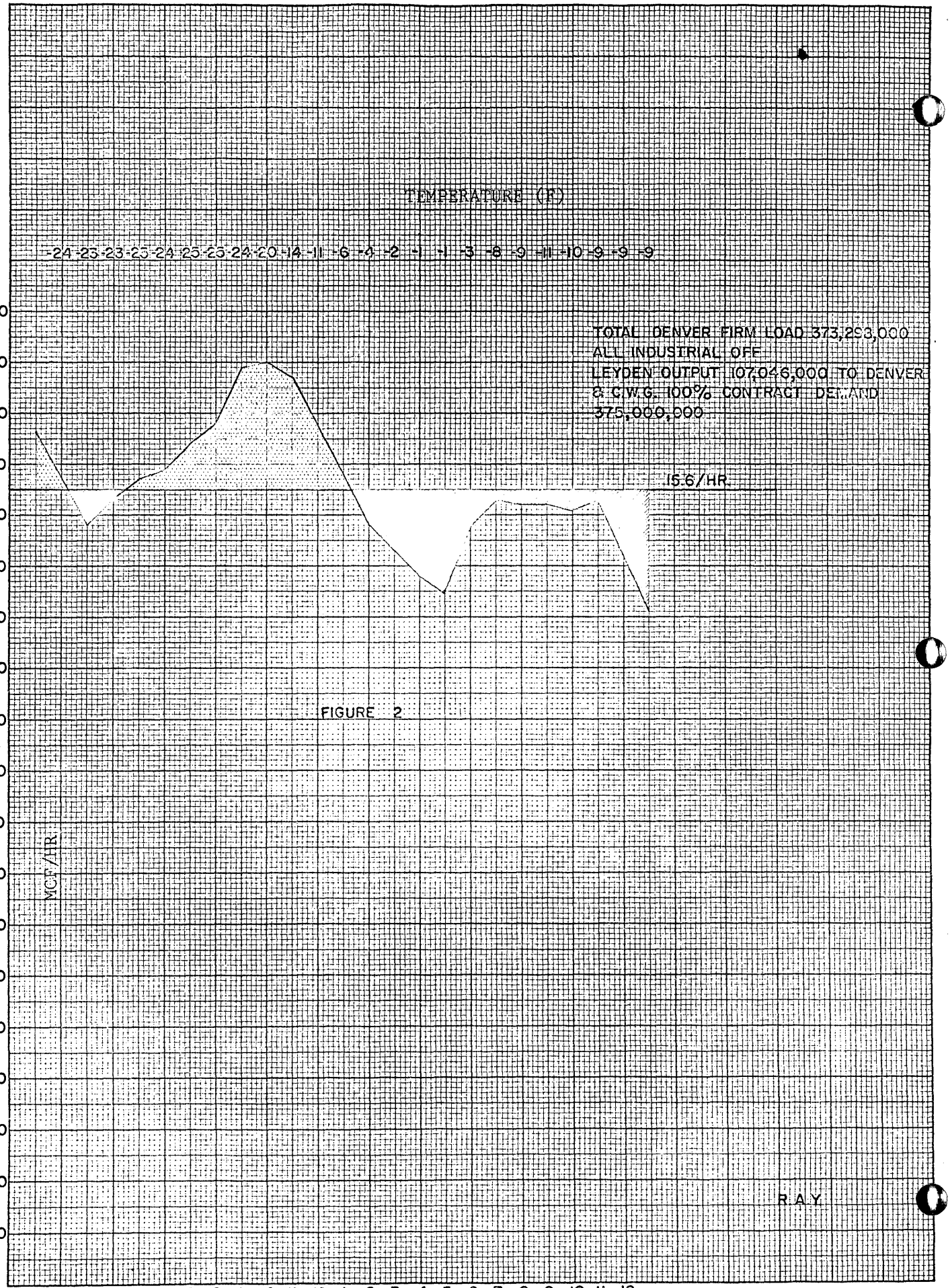


FIGURE 2

K&E 10 X 10 TO THE CENTIMETER 46 1513
18 X 25 CM. MADE IN U.S.A.
KEUFFEL & ESSER CO.

area, the intermediate pressure system connects most of the town border stations with Leyden Mine Storage. This enables peak shaving gas to be injected into the intermediate pressure system which will result in decreasing the purchased gas at the city's town border.

Low pressure systems can operate in the range of 2 psig upward to 60 psig. Their function is to distribute gas to the individual customer.

All of the pressure systems, along with compressors, valves, and regulators, provide a link that is hundreds of miles long which connects a customer in Colorado with the gas fields in Texas, Kansas, Wyoming, Oklahoma and Colorado.

1.6 WEATHER

Colorado area weather is the largest factor in determining the gas load. Weather forecasting and gas load forecasting are necessary to predict gas and pressure requirements. During the winter, it is not unusual for a morning to be 60 degrees F. with the sun shining and by afternoon of the same day have the temperature drop to 10 degrees F., with wind and snow. This is caused by a storm front and, unless the front is predicted, requirements for pressure and volume would be greater than the system could produce.

1.7 CONTROL CENTER

Public Service Company of Colorado, Western Slope Gas Company, Cheyenne Light, Fuel and Power Company, and Pueblo Gas and Fuel Company operate a Gas Load Control Center located in Denver, Colorado. This center is manned 24 hours a day, 7 days a week, and has the following responsibilities:

1 - To calculate and log all of the hourly flow totals from metering stations for all four companies. Pressures, differential pressures, and flowing gas temperatures are telemetered to the Load Control Center from 32 major metering points located throughout the combined service area. This information which totals 114 different inputs is used together with various factors to calculate the gas purchased each hour at each meter station. The formula for the calculation of gas through an orifice meter is shown in Figure 4. A drawing of an orifice meter is shown in Figure 5.

2 - To monitor and control (via phone conversation) compression facilities of the Western Slope Gas Company. Pressures from points located on the transmission lines are telemetered to the Control Center. This

$$Q = F_b F_{pb} F_g \left[1+(.68462-.26923\frac{d^4}{D^4})(\frac{h}{27.7PA})\right]\left[\sqrt{hPA+b'}\right]\sqrt{\frac{(aPA^2+bPA+c-1)(\frac{520}{TA})^{4.825}+\frac{520}{TA}}{1+\frac{h}{27.7PA}}}$$

Q = RATE OF FLOW IN CFH

QR = RATE OF FLOW IN MCFH = Q/1000 =]$^{32,767}_{0}$

F_b = BOFB = BASIC ORIFICE FACTOR]$^{50,000}_{200}$

F_{pb} = PBCF = PRESSURE BASE CORRECTION FACTOR FROM 14.73 TO OTHER = $\frac{14.73}{Pb}$ =]$^{1.0229 = 14.4\ Pb}_{1.0055 = 14.65\ Pb}$ $^{.9804 = 15.025\ Pb}$

d = DIAMETER OF ORIFICE]$^{12.0"}_{0.5"}$
D = DIAMETER OF PIPE]$^{20"}_{4"}$
} $(.68462-.26923\frac{d^4}{D^4}) = D4F$]$^{1}_{0}$

h = H = DIFFERENTIAL IN INCHES OF WATER AT T(60°F)]$^{100"}_{0}$

P = INPUT PRESSURE FROM METER STATION

PB = METERING PRESSURE BASE

PA = ABSOLUTE STATIC PRESSURE PSI = P+14.4
(ALL PRESSURES ARE SET FOR 14.4 & NOT AVERAGE BAROMETER]$^{1014.4}_{14.4}$

b' = BPRIM = CONSTANT FOR REYNOLDS NUMBER]$^{1}_{.01}$

a =]$^{1}_{10^{-6}}$ = 0.000,000,046
b = SUPER COMPRESSIBILITY FACTORS]$^{1}_{10^{-6}}$ = 0.000,148,500
c =]$^{1}_{10^{-6}}$ = 0.997,300

T = TEMPERATURE OF FLOWING GAS]$^{100°F}_{0}$

TA = ABSOLUTE TEMPERATURE (R°) OF FLOWING GAS]$^{560°R}_{460°R}$
= T + 460°F

G = SPECIFIC

F_g = SPFG = SPECIFIC GRAVITY FACTOR = $\sqrt{1/G}$]$^{2}_{1}$

FLOW SUBROUTINE

VALUES PLACED IN COMMON BY THE CALLING ROUTINE: QR, P, T, H, BOFB, PBCF, SPGF, D4F, BPRIM

VALUES PLACED IN COMMON BY THIS SUBROUTINE: QR

FIGURE 4

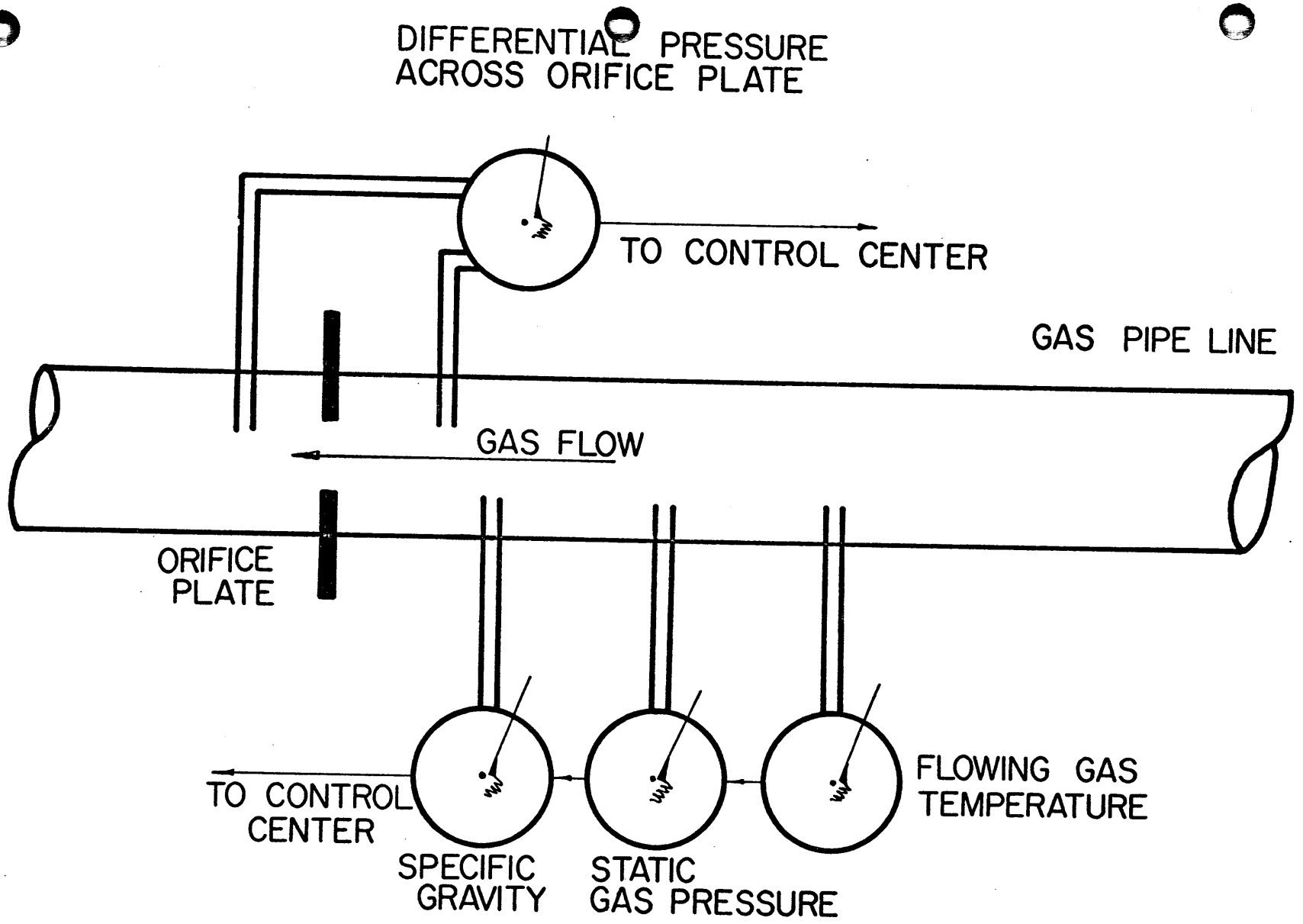


FIGURE 5
METER TUBE

information totals 39 input points. The operator can take action by ordering pressure output from a major compressor station or by notifying personnel in the area that needs attention.

3 - To monitor and control intermediate pressure systems in the Denver area. Pressures from points located on the intermediate pressure systems are telemetered to the Control Center. This information totals 42 input points. Control circuits, which set or regulate the pressure system, are activated from the control room. This totals 38 output points.

4 - To monitor and control low pressure systems in the Denver area. Pressures from the low pressure systems are telemetered to the control center, totaling 61 input points. Control circuitry from the control room to a regulator station enables the operator to adjust the regulator's output to satisfy the demand on that station. There are 54 output points from the control room to various regulator stations.

5 - To operate all of the above pressure systems in a manner that guarantees safety and service to the customer without interruption.

6 - To control purchase rates from suppliers and storage facilities maintaining optimum system requirements without exceeding 100% of the contract demand.

This requires the operator to estimate gas loads, weather conditions, pressure requirements, gas storage needs, and industrial loads throughout the operating day, and make the necessary decisions based on these estimates.

7 - To provide management with reliable operating statistics so that the future growth and operating guidelines can be determined.

The operation of a control center of this size requires that the operator maintain constant surveillance of all variables in the control room, calculate the flow rate for all meter stations, and predict the weather and gas load for storage requirements and industrial curtailment. With these responsibilities constantly increasing due to the continuing growth of the Company's gas service areas and more complex storage and system guidelines, a method was needed to relieve the operator of the time-consuming routine jobs. The time saved could be devoted to weather and system analysis.

The 1800 Data Acquisition and Computer Control system was installed in August 1967 in the Gas Load Control Center. The computer is assisting the operator by providing a constant monitor of all system

pressures along with flow calculation and pressure control. This system is also providing management with valuable statistical information and a higher degree of accuracy. The computer is enabling the operator to optimize the systems control. The chance that the 100% contract demand figure will be exceeded is much less and a higher load factor is being achieved. The computer's speed allows all the pressure inputs to be scanned at a continuous rate and if an erroneous condition occurs, it will alarm the operator as to what values need his attention. Thus, a higher degree of safety is obtained in pressure control. The computer is able to regulate a station's pressure output at a more constant rate than the operator can, which provides a lower average pressure throughout the system with resultant benefit in reducing the volume of lost and unaccounted for gas.

One of the most beneficial aspects of the computer system's ability to handle the routine duties of an operator is that the operator has more time to use his experience and knowledge for planning and supervision of the overall system.

The computer's data flow and configuration are explained starting at 2.0

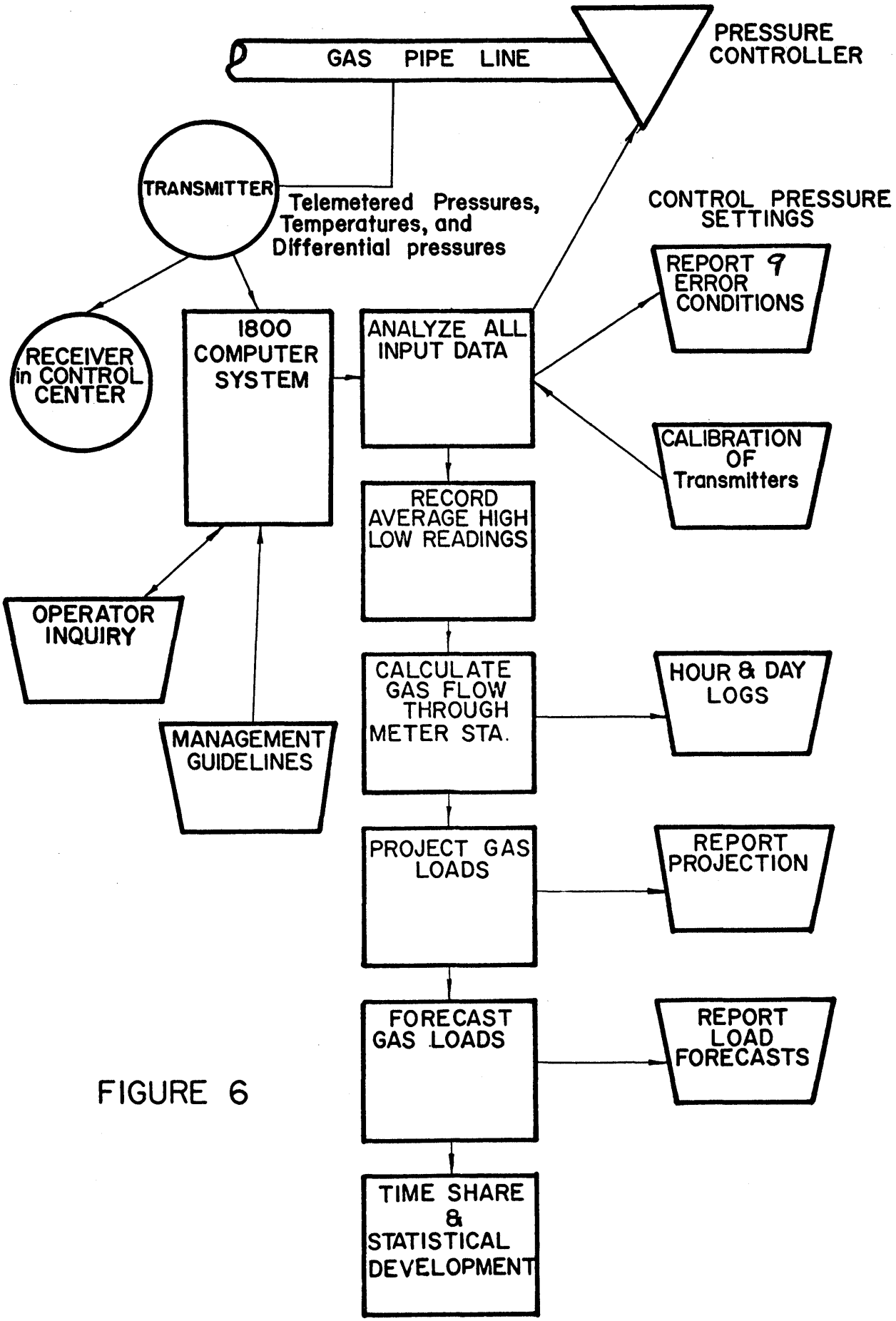

GAS PIPE LINE
PRESSURE CONTROLLER
TRANSMITTER
Telemetered Pressures, Temperatures, and Differential pressures
CONTROL PRESSURE SETTINGS
RECEIVER in CONTROL CENTER
1800 COMPUTER SYSTEM
ANALYZE ALL INPUT DATA
REPORT & ERROR CONDITIONS
CALIBRATION OF Transmitters
RECORD AVERAGE HIGH LOW READINGS
OPERATOR INQUIRY
CALCULATE GAS FLOW THROUGH METER STA.
HOUR & DAY LOGS
MANAGEMENT GUIDELINES
PROJECT GAS LOADS
REPORT PROJECTION
FORECAST GAS LOADS
REPORT LOAD FORECASTS
TIME SHARE & STATISTICAL DEVELOPMENT

FIGURE 6

2.0 COMPUTER CONFIGURATION

The 1800 computer system consists of the following:

1801 Process Controller - 4 micro-second-16K Core
12 Levels of Interrupt
2 - 2310 Disk Drives
1442 Card Reader
1443 Line Printer
3 - 1826 Input Units
2 - 1053 Typewriters
1816 Typewriter - Keyboard
Timer A - 1 ms
Timer B - 8 ms
Timer C - 8 ms
10 PISW Words on Level 0
10 PISW Words on Level 1
IBM "Time Sharing Executive" Operating System

The computer's General Data flow is shown in Figure 6. The explanation of the software and hardware used follows.

2.1 TELEMETER INPUT

The American Standards Association defines the word "telemetering" as "the indicating, recording or integrating of a quantity at a distance by electrical translating means." There are many types of telemetering systems available, and Public Service Company uses an impulse duration type. This system includes transmitters in the field and receivers in the Control Center.

The transmitter contains a measuring element, such as, a Bourdon tube. This element is attached through linkages to a cam follower. The cam follower rides on a linear cam that rotates 360 degrees every five seconds. The pressure in the element causes the cam follower to be positioned on the cam in relation to the pressure in the element. The cam follower is connected to a mercury switch and when the follower is on the cam, an electrical circuit is made. When the follower is off the cam, the circuit is broken. The transmitter generates an electrical impulse every five seconds. The length of the pulse is determined by the time that the cam follower is on the cam, which is determined by the amount of pressure that is applies to the measuring element. If the measuring element is

a Bourdon tube that is connected to a gas pipeline, an increase in gas pressure will cause the Bourdon tube to expand. This expansion will move the cam follower to a higher position on the cam which, in turn, will generate a longer time on the cam or a longer electrical pulse. A transmitter is shown in Figure 7. This pulse is transmitted over telephone lines to the Control Center. The pulse is connected to an electrical receiver which positions a recording pen on a chart that is in relation to the pressure on the transmitter in the field. The length of a pulse will vary from one second (0% pressure) up to four seconds (100% pressure). See Figure 8.

The Control Center receives 250 different pulses from points located on the gas system every five seconds. The distance the pulses travel varies from one mile up to 500 miles.

A method of inputting the values from the transmitter to the computer was needed. At first we planned on using the conventional digital input system. Various programming techniques could be used but all of them required the computer to scan the input data every 3 milliseconds (ms). The 3 ms scan was needed to maintain one tenth of one percent resolution on a 5 second input (out of the 5 second pulse, 3 seconds of the pulse are used for full scale, 0% thru 100%, 3 seconds = 3000 ms .001 of 3000 ms = 3 ms). Using a 4 micro second computer, it would take longer than 3 ms to scan the inputs. If the computer's speed was increased to 2 micro seconds, more than 50% of the computer's time would be spent reading these inputs. A more efficient method of input to the computer was needed. An input system using the power of the 1800 computer's interrupt structure was developed called "Time Duration Telemeter Input".

Time Duration telemeter input is intended for use with time duration signals that have the characteristics of a starting time, an elapsed time, and a stopping time. Input to the computer is achieved through Process Interrupt words. One hardware timer is used to keep account of the elapsed times. If Time Sharing Executive software is used, one modification to the Timer Section is needed. This modification enables one timer to run continuously without requesting service.

A signal that is sent from a transmitter is shown in Figure 9, Part A. Here we have a starting time (A), an elapsed time (X) and a stopping time (B). If the computer is interrupted at point (A) an interrupt servicing program can record the time of the interrupt using

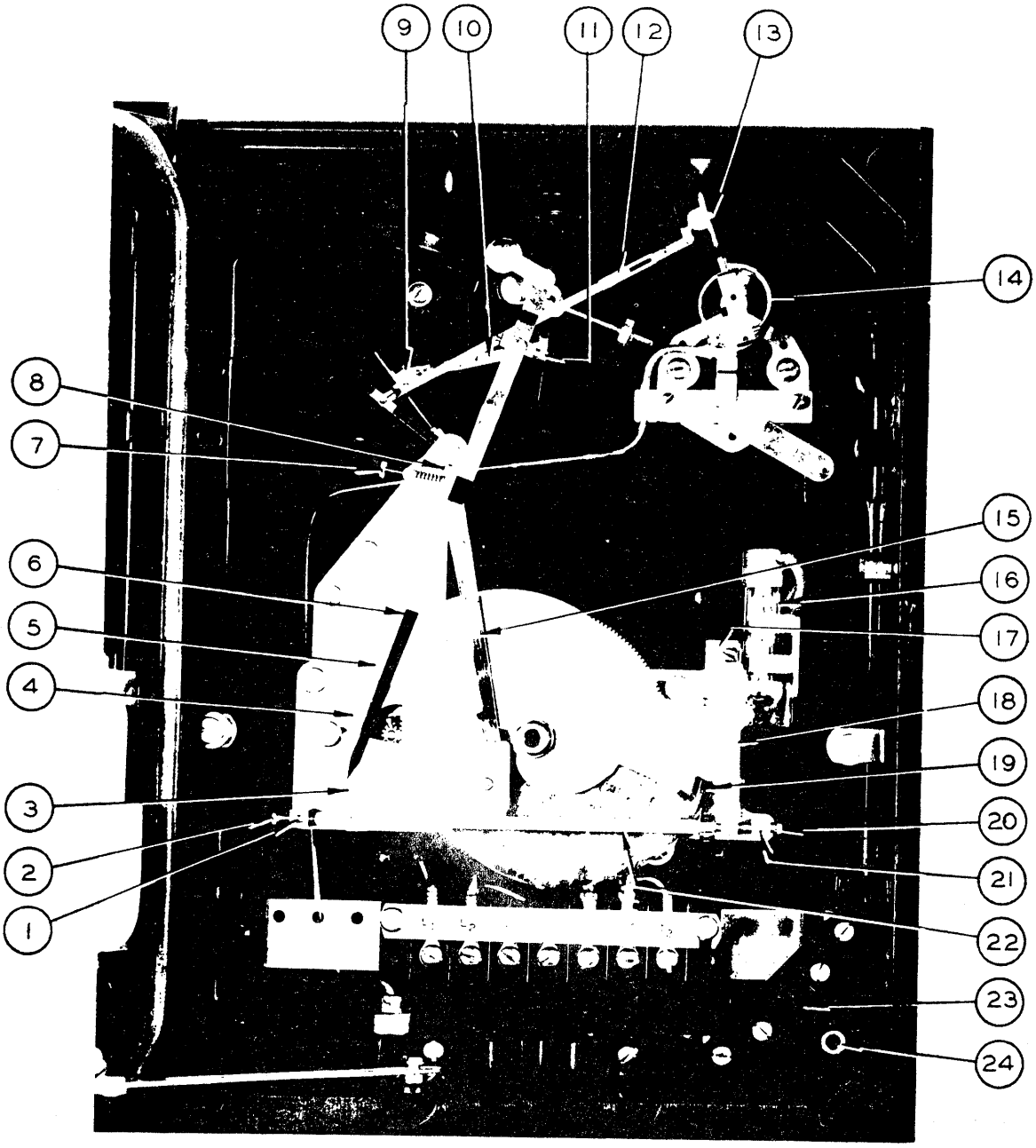


FIGURE 7. TYPICAL [redacted] TRANSMITTER WITH TIMER PLUG ATTACHMENT R14A AND SCALE PLATE REMOVED

CODE TO FIGURE 7

1. Locking Screw
2. Bearing Screw
3. Lifter Plate
4. Fixed Arm
5. Cam
6. Indicating Pointer
7. Zero Adjustment for Cam Follower
8. Locking Screw
9. Span Adjustment for Indicating Pointer
10. Linearity Adjustment for Pointer
11. Zero Adjustment for Indicating Pointer
12. Linearity Adjustment for Cam Follower
13. Span Adjustment for Cam Follower
14. Measuring Element
15. Cam Follower
16. Mercury Switch
17. Permanent Magnet
18. Vane-Arm Assembly
19. Balance Cam
20. Bearing Screw
21. Locking Screw
22. Trip-Plate Shaft
23. Convenient Outlet } Attachment R14A
24. Timer Plug } Optional

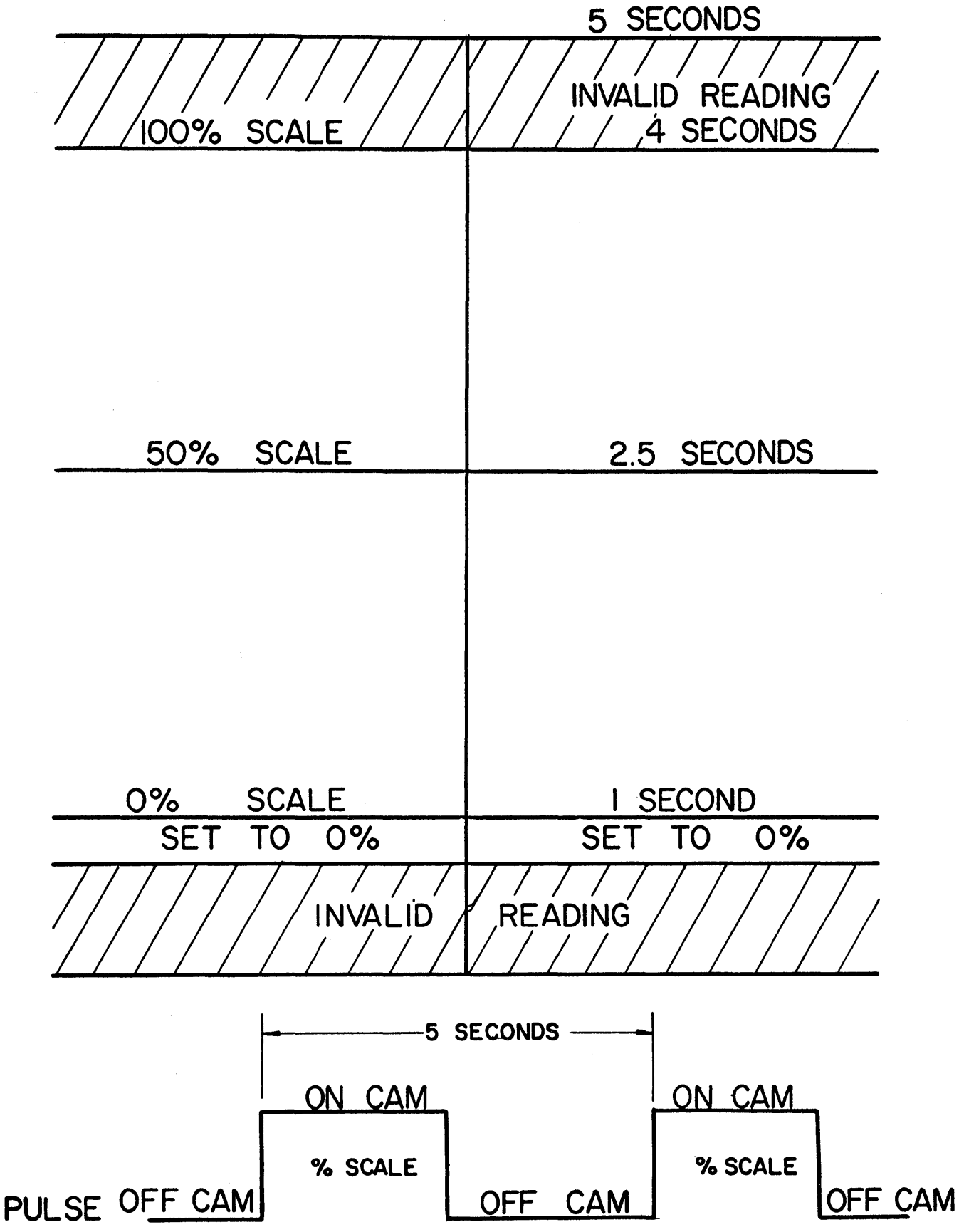
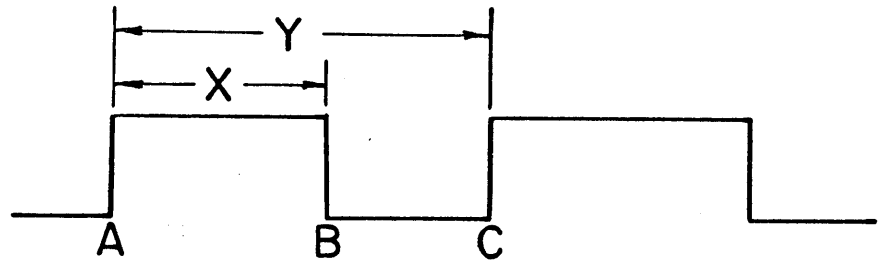

5 SECONDS
100% SCALE
INVALID READING
4 SECONDS
50% SCALE
2.5 SECONDS
0% SCALE
SET TO 0%
1 SECOND
SET TO 0%
INVALID
READING
5 SECONDS
ON CAM
% SCALE
OFF CAM
ON CAM
% SCALE
OFF CAM
PULSE OFF CAM

FIGURE 8

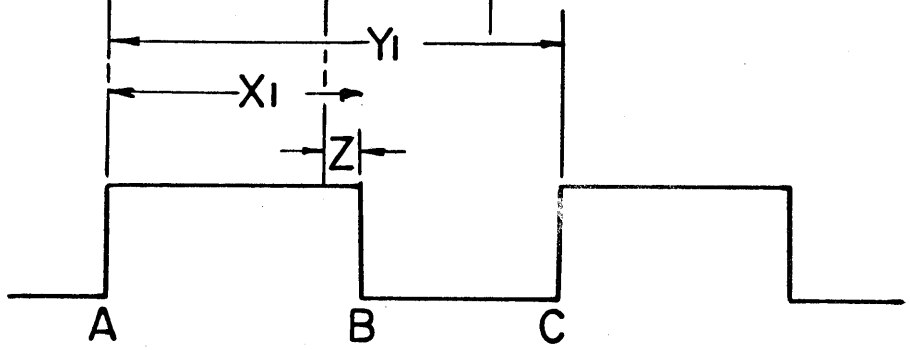
A

1000 P.S.I. CHART
PRESSURE=500 P.S.I.
CAM BASE=5000



Y. TELEMETERED CAM TIME=5000
X = TELEMETERED ELAPSED TIME=2500
A = CIRCUIT CLOSED (TIME ON 1)
B = CIRCUIT OPEN (TIME OFF)
C = CIRCUIT CLOSED (TIME ON 2)

B



Y_1 = TELEMETERED CAM TIME = 5500
X_1 = TELEMETERED ELAPSED TIME = 2750
Z = ERROR IN ELAPSED TIME = 250
Fx = CORRECTED ELAPSED TIME

$$\frac{Fx}{5000} = \frac{X_1}{Y_1} \qquad \frac{Fx}{5000} = \frac{2750}{5500}$$

Fx = 2500

FIGURE 9

one of the hardware timers. This time is kept in an array for the input as Time On. The next sequence would be the loss of this signal, or Time Off. Again the computer is interrupted at point (B) and a program records the time of the interrupt. By subtracting the Time On from Time Off an elapsed time is obtained and placed in an array as elapsed time. The final sequence for this particular input would be the second Time On or point (C). Again the program records the Time On as before, but now a new value is available, the total time required to send this signal. In telemetering systems, such as the one used by Public Service, this value can be used as a correction factor. A transmitter sends one signal every 5 seconds or 5000 ms. In cold weather, the cam speed can slow down causing the total time required to send a value to increase. This would also cause the elapsed time to be longer, thus creating an error in the value sent. The total time required for the cam in the transmitter to make a complete revolution is 5000 ms. If the total time that was actually sent is recorded, a simple ratio can adjust the elapsed time to negate any error caused by the cam either being too slow or too fast. See Figure 9, Part B.

The advantage of using this type of input system is that the computer is freed from scanning. Instead of scanning the inputs to determine a status change, the input interrupts the computer only when the change has occurred. In the Public Service Company application 160 values were input to the computer at an estimated overhead of only 10% of the computer's time. This can be compared to 100% of the computer time using digital input hardware.

All of the input variables are output on logs to technicians so that they can calibrate the transmitter to the computer.

Ten P. I. S. W. words were wired to level zero of the computer, these are used for all Time On interrupts. Ten P. I. S. W. words were wired to level one of the computer; these are used for all Time Off interrupts. One signal line from a transmitter is fed into a type "C" relay creating an On and Off interrupt to the computer from the transmitter.

2.2 DATA CHECKING

Every 10, 15 or 30 seconds, dependent upon the desired operation set by the operator, a program will interrogate the data placed in the computer by the input program. This program will check each input point

for any of nine error conditions. If an error is found in the data, a report will be made to the operator.

Error Conditions:

1. Off Scale - Below 0% or over 100% of scale.
2. Back on Scale - Was Off Scale - now reading is valid.
3. Rate of change is over limits - The present value compared to the last value is increasing or decreasing too rapidly.
4. Out of Limits - Value is out of limit bounds that are pre-set.
5. Back in Limits - Value is now in pre-set bounds.
6. Cam Speed Out of Limits - Cam in transmitter is either too slow or too fast on each 360 degree revolution.
7. Cam Speed back in Limits - Cam in transmitter is now operating satisfactorily.
8. Circuit Out of Service - Input line is out.
9. Circuit Back in Service - Input line is restored.

An array of last good readings is maintained by this program and if an input is found off scale, its last good reading will not be changed. This gives the computer an estimating capability in the event that an input circuit is out of service.

2.3 DATA PROCESSING

Two types of inputs are processed by the 1800 computer, pressure and meter stations. Pressures are values on the system that are used for Control. A file is maintained for each pressure input and includes the following:

1. Instant High - highest value telemetered.
2. Instant Low - lowest value telemetered.
3. Hour Average - last hour's average reading.
4. Day Average - 24 hour average reading.
5. Scale Range - Range of input.
6. Off Scale - Number of times reading was off scale for the day.

Meter Station values include static pressure, differential pressure and flowing gas temperature. See Figure 5. A file is maintained for each meter station and includes the following:

1. Instant high flow - highest flow calculated.
2. Instant low flow - lowest flow calculated.
3. Hour flow - Hour flow calculated for station.
4. Day flow - Flow calculated for the day.
5. Year High - Largest hour flow for the year (Peak Hour)
6. Projected Hour - If the flow rate for this station remains constant, projected rate will be the hour load.
7. Projected Day - If the flow rate for this station remains constant, projected rate will be the day load.

Constants needed for information and calculation, such as orifice meter plate size, are also included in each statistical file.

2.4 OPERATOR INQUIRY AND ENTRY

Data such as gravity and plate size, needed for flow calculation, is input to the computer through the 1816 Keyboard typewriter. A program called "Key Board Monitor" enables the operator to queue up a function program from the keyboard by typing in the name for that program. Once the program has started execution, instructions such as the type and format of the data are typed out to the operator. The operator follows the instructions and enters the data the computer is expecting. This minimizes data errors and leaves typewritten copy on what data was entered, by whom and when. A sample of the input of a plate change for a meter station is shown in Figure 10.

The operator can call out any variable in the same manner, by typing in the program name for the function desired.

This type of output has greatly simplified the training of the operators.

2.5 GAS LOAD FORECASTING

Each year statistics on the last 26 years of weather and gas loads are applied against a model. From this data, a linear equation is developed $y = a - bx$, where y = forcasted load; a = y intercept; b = slope or forecast load per degree (F) change in mean temperature, and x is the expected mean (F) temperature for the day. An equation is developed for each hour for the remaining hours of the day for all four companies along with industrial equations for the industrial loads. These equations are solved in the 1800 computer each hour and a report is typed out to the operator showing the forecasted load that is equal to the forecast for the remaining hours of the day plus the amount already purchased

FIGURE 10

```
07:39:20
PLATE CHANGE
   PLEASE TYPE YOUR 3 INITIALS, THEN DEPRESS -EOF-KEY
EEM
   INPUT IS BY  EARL E. MC LAUGHLIN  METHODS DEPT
   PLEASE INPUT THE MNEMONIC FOR THE STATION YOU WISH TO CHANGE

CHCK
 CHERRY CREEK
                                RUN =  1  HAS A  3.00  INCH PLATE NOW
DOYOU WISH TO CHANGE THE PLATE SIZE IN THIS RUN


   SWITCH TO ALPHA AND TYPE YES OR NO
YES

 PLEASE INPUT THE NEW PLATE SIZE IN 4 DIGITS IN THIS FORMAT XX.XX
04.25
 NEW PLATE SIZE ENTERED   4.25  IF CORRECT TYPE YES,IF NOT TYPE NO

   SWITCH TO ALPHA AND TYPE YES OR NO


YES
                                RUN =  2  HAS A  3.77  INCH PLATE NOW
DOYOU WISH TO CHANGE THE PLATE SIZE IN THIS RUN


   SWITCH TO ALPHA AND TYPE YES OR NO


NO
                                RUN =  3  HAS A  3.77  INCH PLATE NOW
DOYOU WISH TO CHANGE THE PLATE SIZE IN THIS RUN


   SWITCH TO ALPHA AND TYPE YES OR NO

NO
    7:42: 4  KEYBOARD INPUT COMPLETED
```

OPERATORS TYPING IS UNDERLINED – ALL OTHER IS COMPUTER OUTPUT

for all four companies.

The output will also list the storage requirements and industrial curtailment needed to maintain gas purchase contracts. At the end of the day, a full report is made to the operator comparing the actual loads to the forecasted loads.

2.6 DATA LOGGING

System logs are typed out each hour showing flow totals, pressure readings, load forecasts and load projections for all systems controlled by the center. The format of the logs shows all of the variables that the computer used in calculating the flow rates. The logs are held pending for one hour to allow the operator time to correct the log if he disagrees with any of the computer data. If a correction is made, an amended log is typed out showing the computer's figure and the operator's correction. The only logs in use by the control center are the computer's output.

System logs in a schematic format are also output to the operator. These logs show the pressures and flow rates on the system during peak conditions, and are a valuable tool for system analysis.

At the end of each day, logs are output showing a summary of the center's operations for that day.

2.7 CONTROL

The intent of the control programs is to enable the 1800 computer to control the gas regulator stations delivery pressures under normal operating conditions. No provisions were made in the programs to handle abnormal conditions such as electric outages, line breaks or equipment failures.

The operators on duty should have control over the system pressures in the advent of any abnormalities, since their experience would be needed to make the correct decision as to what pressures need adjustment.

Eight low pressure stations were put on computer control. Each station has a tail end pressure. This tail end pressure is located in the stations service area at the lowest pressure point. A minimum pressure limit is held at this tail end point (usually 3 Psig) by raising or lowering the station's output pressure. Each individual station has its own operating characteristics, due to the size of the area served and

the type of load in that area. The eight low pressure stations were chosen to provide statistics for next year's planning. An expansion of the computer system is under study that would enable the computer to control all of the low pressure and intermediate pressure systems.

Both the delivery and tail end pressures are input to the computer to give the programs the information needed to take control of the station. Using this information and a target set pressure at the tail end, five decisions bands are set up to determine what action is needed. See Figure 11.

1. Target Pressure + .250 Psig = Band one.
   If the tail end pressure is above this band, the computer will issue a lower command that is one half of a second in length to the delivery station, providing the end pressure is below the delivery pressure. One-half second is equivalent to approximately .25 Psig at the station.
2. Target Pressure -.250 Psig = Band two.
   If the tail end pressure is above this band and below band one, the computer will take no action.
3. Target Pressure -.30 Psig = Band three.
   If the tail end pressure is above this band and below band two, the computer will issue a raise command that is one half of a second in length or approximately .25 Psig at the station.
4. Target Pressure -.80 Psig= Band four.
   If the tail end pressure is above this band and below band three, the computer will issue a raise command that is one and a half seconds in length or approximately .75 Psig at the station.
5. Target Pressure -.80 Psig = Band five.
   If the tail end pressure is below band four, the computer will issue a raise command that is two and one half seconds in length. If the tail end pressure is in band five, this could indicate a program or equipment failure as the pressure should not be permitted to enter this area. In addition to the raise command, an error message will be printed on the 1816 typewriter.

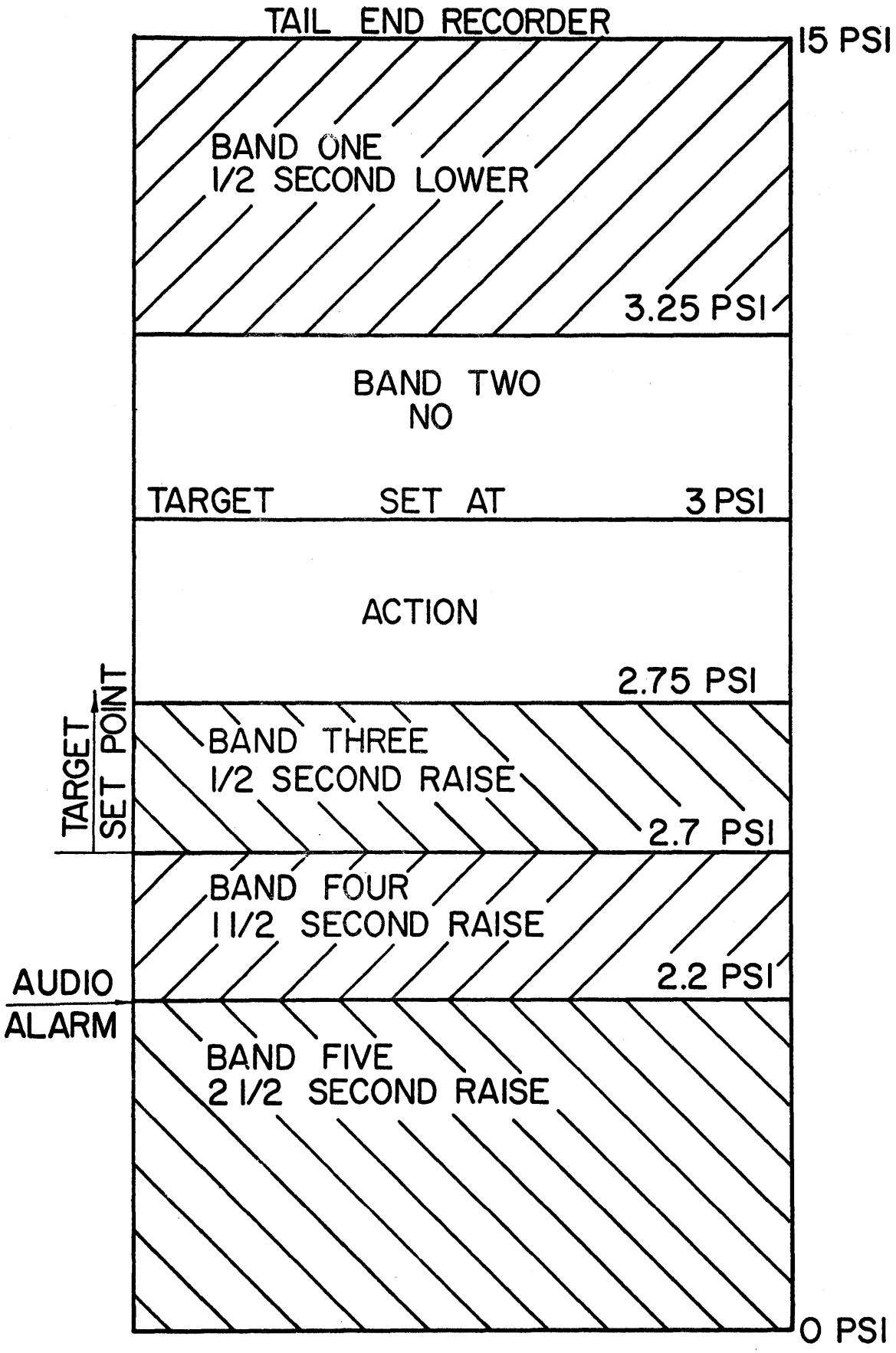

TAIL END RECORDER
15 PSI
BAND ONE
1/2 SECOND LOWER
3.25 PSI
BAND TWO
NO
TARGET SET AT 3 PSI
ACTION
2.75 PSI
TARGET SET POINT
BAND THREE
1/2 SECOND RAISE
2.7 PSI
BAND FOUR
1 1/2 SECOND RAISE
2.2 PSI
AUDIO ALARM
BAND FIVE
2 1/2 SECOND RAISE
0 PSI

FIGURE 11

3.0 PHYSICAL PLANNING

Eight month's prior to the expected delivery of the 1800 computer system, engineering and programming efforts were initiated. Remodeling of the present facilities along with designing and installing the input system were to be completed one month prior to shipment. The programming was done in four distinct phases with a test on each phase at the IBM Test Center at San Jose, California. At the Test Center, experienced IBM personnel assisted Public Service in testing of its system. On the fourth and last test, the complete system was tested as a unit. Upon the completion of the test , all of the operating programs were stored on the disk packs. When the Public Service 1800 computer system was delivered in August of 1967, these same disk packs were put on the system and Public Service was on Line.

4.0 FUTURE PLANNING

At the start of the 1800 computer project, Public Service initiated a two phase program. Phase one would be to input 160 variables from the field to the computer, log hourly calculations to the operators, provide operation logs for the next year's statistical work and test computer control of a regulator station. At the completion of this phase, the computer is doing much more for the control center than was first planned. Eight stations are being controlled in the field instead of one test station. Engineering logs are being output showing the gas system's operation during peak periods. Operator entry has expanded to give the operator access to all of the computer's data, using a method that does not require the operator to be a computer expert.

With the successful completion of Phase one, planning is now under way to initiate phase two. Phase two would expand the computer's input capability from 160 inputs up to 256 inputs. This expansion would allow all of the center's values to be in the computer. Output would expand to enable the computer to control all of the low pressure stations along with the intermediate pressure stations. This would require 100 output points. Both core and disk storage need to be increased to allow room for additional programs.

5.0 COMPUTER OUTPUT

The following four pages are output logs from the computer.

Page one is the hourly system log showing all of the variables and contracts for the four companies controlled by the Center.

Page two and three are engineering logs that were logged during the systems peak period. These logs show flow rates and instant pressures in a schematic format.

Page four consists of system error reports to the operator, gas load forecast and gas load projection called out by the operator.

GAS LOAD CONTROL DENVER HOURLY LOADS, AND CONTRACT DEMAND SYSTEMS OPERATION

MCF AT 14.65 PSIA

CONTRACTS- DENVER G-1(375000.) WSGE P-1(139000.) CHEYENNE P-1(46500.) PUEBLO G-1(53500.)
BEST DECISION- DENVER G-1(372300.) WSGE P-1(139000.) CHEYENNE P-1(46900.) PUEBLO G-1(55100.)

TOTAL HOUR LOADS

	DEN G1	WSEP1	CHYP1	PBLG1	LEYIN	LEYOUT	C + P	DENDIV	NCOLO	DENTOT	PWPTS	FOOT	MAPE	MAPW
(CORR)	18704.	6147.	1400.	2219.	0.	0.	872.	19576.	5275.	20447.	1743.	0.	1469.	0.
COMP	18704.	6147.	1400.	2219.	0.	0.	872.	19576.	5275.	20447.	1743.			
DIFF	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.			

ACCUMULATED LOADS FROM MIDNIGHT

(CORR)	128438.	50531.	8617.	13931.	11633.	0.	14662.	131467.	35869.	148781.	20343.	79.11952.		16.
COMP	128438.	50531.	8617.	13931.	7603.	0.	14662.	135497.	35869.	148781.	20343.			
DIFF	0.	0.	0.	0.	-4030.	0.	0.	4030.	0.	0.	0.			

AVERAGE PER HOUR TO PEAK

								MESA HR	NORF HR	GRLY HR	NORF PR.
	15241.	5529.	2392.	2573.	0.	0.	0.	4278.	282.	1587.	696.5

DENVER HOURLY STATION DATA NUMBER OF CALCULATIONS 13.

	(GRAV)	STAT. PRESS	GAS TEMP	AVG H1	AVG H2	AVG H3	AVG H4	AVG H5	H1MCF	H2MCF	H3MCF	H4MCF	H5MCF	COMPUTER HOUR	COMPUTER ACCUM	(CORRECTED) HOUR	(CORRECTED) ACCUM	ACCUM DIFF
HAMP	0.656	147.	39.5	22.5	26.0	23.6	24.1	23.8	1972.	2118.	2022.	2043.	2027.	12207.	84565.	12207.	84565.	0.
				23.7					2023.									
CHCK	0.666	150.	60.0	0.0	0.0	0.0			0.	0.	0.			0.	0.	0.	0.	0.
NTB	0.656	142.	43.0	25.5	21.9	21.2	19.8		1825.	1691.	1665.	1608.		6791.	53836.	6791.	53836.	0.
ARSE	0.656	238.	44.0	10.8					54.	123.				177.	1489.	177.	1489.	0.
SIXT	0.656	145.	34.0	20.4					403.	403.	403.			1272.	8891.	1272.	8891.	0.
ARAP	0.656	102.	51.0	32.2	0.0	0.0	17.2		836.	0.	0.	0.		836.	6353.	836.	6353.	0.
APIL	0.656	102.	50.0	0.0					0.					0.	22.	0.	22.	0.
														836.	6375.	836.	6375.	0.
CHER	0.656	140.	47.0	0.4	0.0	0.0			38.	0.	0.			38.	7567.	38.	7567.	0.
CPIL	0.656	45.	42.0	56.8					130.					130.	718.	130.	718.	0.
														168.	8285.	168.	8285.	0.
ZUNI	0.656	66.	56.0	42.8	0.0	17.3			712.	0.	0.			712.	5467.	712.	5467.	0.
ZPIL	0.656	21.	57.0	17.9					27.					27.	216.	27.	216.	0.
														739.	5683.	739.	5683.	0.
LEYI	0.683	233.	29.0	0.0	0.0				0.	0.				0.	7603.	0.	11633.	-4030.
LEYO	0.683	233.	36.0	0.0	0.0				0.	0.				0.	0.	0.	0.	0.
CARR	0.688	486.	57.0	13.5					472.	400.	0.			872.	14655.	872.	14655.	0.
PLAT	0.688	158.	46.0	0.0					0.	0.	0.			0.	7.	0.	7.	0.

(ESTIMATED MEAN TEMPS. REMAINDER OF DAY) DEN/NO COLO 41. CHEY 34. PUEBLO 41.

LOAD FORECAST	*MMCF AT 14.65 PSIA* EST FIRM	EST IND	LOG ACCUM	-EST CURT	+ OR - (SY.X)	TOTAL	TOTAL LOAD MINUS BEST DECISION	EST CARR+ PLAT	CARR+ PLAT ACCUM	CIG EST LOAD	CIG ACCUM LOAD
DENVER DIVISION	163.2	45.4	131.4	0.0	12.0	352.1				398.8	189.8
DENVER G-1			128.4			349.0	-20.1				
NORTHERN COLO	41.3	17.5	35.8	0.0	8.0	102.8		21.4	14.6		
WEST SLOPE P-1			50.5			117.5	-21.4				
CHEYENNE	22.5	3.9	8.6	0.0	-85.0	-49.8	-96.7				
PUEBLO	18.2	6.6	13.9	0.0	-45.0	-6.2	-61.3				

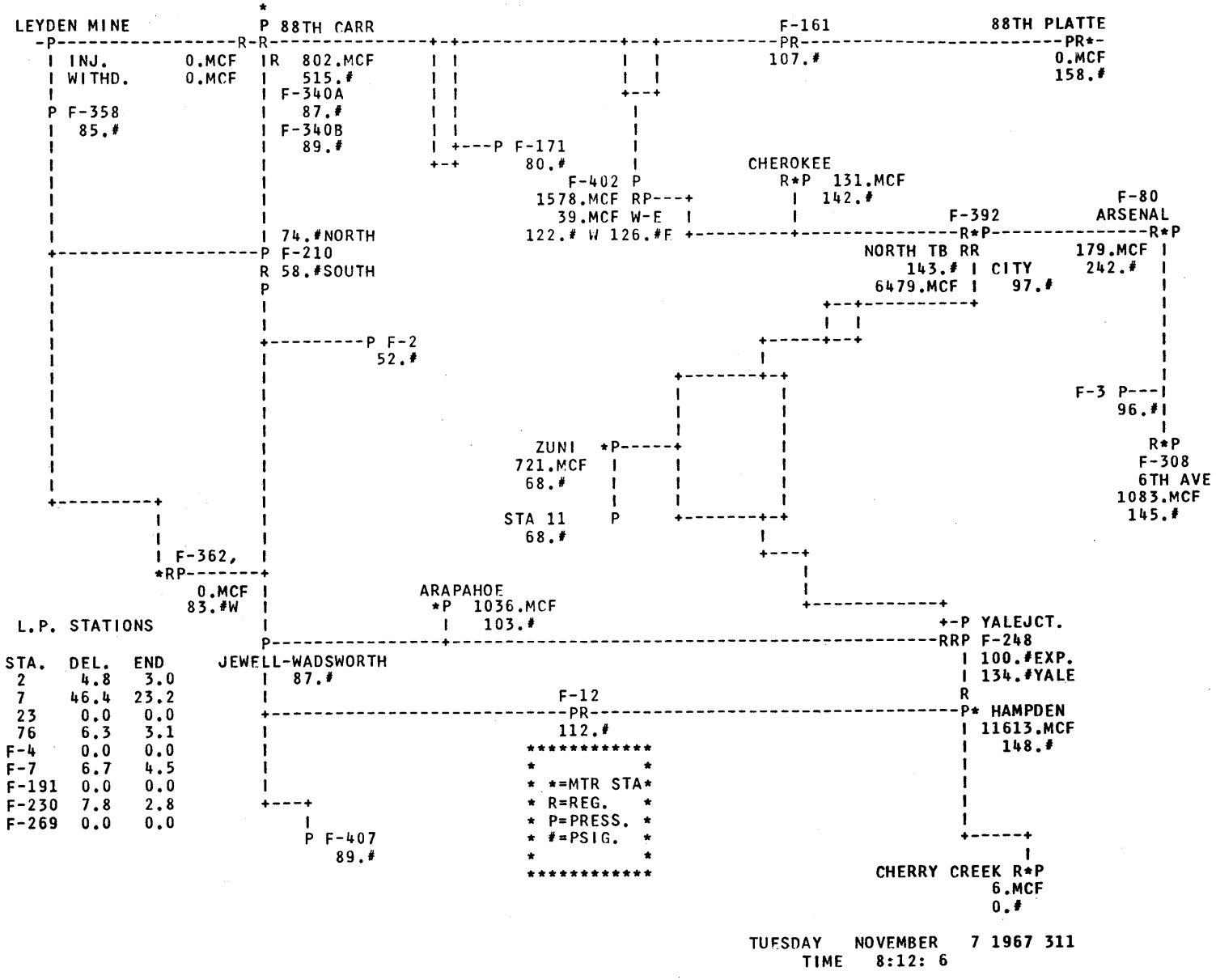
LIPAN AIR TEMP 35.0 CHEYENNE AIR TEMP 36.6 PUEBLO AIR TEMP 24.9 FT. COLLINS AIR TEMP 30.2

( )-MANUAL INPUT

402

OPERATIONS ENGINEERING REPORT
TOTAL DENVER LOAD AT 14.65 PSIA BASE

G-1 19354. + PLANTS 1888. + LEYOUT 0. - LEYIN 0. CARR-PLATTE 802. TOTAL 22044.MCF
PS-1 GRANTED 0. IS-2 GRANTED 0.
LIPAN AIR TEMP = 36.6 DEG.F. COINCIDENCE =14.0%
TOTAL DENVER CURTAILABLE LOAD (41.0) DEG. MEAN TEMP. = 0.MCF
TOTAL DENVER CURTAILMENT LOAD (41.0) DEG. MEAN TEMP. = 10182.MCF
F-S = 0. E-S = 0. D-S = 0. 24G-S = 0. C-S = 0. TOTAL = 10182.MCF



L.P. STATIONS

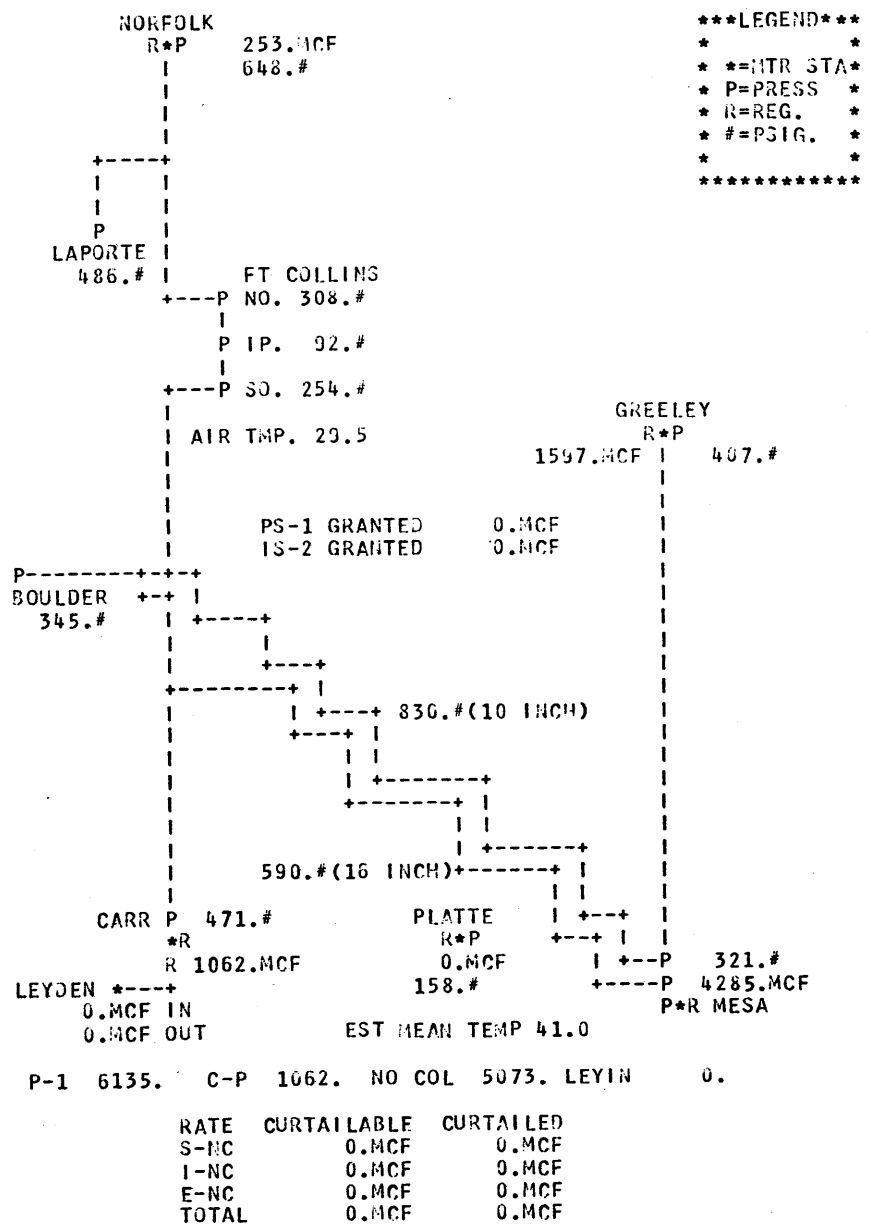
STA.	DEL.	END
2	4.8	3.0
7	46.4	23.2
23	0.0	0.0
76	6.3	3.1
F-4	0.0	0.0
F-7	6.7	4.5
F-191	0.0	0.0
F-230	7.8	2.8
F-269	0.0	0.0

TUESDAY NOVEMBER 7 1967 311
TIME 8:12: 6

403

OPERATIONS ENGINEERING REPORT
LOADS AT 14.65 PSIA.

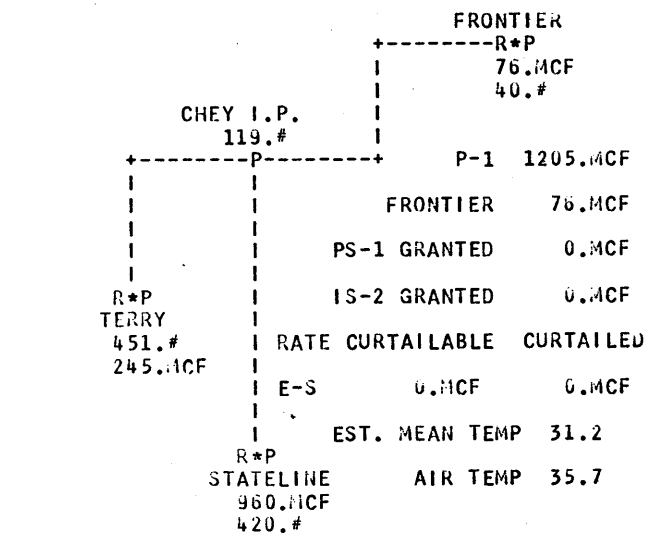
WESTERN SLOPE GAS EASTERN DIVISION



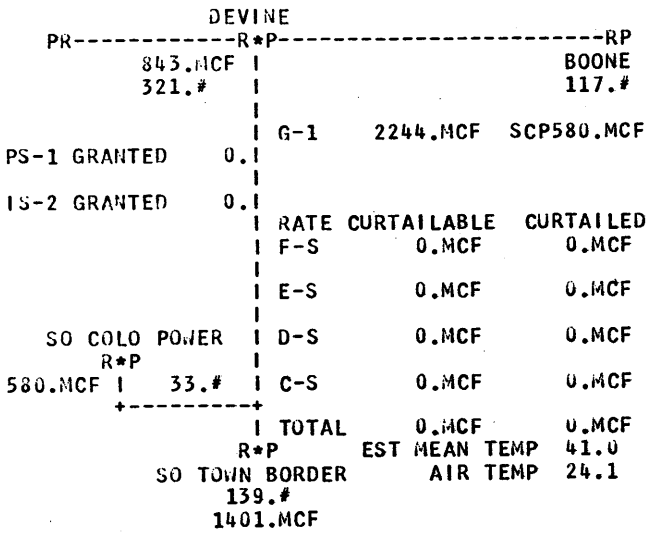
P-1 6135. C-P 1062. NO COL 5073. LEYIN 0.

RATE	CURTAILABLE	CURTAILED
S-NC	0.MCF	0.MCF
I-NC	0.MCF	0.MCF
E-NC	0.MCF	0.MCF
TOTAL	0.MCF	0.MCF

CHEYENNE LIGHT FUEL POWER CO.



PUEBLO GAS AND FUEL CO.



( ) MANUAL HOURLY ENTRY

TUESDAY NOVEMBER 7 1967 311
TIME 7:32:37

404

TIME IS 7:43: 2 ALAMOSA HIGH PRESSURE OUT OF LIMITS 160.8
TIME IS 7:43: 7 CHEYENNE INTERMEDIATE PRESSURE OUT OF LIMITS 124.1
TIME IS 7:43:16 F -210 INTERMEDIATE PRESSURE NORTH NOW BACK IN LIMITS 69.7

(ESTIMATED MEAN TEMPS. REMAINDER OF DAY) DEN/NO COLO 41. CHEY 34. PUEBLO 41.

LOAD FORECAST *MMCF AT 14.65 PSIA*

	EST FIRM	EST IND	LOG ACCUM	-EST CURT	+ OR - (SY.X)	TOTAL	TOTAL LOAD MINUS BEST DECISION	EST CARR+ PLAT	CARR+ PLAT ACCUM	CIG EST LOAD	CIG ACCUM LOAD
DENVER DIVISION	176.3	48.2	111.8	0.0	10.0	346.4				507.7	161.4
DENVER G-1			109.7			344.2	-25.8				
NORTHERN COLO	44.0	18.6	30.5	0.0	10.0	103.2		21.9	13.7		
WEST SLOPE P-1			44.3			117.0	-21.9				
CHEYENNE	24.2	4.1	7.2	0.0	-12.0	23.6	-23.2				
PUEBLO	19.7	7.0	11.7	0.0	-4.0	34.4	-20.6				

LIPAN AIR TEMP 28.9 CHEYENNE AIR TEMP 31.2 PUEBLO AIR TEMP 13.3 FT. COLLINS AIR TEMP 23.6

LOAD PROJECTION AT 7:44:42 MST

DENVER G-1	CARR	PLATTE	WSGE	LEYDEN IN	LEYDEN OUT	CHEYENNE	FRONTIER	PUEBLO	SCP
HOUR PROJECTION									
18997.	1118.	0.	6141.	0.	0.	1316.	75.	2250.	580.
DAY PROJECTION									
425291.	34421.	7.	147725.	7603.	0.	28149.	1783.	49802.	13942.

TIME IS 7:46: 3 ALAMOSA HIGH PRESSURE OUT OF LIMITS 158.4
TIME IS 7:46: 4 ANTONITA HIGH PRESSURE OUT OF LIMITS 169.5
TIME IS 7:46: 9 CHEYENNE INTERMEDIATE PRESSURE NOW BACK IN LIMITS 108.6
TIME IS 7:46:21 CHEYENNE STATIC PRESSURE OUT OF LIMITS 420.0

TIME IS 7:46:32 NORFOLK GAS TEMPERATURE RATE OF CHG IS OVER LIMITS
TIME IS 7:46:34 TERRY ROAD STATIC PRESSURE OFF SCALE
TIME IS 7:46:41 NORFOLK GAS TEMPERATURE NOW ON SCALE
TIME IS 7:46:41 TERRY ROAD STATIC PRESSURE NOW ON SCALE
TIME IS 7:46:44 CHEYENNE NUMBER ONE DIFFERENTIAL CIRCUIT OUT OF SERVICE

T.2.7 (a)

THE CATALYTIC CONSTRUCTION CO.
1528 Walnut St. Phila. Pa. 19102

A Program for Making a Non-Linear Regression Analysis with up to Three Independent Variables

By T.E. Bridge

This program will be very useful for curve fitting. It will develop coefficients for calculating a dependent variable (or function) when up to three independent variables (arguments) are specified. There are nine options available involving various combinations of polynomial, logarithmic, or reciprocal relationships between the variables. It was designed for system 360 Model 30 - 32K with 2 disk drives.

This program will also be useful if you need to correlate a large mass of test or operating data.

Four numbers and a title are read by the machine on the first card (4I2,18A4) N1, N2, N3, N4, title:

N1 specifies the degree of the fit for the first argument, N2 for the second, and N3 for the third:

N	Type of Fit
0	Is not permitted.
1	Calls for a constant or average value for the function not influenced by the argument.
2	Calls for a linear fit. The machine will find a straight line that gives a minimum root mean square error.
3	Calls for a parabolic fit. The machine will find a quadratic equation to give a minimum RMS error.
4	The machine will find a cubic equation to give a minimum RMS error.

Any positive whole number or combination of numbers may be entered for N1, N2, or N3, so long as their product does not exceed 100.

You must enter a total number of data cards greater than the product of N1 x N2 x N3. The more the better.

N4 may be any digit 1 through 9. It specifies the option that is desired -- or whether you want to try for a smooth fit on a graph with linear or logrithmic scales, or a semilogrithmic graph, or even on a graph with a reciprocal scale.

Let Y represent the function, and let X1, X2, and X3 represent three arguments fitted to a degree specified by N1, N2, and N3 respectively. Then, the following table will, for each option specified by N4, show the type of graph that is assumed by the program.

N4	Polynomial	Logrithmic	Reciprocal
1	Y,X1,X2,X3		
2	X1,X2,X3	Y	
3	X2,X3	Y,X1	
4	X3	Y,X1,X2	
5		Y,X1,X2,X3	
6	Y,X2,X3	X1	
7	Y,X3	X1,X2	
8	Y	X1,X2,X3	
9	X1,X2,X3		Y

Enter any desired symbol in columns 76-80 of the title card. The correlation coefficients (the answers) will be punched out by the machine ready for insertion in a Fortran program deck. Each coefficient will be indentified by the symbol entered in columns 76-80 of the title card. It is then a simple matter to write a Fortran program to quickly calculate the function from any given set of the arguments. A subroutine (TB$PLY) should be put in your relocatable library for doing this.

```
CALL TB$PLY(N1,N2,N3,N4,X1,X2,X3,Y,C)
```

C in the above call statement must be replaced by whatever symbol you put in columns 76-80 of the title card. Y is the answer or function, and X1, X2, X3 are arguments corresponding to the fit parameters N1, N2, and N3 respectively.

Data follows immediately behind the title card. Each card (or row on the data sheet) will include four numbers in order -- Y,X1,X2,X3. (4F10.3,I5)

The problem is terminated if a digit is read in columns 41-45. If a zero (or blank) is entered for X1, X2, or X3, the program will enter the preceeding value for that argument. Therefore, if you want to enter a real zero, enter some very small number close to it, as, for example -- .0000001. Of course, if you are making a logrithmic fit, you must not enter a negative value for the variable.

Explanation of the Program

I think that my explanation of this program will be a lot simpler if I talk about a specific formula rather than about a generalized one. Consider the following formula:

$$Y = Ca + Cb\ Xa + Cc\ Xa^2 + Cd\ Xb + Ce\ Xb\ a + Cf\ Xb\ Xa^2$$

In the above equation, we are making a 3 point fit with respect to Xa and a 2 point fit with respect to Xb. That is; if we hold Xb constant, Y will plot as a parabola against Xa; and if we hold Xa constant, Y will plot as a straight line against Xb. We will read from a family of curves a value of Y for each of many different values of Xa and Xb. Our program is to find values for Ca, Cb, Cc, Cd, Ce, and Cf that will best fit the data that we give. Let Yg be the given value of Y for any point corresponding to Xa and Xb. Let E be the sum of the square of all of the errors for N given points.

$$E = \sum (Y-Yg)^2$$

Differentiate with respect to Ca. Since we are looking for a minimum value of E, this derivative may be made equal to zero.

$$dE/dCa = 2.\sum (Y-Yg)dY/dCa = 0.$$

but $dY/dCa = 1$, therefore : $\sum Y = \sum Yg$

$$\sum Ca + \sum Cb\ Xa + \sum Cc\ Xa^2 + \sum Cd\ Xb + \sum Ce\ Xa\ Xb + \sum Cf\ Xa^2\ Xb = \sum Yg$$

Similarly if we differentiate with respect to Cb, we can write another equation:

$$dE/dCb = 2\sum (Y-Yg)\ dY/dCb = 0$$

but $dY/dCb = Xa$, therefore : $\sum Y\ Xa = \sum Yg\ Xa$

$$\sum Ca\ Xa + \sum Cb\ Xa^2 + \sum Cc\ Xa^3 + \sum Cd\ Xb\ Xa + \sum Ce\ Xb\ Xa^2 + \sum Cf\ Xb\ Xa^3 = \sum Yg\ Xa$$

In this way, we can write 6 equations. Lets put them in matrix form. The unknowns are Ca, Cb, Cc, Cd, Ce, and Cf

Note that $\sum(1) = N$

N	$\sum Xa$	$\sum Xa^2$	$\sum Xb$	$\sum XbXa$	$\sum Xb\ Xa^2 = \sum Yg$
$\sum Xa$	$\sum Xa^2$	$\sum Xa^3$	$\sum Xb\ Xa$	$\sum Xb\ Xa^2$	$\sum Xb\ Xa^3 = \sum Xa\ Yg$
$\sum Xa^2$	$\sum Xa^3$	$\sum Xa^4$	$\sum Xb\ Xa^2$	$\sum Xb\ Xa^3$	$\sum Xb\ Xa^4 = \sum Xa^2\ Yg$
$\sum Xb$	$\sum Xb\ Xa$	$\sum Xb\ Xa^2$	$\sum Xb^2$	$\sum Xb^2Xa$	$\sum Xb^2Xa^2 = \sum Xb\ Yg$
$\sum Xb\ Xa$	$\sum Xb\ Xa^2$	$\sum Xb\ Xa^3$	$\sum Xb^2Xa$	$\sum Xb^2Xa^2$	$\sum Xb^2Xa^3 = \sum Xb\ Xa\ Yg$
$\sum Xb\ Xa^2$	$\sum Xb\ Xa^3$	$\sum Xb\ Xa^4$	$\sum Xb^2Xa^2$	$\sum Xb^2Xa^3$	$\sum Xb^2Xa^4 = \sum Xb\ Xa^2\ Yg$

As we read in the data (in an array called X) we will calculate each term in the first equation and put in an array called U.

The data read on each card is stored in file 11 so that it can be re-read after completing the analysis. At that time, we will calculate a value Y and compare it with the given value to see if we have a fit. Any positive number in columns 41-44 of a data card will signal the end of data.

Each column of the matrix is stored as a record in file 13. At the beginning, we have to zero the matrix. This is done in statement 44,

Immediately after we read each card, we must increment every term in the matrix as follows:

```
      Do 9 M=1,NP
      Read(13'M)(T(K),K=1,N)
      Do 10 L=1,N
   10 T(L)=T(L)+U(L)*U(M)
    9 Write(13'M)(T(K),K=1,N)
```

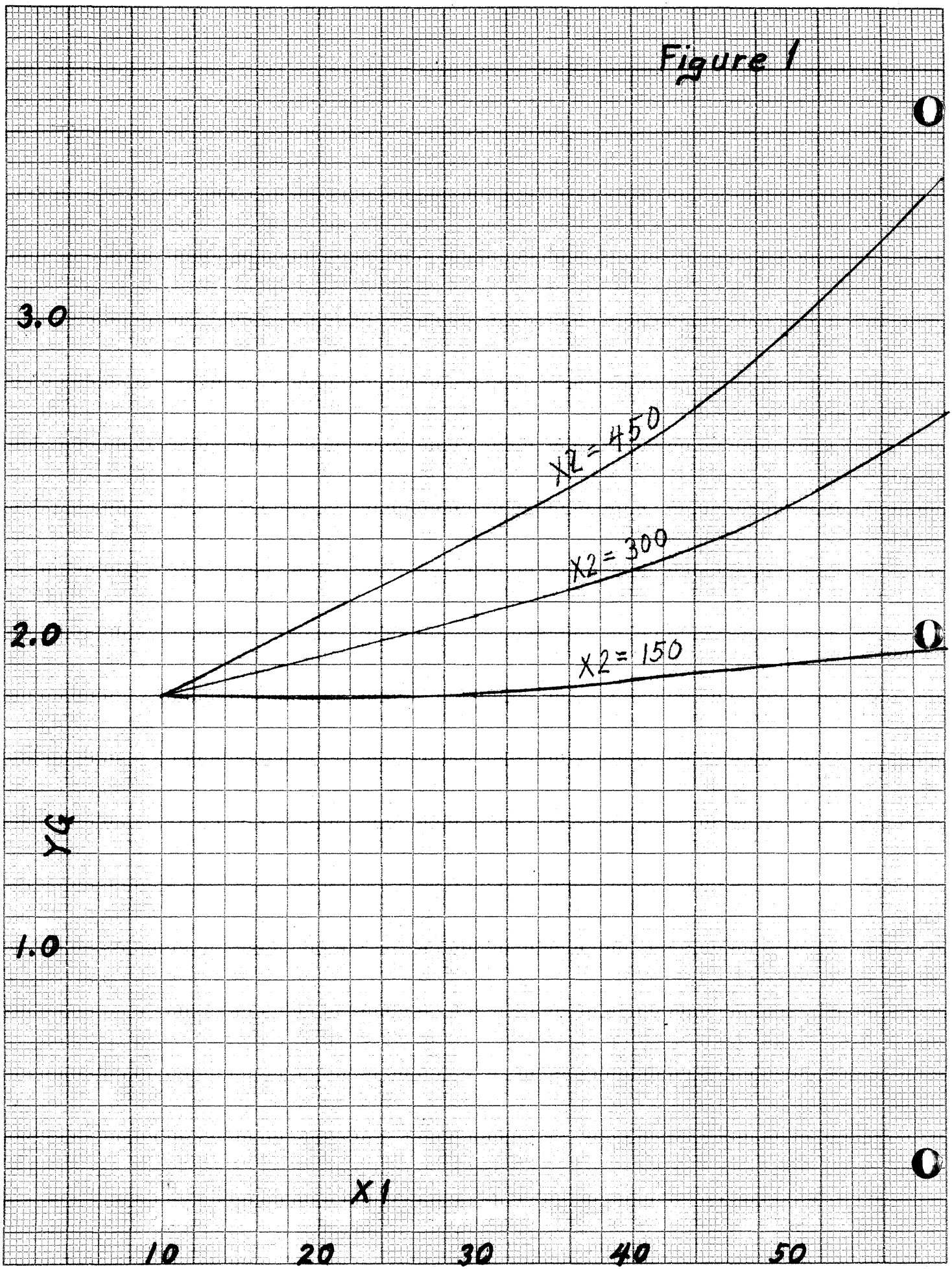
Now, all we have to do, is to solve for the coefficients Ca, Cb, Cc, etc. These answers are punched on cards in a Fortran format so that they can be put right into your Fortran source deck to initialize your array. A symbol name was picked up from columns 77-80 on the title card, You need only write a dimension statement to reserve space for the array.

If you put the subroutine TB$PLY in your relocatable library, you are all set to use it in any fortran program.

Table 1 is a listing of the main program TB$RA and the subroutine TB$PLY that expands the polynomial to fit the curve.

Table 2 is an example of a regression that was made for the curves of figure 1 .

The end of program TB$RA (starting with statement 41) shows how you would use the program. This statement rereads all of the data from file 11 and then makes a comparison between the given and calculated values for Y .



KEUFFEL & ESSER CO. N. Y.

TABLE 1 LISTING OF PROGRAM FOR REGRESSION ANALYSIS

```
// JOB TB$RA                                                          0001
// OPTION CATAL                                                       0002
     PHASE TB$RA01,S                                                 0003
// EXEC FORTRAN                                                       0004
C                                                                     0005
C     MULTIPLE REGRESSION   PROGRAM BY TEB  1967                      0006
C                                                                     0007
C     SAMPLE TITLE CARD ASSUMING  4  TERMS IN EQUATION FOR  X1 , 3 FOR 0008
C     X2 , AND  2  FOR  X1 .                                          0009
C                                                                     0010
C     04030201  TEST PROBLEM  1-23-67  Y  VS  X1 X2 X3                0011
C                                                                     0012
C     THE  04  IS CALLED N1, 03  IS CALLED N2,  02 IS CALLED N3 , AND 01  IS 0013
C     N4  IS A VARIABLE GIVING ONE OF NINE OPTIONS. IF N4 IS          0014
C      1   WE HAVE A STRAIGHT POLYNOMIAL OPTION                      0015
C      2   WE HAVE A SEMI LOG OPTION. LOG OF  Y  ONLY                0016
C      3  WE HAVE  LOG OF Y  AND  X1  ONLY                          0017
C      4  WE HAVE  LINEAR WITH RESPECT TO X3  ONLY, LOG OF OTHER VARIABLES 0018
C      5  WE HAVE LOG OF ALL VARIABLES                              0019
C      6  WE HAVE LOG OF  X1  ONLY                                  0020
C      7  WE HAVE LOG OF  X1 , AND X2  ONLY                         0021
C      8  WE HAVE LOG OF  X1 , X2 , AND X3  ONLY                    0022
C      9  WE HAVE THE RECIPROCAL OF  Y  , AND LINEAR WITH ALL OTHERS 0023
C                                                                     0024
C     SAMPLE DATA CARD. USE FIELDS OF TEN COLUMNS EACH                0027
C                                                                     0028
C     YG         X1         X2         X3                           0029
C     34.1       3.9        .42        4.                           0030
C                                                                     0031
C     IF AN ARGUMENT IS BLANK THE PRECEEDING VALUE WILL BE USED.      0032
C     THEREFORE, DO NOT USE A TRUE ZERO. ANY VERY SMALL NUMBER WILL DO. 0033
C                                                                     0035
C     USE A TAIL CARD WITH A  1  IN COLUMN 41.                        0036
C                                                                     0037
C                                                                     0038
C     WE READ DATA AND BUILD MATRIX                                   0039
C                                                                     0040
      DOUBLE PRECISION  T(100) , D(100) , F, G                        0041
      COMMON N1, N2, N3, N4, N, NP, SYM                               0042
      DIMENSION X(4,2), U(100)                                        0043
      EQUIVALENCE (N123 , N)                                          0044
      DEFINE FILE 11(1200,51,U,IN)  , 13(800,200,U,IN3)               0045
  102 FORMAT ( 1H1, 4I3,2X, 18A4 )                                    0046
  101 FORMAT ( 1HC, 4I3, 2X, 17A4 )                                   0047
  100 FORMAT( 4I2, 18A4 )                                             0048
   33 READ (1,100) N1, N2, N3, N4, (U(K), K = 1,18 )                  0049
      SYM = U(18)                                                     0050
      DO 35  K = 1,8                                                  0051
   35 X(K,1) = 1.                                                     0052
      WRITE (2,101) N1, N2, N3, N4, (U(K) , K = 1,17)                 0053
      WRITE (3,102) N1,N2,N3,N4,(U(K) , K= 1,18)                      0054
      N123 = N1 * N2* N3                                              0055
      NP = N123 & 1
C     ZERO THE MATRIX
```

```
      DO 8 K = 1,N123                                                    0056
    8 T(K) = 0.                                                          0057
      DO 44 K = 1,NP                                                     0058
   44 WRITE (13'K)(  T(L) , L = 1,N )                                    0059
      I = 1                                                              0060
      J= 2                                                               0061
      IN = 1                                                             0062
      GO TO 36                                                           0063
    3 WRITE(11'IN) IT,(X(K,J) ,K = 1,4 )                                 0064
      GO TO (11,12,13,14,15,16,17,18,19), N4                             0065
   19 X(4,J)=1./ X(4,J)                                                  0066
      GO TO 11                                                           0067
   18 X(3,J) = ALOG(X(3,J))                                              0068
   17 X(2,J) = ALOG(X(2,J))                                              0069
   16 X(1,J) = ALOG(X(1,J))                                              0070
      GO TO 11                                                           0071
   15 X(3,J) = ALOG( X(3,J))                                             0072
   14 X(2,J) = ALOG( X(2,J))                                             0073
   13 X(1,J) = ALOG( X(1,J))                                             0074
   12 X(4,J) = ALOG(X(4,J))                                              0075
   11 CONTINUE                                                           0076
      K = 0                                                              0077
CALCULATE THE FIRST ROW IN THE MATRIX  IN U                              0078
      C = 1.                                                             0079
      DO 7 NC= 1,N3                                                      0080
      B = 1.                                                             0081
      DO 6 NB= 1,N2                                                      0082
      A = 1.                                                             0083
      DO 5 NA= 1,N1                                                      0084
      K = K & 1                                                          0085
      U(K) = A*B*C                                                       0086
    5 A = A* X(1,J)                                                      0087
    6 B = B * X(2,J)                                                     0088
    7 C = C * X(3,J)                                                     0089
      U(K&1) = X(4,J)                                                    0090
CALCULATE UPDATED MATRIX                                                 0091
      DO 9 M = 1,NP                                                      0092
      READ( 13'M)( T(K), K = 1,N)                                        0093
      DO 10 L = 1,N                                                      0094
   10 T(L) = T(L) & U(L) * U(M)                                          0095
    9 WRITE (13'M) (T(K) , K = 1,N)                                      0096
      K = I                                                              0097
      I = J                                                              0098
      J = K                                                              0099
   36 READ (1,104)     X(4,J), X(1,J), X(2,J), X(3,J) , IT               0100
  104 FORMAT ( 4F10.3 , I5 )                                             0101
CHECK FOR  ZERO ARGUMENTS                                                0102
      DO 1 K = 1,3                                                       0103
      IF(X(K,J)) 1,2,1                                                   0104
    2 X(K,J) = X(K,1)                                                    0105
    1 CONTINUE                                                           0106
CHECK FOR TAIL CARD                                                      0107
      IF(IT) 3,3,4                                                       0108
    4 WRITE (11'IN) IT                                                   0109
C                                                                        0110
C     THE FOLLOWING IS GIVEN TO HELP YOU UNDERSTAND THIS PROGRAM . CAN YOU 0111
```

```
C     SOLVE  N  SIMULTANEOUS EQUATIONS  WITH ONLY  7  FORTRAN STATEMENTS      0112
C     NP = N& 1                                                               0113
C     DO 1 K = 1,N                                                            0114
C     F = T(K,K)                                                              0115
C     DO 1 J = K,NP                                                           0116
C     T(K,J) = T(K,J) / F                                                     0117
C     DO 1 I = 1 , N                                                          0118
C   1 IF(I.NE. K) T(I,J) = T(I,J) - T(K,J) * T(I,K)                           0119
C     SOLVE THE SIMULTANEOUS EQUATIONS                                        0120
      DO 21 K = 1,N                                                           0121
      READ(13'K) (D(L) , L = 1,N)                                             0122
      F = D(K)                                                                0123
      K1 = K & 1                                                              0124
      DO 21 J = K1 , NP                                                       0125
      READ( 13 ' J) ( T(L) , L = 1,N )                                        0126
      T(K) = T(K) / F                                                         0127
      G = T(K)                                                                0128
      DO 22 I = 1,N                                                           0129
      IF (I-K) 23,22,23                                                       0130
   23 T(I) = T(I) - D(I) * G                                                  0131
   22 CONTINUE                                                                0132
   21 WRITE (13'J) (T(L), L = 1,N  )                                          0133
      DO 24 K = 1,N                                                           0134
   24 U(K) = T(K)                                                             0135
C     PUNCH COEFFICIENTS                                                      0136
  105 FORMAT(7X, A4, 1H( , I4, 5X, 4H)=  , E15.8  )                           0137
      WRITE (2,105) (SYM,K,U(K), K = 1,N)                                     0138
      IN = 1                                                                  0139
  106 FORMAT( 1X, 5E13.5 , F10.3)                                             0140
  107 FORMAT(/6X, 2HX1, 11X, 2HX2, 11X, 2HX3, 11X,2HYG, 11X,1HY,              0141
     1 12X, 5HER PC  / )                                                      0142
      WRITE (3,107 )                                                          0143
      GO TO 41                                                                0144
   32 CALL TB$PLY ( N1, N2, N3, N4, X1, X2, X3, Y,U )                         0145
      ER = 100. * (YG - Y) / Y                                                0146
C     TABULATE RESULTS                                                        0147
      WRITE (3,106) X1, X2, X3,  YG, Y, ER                                    0148
   41 READ( 11 ' IN ) IT,    X1,X2,X3 ,YG                                     0149
      IF (IT) 32,32,33                                                        0150
      END                                                                     0151
/*                                                                            0152
// EXEC FORTRAN                                                               0153
      SUBROUTINE TB$PLY ( N1,N2,N3,N4,Z1, Z2, Z3, F3, C )                     0154
      DIMENSION C(36)                                                         0155
      X1 = Z1                                                                 0156
      X2 = Z2                                                                 0157
      X3 = Z3                                                                 0158
      GO TO (11,11,13,14,15,13,14,15,11), N4                                  0159
   15 X3 = ALOG(Z3)                                                           0160
   14 X2 = ALOG(Z2)                                                           0161
   13 X1 = ALOG(Z1)                                                           0162
   11 CONTINUE                                                                0163
      N = N1 * N2 * N3                                                        0164
      F3 = 0.                                                                 0165
      DO 3 K = 1,N3                                                           0166
      F3 = F3 * X3                                                            0167
```

```
      F2 = 0.                                                          0168
      DO 2 J = I,N2                                                    0169
      F2 = F2 * X2                                                     0170
      F1 = 0.                                                          0171
      DO 1 I = 1,N1                                                    0172
      F1 = F1 * X1 & C(N)                                              0173
    1 N = N - 1                                                        0174
    2 F2 = F2 & F1                                                     0175
    3 F3 = F3 & F2                                                     0176
      GO TO (21,22,22,22,22,21,21,21,29) , N4                          0177
   29 F3 = 1. / F3                                                     0178
      GO TO 21                                                         0179
   22 F3 = EXP( F3)                                                    0180
   21 CONTINUE                                                         0181
      RETURN                                                           0182
      END                                                              0183
/*                                                                     0184
// EXEC LNKEDT                                                         0185
/*                                                                     0186
// EXEC  TB$RA01                                                       0187
04030101  ELCD  VS  DS & DSGN                                          0188
 1.8      10.       150.                                               0189
 1.8      30.                                                          0190
 1.85     40.                                                          0191
 1.9      50.                                                          0192
 1.95     60.                                                          0193
 1.8      10.       300.                                               0194
 2.05     30.                                                          0195
 2.2      40.                                                          0196
 2.4      50.                                                          0197
 2.7      60.                                                          0198
 1.8      10.       450.                                               0199
 2.3      30.                                                          0200
 2.57     40.                                                          0201
 2.95     50.                                                          0202
 3.45     60.                                                          0203
                                                                       0204
/*                                                                     0205

C  4  3  1  1    ELCD  VS  DS & DSGN
         CL  (   1      )=   0.22859659E 01
         CL  (   2      )=  -0.65686345E-01
         CL  (   3      )=   0.19246084E-02
         CL  (   4      )=  -0.18875580E-04
         CL  (   5      )=  -0.31371277E-02
         CL  (   6      )=   0.42393967E-03
         CL  (   7      )=  -0.12535511E-04
         CL  (   8      )=   0.12892923E-06
         CL  (   9      )=   0.30227648E-05
         CL  (  10      )=  -0.43952537E-06
         CL  (  11      )=   0.15635532E-07
         CL  (  12      )=  -0.15259277E-09
```

4 3 1 1 TABLE 2 ELCD VS DS & DSGN CL

X1	X2	X3	YG	Y	ER PC
0.10000E 02	0.15000E 03	0.10000E 01	0.18000E 01	0.18002E 01	-0.011
0.30000E 02	0.15000E 03	0.10000E 01	0.18000E 01	0.18002E 01	-0.009
0.40000E 02	0.15000E 03	0.10000E 01	0.18500E 01	0.18477E 01	0.124
0.50000E 02	0.15000E 03	0.10000E 01	0.19000E 01	0.19032E 01	-0.167
0.60000E 02	0.15000E 03	0.10000E 01	0.19500E 01	0.19488E 01	0.062
0.10000E 02	0.30000E 03	0.10000E 01	0.18000E 01	0.17994E 01	0.031
0.30000E 02	0.30000E 03	0.10000E 01	0.20500E 01	0.20529E 01	-0.144
0.40000E 02	0.30000E 03	0.10000E 01	0.22000E 01	0.21967E 01	0.149
0.50000E 02	0.30000E 03	0.10000E 01	0.24000E 01	0.24004E 01	-0.017
0.60000E 02	0.30000E 03	0.10000E 01	0.27000E 01	0.27004E 01	-0.015
0.10000E 02	0.45000E 03	0.10000E 01	0.18000E 01	0.18004E 01	-0.023
0.30000E 02	0.45000E 03	0.10000E 01	0.23000E 01	0.22962E 01	0.164
0.40000E 02	0.45000E 03	0.10000E 01	0.25700E 01	0.25769E 01	-0.268
0.50000E 02	0.45000E 03	0.10000E 01	0.29500E 01	0.29454E 01	0.157
0.60000E 02	0.45000E 03	0.10000E 01	0.34500E 01	0.34511E 01	-0.031

IJT2191

T.2 7
(6)

USER EXPERIENCE WITH 1130 LP-MOSS

An Abstract

U. S. Reduction Co. has used the 1130 LP-MOSS application package since its first release. Various problems have been encountered during its use; however, useful answers have been obtained. Due to the size of the aluminum alloy blending problem being studied, a second 1130 had to be leased for full dedication to this problem. Early results indicate a reasonable payback may be obtained from this hardware and software combination.

An Outline

1. Company background
2. Early uses of linear programming
3. Problems encountered in implementing 1130 LP-MOSS
4. Results to date and future planning
5. Specific application modifications desired

APPLICATION OF SIMULATION IN CONTROL, DESIGN, AND OPTIMIZATION OF CHEMICAL PROCESSES

M. J. Shah

ABSTRACT

The problem of supervisory control and optimization of a chemical process requires that the control as well as the optimization programs be provided with a mathematical relationship between the dependent and independent variables in the process.

In this presentation we will discuss methods of arriving at this relationship based on the fundamental laws of physics and chemistry. We will make the proper approximations valid for control purposes to obtain the solution of the differential equations, describing this relationship. Use of simulation languages helps in reducing the programming time as well as the number of trials required to attain successful solutions.

An example of methanol plant will be discussed in some detail to illustrate the mathematical and programming techniques to achieve the desired relationships for control and optimization. It will be shown how these differential equations with modification of the objective function when used in the optimization program can lead to optimum design of the plant.

Paper to be presented at the Common User's Meeting, San Francisco, December 11-12, 1967.

INTRODUCTION

In the early applications of digital computers for controlling chemical processes, the work performed by the computers was limited not only by the restricted capability of the computers, but also by the fact that control and optimization techniques had not been achieved by personnel of the chemical plants. Much time and effort was spent in data gathering, alarm scanning and limit checking of important plant variables. Since the interrelation between the plant variables was not too well known, or in many cases not defined mathematically, virtually all of the control work was done by using so-called 'regression models' derived from plant data gathered by the control computer. These regression relationships were generally simple enough that computer control could be achieved without overtaxing the capability of the available digital computers.

More recent digital computers in the process-control field are much more powerful in their computing capabilities. They also provide some very useful ready-made programs for performing the various control tasks in a plant over and above the normal logging, alarming, and scanning functions. These programs will be discussed in subsequent sections.

It is important to realize that a control computer must operate in a continuous mode, just like the chemical process it is controlling. The computer thus must perform all the routine data logging, alarm scanning, and limit checking functions, plus the functions of supervisory control, direct digital control, optimization, engineering simulation and other desirable tasks.

In this paper we will discuss the various process control tasks of a control computer and, in particular, the use of simulation -- off-line as well as on-line -- in control, optimization and engineering design calculations of a chemical plant with a process-control computer. The discussion will, needless to say, relate to the available software for the IBM 1800 Control System, although the technique is not limited to any particular control computer.

Figure 1 shows the various functions which the presently available IBM control computer performs. The central executive program is the time-shared executive system (TSX).[3] This program allocates the processing time available for the various functions which a control computer is required to perform. The time allocation is done in an optimum fashion, in that the calculations which have the utmost importance have the highest priorities. There are twenty-four or more levels of priorities available on a computer, allowing any higher level priority calculation or computer function to interrupt a lower priority level function or calculation which is being carried out at a given time. The interrupted calculations are not destroyed, but stored in the core or on an auxiliary memory storage such as a disk. Thus, after the higher priority calculations are completed, the lower priority calculations which were interrupted are resumed at the point where they were interrupted. When any priority interrupt calculation is completed, the computer seeks for lower and lower priority calculations, until the interrupt calculations are completed. At that time, it is available for offline nonprocess calculations or it is in idle mode. A core clock is provided to perform routine functions such as data logging and report generation at specified time intervals. The limit checkup of plant variables (violation of upper or lower limits) is done by a comparator, and the normal computer calculations are not interrupted unless a limit is violated, and specific computer processing action is required as a result of the variable limit violation.

COMPUTER FUNCTIONS

Let us look at the various functions and steps in process control that are performed by the computer.

Data Logging and Management Information

As soon as a computer is installed, it is used for data logging and management information such as production of a particular chemical in 24 hours, the amount of fuel used, etc. The logged data is also important in model verification which will be discussed in subsequent sections. The importance of obtaining reliable plant data for model verification cannot be overemphasized. The computer is much more reliable for logging than a human operator. Furthermore, a programmed filter (such

as time averaging, exponential smoothing, etc.) can be used to sort out the instrument signal from noise. Other routine functions such as alarm messages and emergency operator action sequence can be handled as a first step in computer control.

Regulation

A second function which a computer performs is regulation. The easiest regulation is by means of an operator guide which is defined as a sequence of control action messages to be taken by the plant operator when disturbances or upsets in certain plant variables are detected by the computer. To cite an example, a cement plant had a set of four kilns of identical design, all fed by a single raw material slurry feeder and burners using a single fuel. Each kiln was controlled by a single operator for every shift. From a total of twelve operators, two were able to control their kilns in the best manner. It is easy in a case like this to program within the computer the sequence of actions which the best operator takes as a result of the disturbances. This sequence of instructions is printed for all operators to follow when similar disturbances occur. In this manner, the operation of all kilns reaches the level of best operator performance.

Supervisory Control

In a normal plant operation, the operator makes changes in set-point positions for one or more control loops when a disturbance occurs. For a chemical plant with a large number of control loops, there are unavoidable interactions between control loops. An experienced operator can handle two to three loop interactions and make simultaneous changes in set points of the loops to bring the plant to a desired level of control. It is difficult, if not impossible, for the operator to handle more than three loop interactions. On the other hand, a digital computer can be programmed to adjust one or more of the set points simultaneously by solving a set of equations which relate the variables of the plant to the multiple set points (target variables). A typical equation may look like equation 1.

$$0 = F_1 \Delta T + F_2 \Delta M + F_3 \Delta U + F_4 \Delta V_1 + F_5 \Delta V_2 \cdots \quad (1)$$

where F_1, F_2, F_3 etc. are constants, T is the target variable to be controlled, M is the manipulated variable, U is the uncontrolled (disturbance) variable and V_1, V_2, V_3 are other measured variables which enter from interactions with other loops. Thus, for interacting loops a set of equations such as equation 1 is solved to control a set of target variables.

In this manner the computer can not only handle multiple-loop interactions, but also make the changes in a consistent manner. In addition, since each loop has different dynamic characteristics (slow or fast response, short or long dead time, first, second or higher order system), each set-point adjustment can be carried out by the computer in such a manner that different dynamic characteristics of each individual loop are taken into account. Although this technique is in no sense an optimum dynamic control scheme, it is, because of its practical value labeled a poor man's dynamic control scheme. Figure 2 shows how this scheme works. Let t be the time at which the computer calculations require that an adjustment of set points from say 60% of scale to 80% of scale. The adjustment may be made in several steps of time intervals. Let us say that the interval between two adjustments is Δt and the full adjustment is to be carried out in five steps. If the loop response is fast, the adjustment can be performed in five equal increments, or any combination of unequal increments which make the total shift of 20% scale. For a sluggish loop, one can provide incremental steps which includes overshoot as exemplified by the dotted line in Figure 2. If there is a dead time involved, one may incorporate a delayed action adjustment for the set point.

In certain areas of the plant, it is possible that one or more control valves are changed manually, and no analog feedback controllers are provided. The computer can perform the adjustment of these valves via a stepping motor if adjustment equations for changing the valve positions are programmed in the computer. Thus, the supervisory control can also provide a direct computer control (DCC) of the plant valves. As opposed to direct digital control (DDC), the DCC function

only provides proportional control and at more prolonged intervals (1 minute or more). DDC takes only seconds.

SUPERVISORY PROGRAM

The various functions described above, routine data logging, alarm scanning, limit checking and operator action as a result of limit violation, programmed operator guide control, supervisory control and direct computer control can be performed with an IBM program known as PROSPRO/1800 (Process Supervisory Program for 1800). The details of this program are in reference 4.

In brief, PROSPRO program is written in such a manner that virtually no programming knowledge is required on the part of plant personnel. However, the program requires that personnel know the behavior of the plant thoroughly, and, more important for our discussion, that relationships for interacting variables in the plant be written in mathematical form. A general type of relationship is available within the program, and one only needs to fill in the value of constants in the general adjustment equation.

PROSPRO requires that each variable in the plant be uniquely identified by a number. Then, for each variable, six standard information sheets are filled out. Two of the six sheets, the variable information sheet and the adjustment information sheet, are shown in Figure 3. Most of the required information is given by entering a 0 or 1 in certain columns, giving values of limits, actions as a result of violating these limits, and so on. The adjustment equation relating the variables of the plant is linear; however, the values of the coefficients F_1, F_2 ... may be calculated by using a general type of nonlinear equation such as

$$F_j = A + B\,[C + (F_k / D)]^E \qquad (2)$$

where A, B, C, D and E are values of constants to be supplied by plant engineers. Once the six sheets are completed for each variable, the information from the sheets is transferred to punched cards and read in PROSPRO as data. The sheets provide permanent documentation and any alteration in information on one or more variables is carried out by altering the corresponding sheet and the data card, and reading in the new card in PROSPRO. Some of the features and capabilities of PROSPRO are outlined in Table 1.[1]

DIRECT DIGITAL CONTROL

In some chemical plants, it may be desirable to install a digital computer instead of analog controllers, especially when a new plant is being built. This is commonly referred to as direct digital control. It is important to note that a digital computer can not only perform the function of analog controllers such as proportional, derivative, and integral control, but also the more sophisticated control functions such as nonlinear control, control of loops with fixed dead time and variable dead time, adaptive gain tuning, and so forth, at no additional cost. Analog control devices which perform these functions can become prohibitively expensive as the number of loops requiring sophisticated control increases. In addition, a digital computer performs the routine logging, alarm scanning, and limit checking functions.

Direct digital control on a chemical plant with an IBM 1800 is attained by the standard program, DDC/1800. Again, for details of the program and control techniques, refer to refernce 5.

OPTIMIZATION

Although significant benefits are achieved by using digital computer control for better plant regulation, improved product quality, and reliable plant information for operating personnel and management, it is felt that a substantial additional benefit can be achieved by means of optimization, especially in petro-chemical plants. Optimization is defined as maximization of plant profit by means of controlling the plant target variables in such a way that the constraints on all plant variables are satisfied. All chemical plants are dynamic in the sense that the constraints on the variables are always moving. It can be shown mathematically that the optimum operation of the plant shifts as a result of shifts in constraints. For example, the maximum heat load on a heat exchanger using cooling water may depend on the amount of cooling water and its temperature, the extent of fouling in the heat exchange tubes, and so on. The heat load constraint may limit a multistage compressor load if the heat exchanger is used for interstage cooling of the gas being compressed. Any change in heat load capacity of the heat exchanger will affect the optimum operation of the plant.

To define optimization in real time mathematically, let $x_1, x_2, x_3 \cdots x_n$ be a set of independent variables (generally set-point positions or valve positions) and $y_1, y_3, y_3 \cdots y_m$ be a set of dependent variables (temperatures, pressures, flow rates, heat load, compressor load, etc.). Let y_1 be a profit function

$$y_1 = f_1 (x_1, x_2, x_3 \cdots x_n, y_2, y_3, y_4 \cdots y_m) \tag{3}$$

and let y_2, y_3, y_n be functions of x_i and y_j.

$$y_2 = f_2 (x_1, x_2, x_3 \cdots x_n, y_3, y_4, y_5 \cdots y_m) \tag{4}$$

$$y_3 = f_3 (x_1, x_2, x_3 \cdots x_n, y_2, y_4, y_5 \cdots y_m) \tag{5}$$

and so on.

The optimization is performed by selecting a set of x_i which maximizes the function f_1 subject to the constraints

$$x_{1L} < x_1 < x_{1U}$$

$$x_{2L} < x_2 < x_{2U} \qquad (6)$$

$$\vdots$$

$$x_{nL} < x_n < x_{nU}$$

and

$$y_{2L} < y_2 < y_{2U}$$

$$y_{3L} < y_3 < y_{3U} \qquad (7)$$

$$\vdots$$

$$y_{mL} < y_m < y_{mU}$$

where subscripts U and L indicate maximum and minimum values, respectively.

The mathematical problem of optimization may be further complicated by imposing additional constraints; that is, one or more x and y may have an upper or lower limit which should be respected <u>most</u> of the time. Furthermore, plant experience may dictate that a certain independent or dependent variable should be close to a particular target value <u>most</u> of the time. These two types of constraints are defined as soft and target constraints, as opposed to hard constraints defined by equations 6 and 7. Mathematically, a cost penalty is invoked for violating upper or lower limits. In case of target constraints, a penalty is imposed for deviation on either side of the target.

Plant experience may dictate a minimum move on x_i, in order to avoid unnecessary disturbances which cause regulation problem, offsetting a small gain in profit as a result of the move of the independent variable by the optimizer. This is defined as a dead zone.

A program, designed to perform online optimization with a control computer taking into consideration, hard, soft and target constraints, dead zones, and non-linearity of the plant behavior, is now available for the IBM 1800. This program is called the control optimization program (COP/1800).

It requires a program written either in FORTRAN or assembly language called MODL0 which calculates the value of $y_1, y_2 \ldots y_m$ from a given set of x_i values. The optimizer calls the program MODL0 to calculate a gradient matrix $\frac{dy}{dx}$ at a starting point x. Here x and y are the vectors of independent and dependent variables. The numerical evaluation of dy/dx is carried out with a different value of Δx_i for each i to avoid difficulties because of differences in dimensions of individual x_i's. Equations 3-5 are then approximated by linear expansion in a 'linearity limit' which is larger than the value of Δx_i for each i. With linear programming technique, a linear optimum profit is calculated which satisfies the various constraints imposed on the optimizer. The set of optimum x_i's are then used to calculate a corrected nonlinear profit y_1 which is compared with the previous value of y_1. If the profit shows an increase, a new gradient matrix dy/dx is calculated at the new set of x_i and the above procedure is repeated. Several features are included in the optimizer to help avoid the well-known occurrence of polymodality -- one such feature being the adaptive linearity limit. Experience indicates that the best procedure to avoid getting trapped into a suboptimum is by judicious selection of the ratio of linearity limit to Δx_i for each i. The ratios may be obtained by performing optimization off-line on the computer, using various values of these ratios. See references 2 and 10 for details on the program and techniques employed.

SIMULATION

In the discussion of control so far, it has been implied that the relationships between the variables within the plant are known in the form of equations 1 and 2 for supervisory control and in the form of equations 3 through 5 for optimization. A chemical plant, in general, is quite complex and, in many instances, these relationships have not been developed by plant personnel. An alternative is to obtain the relationships using some sort of regression analysis on the plant data collected in the initial phases of the computer installation.

It is much better, in my opinion, to simulate the process offline on a digital computer by writing down the equations describing the physical and chemical phenomena occurring in the plant, and then solving these equations with the aid of the computer to yield the relationships in the form of equations 1 through 7. The digital simulation can be performed during idle time of the control computer (Fig. 1, block IV) as a nonprocess program or preferably on an offline computer several months prior to installation of the control computer. In either case, there are several computer languages which could be used for this purpose: FORTRAN, CSMP (Continuous System Modelling Program) and DSL (Digital Simulation Language). The latter two are especially useful to engineers, since they are oriented to the mathematical language of process engineers.

Simulation of a process also has some added benefits. An engineer who has relatively little experience in the plant behavior can get 'hands on' training on the sensitivity of the plant to changes in certain variables with the aid of the simulation equations, provided that the equations do represent the plant behavior with fair accuracy. Furthermore, the range of changes in the simulated plant can be much larger than that observed in day-to-day drift in the plant. Often, the simulation may indicate a plant operation significantly beyond the normal operating point, with higher plant throughput or product quality. In any case, the reliability of a good simulation model is much better than the reliability of a model generated by regression equations.

The rest of this paper, then, will deal with simulation model development for the purposes of supervisory control and optimization of the process.

MODEL DEVELOPMENT

On examining a process under consideration for simulation, one finds that, in general, large numbers of units are involved in a plant, and it is virtually impossible to carry out detailed mathematical analysis for each unit. The desire is to obtain, ultimately simple mathematical relationships which can perform control and optimization calculations with a control computer in real time. It is then necessary to select for detailed analysis those units in the plant which are most important from a profitablility standpoint and those units which are difficult to regulate. The other units may be capacity limiting and can be simply handled as constraints.

A mathematical model of a chemical plant unit generally consists of differential equations describing:

1. Material balance
2. Energy balance
3. Pressure balance

The material balance equation may involve chemical reaction as well as diffusion, whereas the energy balance may involve heat of reaction. For all the three balances, the standard form of equation in qualitative terms is

$$\text{Input} - \text{Output} = \text{Accumulation} + \text{Generation (or Depletion)} \qquad (8)$$

The 'generation' term takes into account the formulation or disappearance of a reactant in chemical reaction and heats of reaction.

For steady state, the 'accumulation' term is zero. The nonsteady-state case gives rise to the so-called 'dynamic models.' Dynamic models are useful in processes which have a long residence time for the material in the plant unit. Manufacture of cement in a kiln is such a process. When the residence time is short, the steady-state models are adequate for representing the process. The dynamic models in most cases consists of a set of nonlinear simultaneous partial differential equations which are difficult to solve even on large digital computers. The steady-state models are somewhat easier to solve since they give rise to ordinary differential equations. In this paper we will deal only with the steady-state case, because most significant units in petro-chemical plants have short residence times for the throughput material.

In this paper we will illustrate the various steps involved in developing a mathematical model for on line optimization and supervisory control of a methanol plant. The type of plant analyzed here employs high-pressure catalytic reaction

$$CO + 2H_2 \rightleftharpoons CH_3OH$$

and the pressure is generated by means of a high-speed centrifugal compressor driven by steam turbines.

We will show how simulation helps in developing online relationships for controlling the plant.

Figure 5 shows a simplified diagram of the plant. There are two sections: one is the reformer and the other the synthesis loop. In the reformer section, natural gas is reacted with steam to yield CO and CO_2.

$$CH_4 + H_2O \rightleftharpoons CO + 3H_2$$

$$CO + H_2O \rightleftharpoons CO_2 + H_2$$

CO_2 is removed from the reformed gas after heat is recovered in the waste heat boiler and the gas mixture containing H_2, CO, CH_4 and small amounts of CO_2 is fed to compressor. Often, some amount of CO_2 is added to the natural gas in the reformer to have a more favorable chemical equilibrium for CO. The gases after compression to 3500 psig or more are mixed with a recycle stream and sent to the synthesis converter where methanol is formed. The product gases from the converter are cooled and liquid methanol is removed in separator tanks and led to a distillation train for purification. Part of the gas from the separator tanks is purged and the rest is fed to the recycle wheel of the compressor.

First of all, three plant units, namely, (1) primary reformer, (2) methanol synthesis converter, and (3) compressor are important from a computer control standpoint.

Compressor Control Problem

The compressor has two control problems. Steady-state analysis of the compressor yields horsepower requirements (and hence steam requirements in turbines) and pressure to the methanol converter, whereas the dynamic analysis is necessary for surge control. With the help of plant data or compressor curves supplied by the compressor manufacturer, one can design a simple control scheme for spill-back valve control in compressors to avoid violation of surge limits. A simple scheme is described by Magliozzi[6] wherein the surge limit line on the compressor curves is linearized and a controller is designed to avoid the surge. The same scheme can be expanded by fitting a polynomial relationship to the surge line, and a control algorithm can be written to perform nonlinear surge control functions by means of a digital computer.

The simulation of the compressor is used in conjunction with the model of the synthesis converter to predict compressor load.

We will briefly discuss the model development of three units used in methanol plants; for a detailed analysis, see reference 7.

Synthesis Converter

This unit is the most important since the production of methanol takes place here. Apart from the reaction of methanol formulation, other side reactions occur at high temperature, the most significant being the formation of ether

$$CO + 3H_2 \rightleftharpoons CH_3 OCH_3$$

In figure 6 we show a simplified diagram of the converter. There are several catalyst beds and a heat exchanger. For each bed the material, energy and pressure balance equations are

$$\frac{d N_{methanol}}{d z} = f_1 (N_{CO}, N_{H_2}, N_{CH_3 OH}, P, T) \tag{9}$$

$$\frac{d T}{d z} = f_2 (N_{CO}, N_{H_2}, N_{CH_3 OH}, T, \Delta H (T, P)) \tag{10}$$

$$\frac{d P}{d z} = P_0 - a z \tag{11}$$

z is the tube length, and ΔH is the heat of reaction. P_0 is the gas pressure at the entrance of each bed. The functions f_1 and f_2 are highly nonlinear and, because equations 9 and 10 are interdependent, they must be solved numerically. Furthermore, as pointed out in reference 8, the solution of these equations for the entire reactor poses a trial and error problem due to the fact that the temperature boundary conditions for the first bed are dependent on reactor product temperature, a classic example of a two-point boundary value problem.

It must be pointed out that the differential equation model for the reactor (equations 9 through 11) is not suitable as an online optimization relationship, because of the long time required for the solution. Since for our purpose it is only desired to relate converter exit temperature and methanol production to temperature, pressure, and composition of feed, a number of simulation runs are made on the

Synthesis Converter

This unit is the most important since the production of methanol takes place here. Apart from the reaction of methanol formulation, other side reactions occur at high temperature, the most significant being the formation of ether

$$CO + 3H_2 \rightleftharpoons CH_3 OCH_3$$

In figure 6 we show a simplified diagram of the converter. There are several catalyst beds and a heat exchanger. For each bed the material, energy and pressure balance equations are

$$\frac{d N_{methanol}}{d z} = f_1 (N_{CO}, N_{H_2}, N_{CH_3 OH}, P, T) \tag{9}$$

$$\frac{d T}{d z} = f_2 (N_{CO}, N_{H_2}, N_{CH_3 OH}, T, \Delta H (T, P)) \tag{10}$$

$$\frac{d P}{d z} = P_0 - a z \tag{11}$$

z is the tube length, and ΔH is the heat of reaction. P_0 is the gas pressure at the entrance of each bed. The functions f_1 and f_2 are highly nonlinear and, because equations 9 and 10 are interdependent, they must be solved numerically. Furthermore, as pointed out in reference 8, the solution of these equations for the entire reactor poses a trial and error problem due to the fact that the temperature boundary conditions for the first bed are dependent on reactor product temperature, a classic example of a two-point boundary value problem.

It must be pointed out that the differential equation model for the reactor (equations 9 through 11) is not suitable as an online optimization relationship, because of the long time required for the solution. Since for our purpose it is only desired to relate converter exit temperature and methanol production to temperature, pressure, and composition of feed, a number of simulation runs are made on the

differential equation model, and the results of the model are fitted to simple polynomial type relationships using regression analysis on the results. The solution of equations 9 through 11 must, of course, agree with any available plant measurements of temperature and composition at the reactor exit. Any discrepancy between plant data and model results can be reduced by adjusting kinetic parameters and heat-transfer coefficient values, since these are not known too accurately in the model. Once the online relationship for the reactor is obtained from simulation results, we are ready to combine with the model of the rest of the plant.

Compressor Capacity Relationships

As indicated earlier, and in reference 8, the compressor model is developed from the compressor curves relating polytropic heat to flow rate with compressor speed as a parameter. The efficiency curves for each stage may be fitted in the same manner to flow rate.

The pressure calculation for each stage is then carried out by using the standard equation

$$P_{n+1}/P_n = \left(\frac{H_p}{E} \cdot \frac{(k-1)}{k} \frac{1}{R\,T_n} + 1 \right)^{\frac{k \cdot E}{k-1}} \frac{C_{n+1}}{C_n} \qquad (12)$$

where H_p is the heat developed at stage n, E is the efficiency, R is the ratio of heat capacities at constant pressure to heat capacity at constant volume, and P_n and T_n are pressures and temperatures of gas entering stage n. C_n and C_{n+1} are compressibility factors for entrance and exit gases in stage n.

Compression horsepower requirement is given by

$$\text{Horsepower} = \frac{W \cdot H_p}{550} \qquad (13)$$

where W is the gas flow in lbs/second.

The gas temperature at the exit of stage n is given by

$$T_{n+1} = T_n \cdot (P_{n+1}/P_n \cdot C_{n+1}/C_n)^{k-1/k} \quad (14)$$

The separator section model is again of a simple form [8]

$$\text{Moles gas in liquid methanol} = (N_i)_L = a_i + b_i p + c_i T \quad (15)$$

and

$$\text{Moles methanol in gas} = (N_{meth})_g = d + ep + fT + gT^2 \quad (16)$$

This completes the model for the methanol synthesis loop.

Reformer

The reformer model uses the kinetics of Moe[7] CO and CO_2 are assumed to reach chemical equilibrium whereas the formation of CO. from CH_4 and H_2 is dependent on kinetic rate. As indicated by Shah,[9] the following equations are obtained on the basis of these assumptions.

$$(N_{CH_4})_{\text{exit reform}} = f_3 (T_{exit}, P_{exit}, (N_{H_2O})_{inlet}, (N_{CH_4})_{inlet}) \quad (17)$$

$$(N_{H_2O})_{\text{exit reform}} = f_4 (T_{exit}, P_{exit}, (N_{H_2O})_{inlet}, (N_{CH_4})_{inlet}) \quad (18)$$

$$\text{Fuel required} = FU = f_5 (T_{inlet}, f_3) \quad (19)$$

Stoichiometric equations subsequently give values of H_2, CO and CO_2 moles at the reformer exit from equations 17 and 18.

Equations 9 through 18 represent simulation of the most important units of the methanol plant, namely, the synthesis converter, compressor, separator, and the

reformer. The CO_2 absorption unit generally removes 99.5% of CO_2, and the various heat-exchanger simulation uses standard heat-transfer equations.

The next important problem is how to use the various simulation models to derive the profit equation for the plant in the form of equations 3 through 7.

A selection of independent and dependent variables, their constraints, and cost factors is required at first. Table II shows a list of the independent and dependent variables.

The stream composition of gas entering the synthesis reactor is treated as an independent variable, simply because experience suggests that the synthesis loop is the plant bottleneck which cannot be overcome easily. The compressor make-up steam , flow, and composition are thus treated as dependent variables, and the only independent variables in the reformer are exit temperature (set point) and steam-to-natural gas ratio in feed. This selection of independent and dependent variables also makes it convenient from a calculation standpoint, since the trial and error problem as a result of the recycle steam in synthesis loop (which arises if the compressor make-up steam is treated as an independent variable) is eliminated.

The profit function calculation for optimizing then follows the steps listed below:

1. Calculate converter exit temperature and composition, based on feed rate, temperature and pressure.
2. Calculate heat load on the heat exchanger of the boiler feed water.
3. Calculate temperature of product entering separator.
4. Calculate vapor-liquid equilibrium and hence vapor and liquid composition in separator.
5. Calculate recycle stream, purge gas flow, and composition.

6. Calculate make-up steam to compressor, compressor horsepower, and pressure to synthesis converter.

7. Calculate natural gas required for make-up stream and composition (initial guess).

8. Calculate reformer exit composition and correct natural gas requirement, if CH_4 and H_2 do not agree with the make-up stream requirement.

9. Calculate profit function as follows:

$y_1 = A_1$ (Methanol production) - A_2 (Methanol loss in purge) + A_3 (Heat recovery in boiler feed water heater + Waste heat boiler + reform boiler) - A_4 (Natural gas to reformer) - A_5 (Fuel) - A_6 (Steam to compressor turbines) - A_6 (Steam to reformer).

We do not wish to present a detailed discussion on the optimization results. Briefly, the optimizer pushes the plant to one or more of the following constraints:[8]

1. Maximum compressor horsepower (shaft horsepower or steam availability limiting horsepower).

2. Maximum cooling water flow rate in heat exchanger prior to separator tank.

3. Minimum steam/gas ratio.

4. Maximum reformer exit temperature.

5. Maximum waste-heat boiler capacity.

SUPERVISORY CONTROL EQUATIONS

We will illustrate the method of obtaining control equations, similar to equation 1 for use in supervisory control with PROSPRO as applied to the reformer. As pointed out in reference 9, the reformer control is based on maintaining exit temperature and percent of methane in the gas to a certain level. The equations relating the exit temperature and percent methane to steam feed and burner fuel

flow in the reformer are quite complex (equations 17 through 19). However, in the plant operating range, these equations may be linearized.

$$F_1 \Delta(\%\ CH_4) = F_2 \Delta(1/\text{steam flow}) + F_3 \Delta(1/\text{exit temp}) \qquad (20)$$

$$\begin{aligned} F_4 \Delta(\text{Exit temp}) = {} & F_5 \Delta(\text{Fuel}) - F_6 \Delta(\text{Gas temp inlet}) \\ & -F_7 \Delta(\text{Steam inlet temp}) \\ & -F_8\ (\text{Inlet gas flow}) - F_9\ (\text{Inlet steam flow}) \end{aligned} \qquad (21)$$

The value of the coefficients F_1 through F_9 are derived from simulation runs around the operating point by fitting the simulation results to equations 20 and 21. If the sampling rates for measuring exit methane concentration is fast (infrared analyzer), one can improve regulation by including in equations 20 and 21 past values of the changes in the variables, appropriately weighted.

Comparing equations 1 and 20, it is obvious that the target variable T is percent of CH_4 in the exit gas, and the manipulated variable U is the steam flow. In equation 21, exit temperature is T and fuel flow is U. Equations 20 and 21 must be solved simultaneously to affect feedback control of the reformer.

Similar equations can be written for the methanol synthesis converter, heat exchanges, and purge control in the synthesis loop.

APPLICATION TO DESIGN

In the above discussion we have illustrated the use of simulation in deriving control equations for supervisory control, as well as developing a model profit-function for plant optimization (COP).

It may be desirable to make changes in the plant unit during the course of a computer control project. Plant optimization with COP, for example, may indicate

a heat exchanger, or reactor as a bottleneck because it is underdesigned. The computer then can be utilized for redesigning the plant unit in an optimum manner. Either the available simulation languages or FORTRAN can be used for simulating the process unit. The basic simulation equations are generally the same for design as well as operation. However, in plant simulation the design parameters such as reactor area and length are constant, but the flow, temperature, and operating conditions are variant. In design simulation, one needs to vary the design parameters for a given set of operating conditions. With a change in the independent and dependent variable definition, the control optimization program can be used for optimizing the design of a plant unit. The profit function which is maximized will in this case be the negative value of the design cost, so that a minimum cost will be obtained.

SUMMARY AND CONCLUSIONS

In this paper we have described the various steps involved in the application of a digital computer control on a chemical plant. We have described the various functions that an IBM 1800 performs with the assistance of an executive program which performs optimum allocation of computer processing time to these functions on a predetermined priority basis. We have also described the standard IBM software available for performing some of the functions.

To attain the maximum benefit with a digital computer, the use of simulation is described in deriving the necessary supervisory equations for the controller and profit equation for the optimizer. An example of a methanol plant is given to illustrate the use of simulation in deriving these equations on the basis of fundamental laws of physics and chemistry. It is also shown how the same simulation model can be applied to design and even to optimizing a plant unit design.

It is to be emphasized, in conclusion, that all these functions can be and are being performed with an 1800 on various chemical plants around the world, providing an optimum utilization of control computers.

REFERENCES

1. Ewing, R. W., G. L. Glahn, R. P. Larkins, and W. N. Zartman, Chem. Eng. Progress, 63, No. 1, 104 (1967).
2. IBM 1800 Control Optimization Program, Application Description, 1800-CC-01X, IBM Program Information Dept., Hawthorne, N.Y.
3. IBM 1800 Time Sharing Executive System, Form No. C26-5990-02, IBM Program Information Dept., Hawthorne, N.Y.
4. IBM 1800 Process Supervisory Control Program, Application Description, 1800-CC-2X, IBM Program Information Dept., Hawthorne, N.Y.
5. IBM 1800 DDC, Direct Digital Process Control, 1800-23.5-002, IBM Program Information Dept., Hawthorne, N.Y.
6. Magliozzi, T.L., Chemical Engineering, p. 139, May 8, 1967.
7. Moe, J.M., and Gerhard, E.R., "Chemical Reaction and Heat Transfer Rates of Steam-Methane Reaction," paper presented at the A.I.Ch.E. 56th National Meeting, San Francisco, May 1965.
8. Shah, M.J., and Stillman, R.E., "Control Simulation and Optimization of a Methanol Plant," paper presented at the Joint A. I. Ch.E.-I.M.I.Q. Meeting, Mexico City, Sept. 24, 1967.
9. Shah, M.J., and Weisenfelder, A.J., "Digital Computer Control and Optimization of a Low Pressure Centrifugal Compressor Ammonia Plant," paper to be presented at the IFAC Symposium on Multivariable Control System, Dusseldorf, Oct. 7, 1968.
10. Weisenfelder, A.J., Fritz, J.C., and Thompson, W.G., ISA Preprint, 33-3-66.

Table I PROSPRO System Summary

Standard fill-in-the-blank programming forms
Compatible supervisory control configuration
Incremental feedback and feedforward control action
Multitype and multivalued process variables
Consistent processing procedures and interprogram communication
Versatile logic, calculation, and conversion routines
Advanced interacting and sequential control capabilities
Streamlined program modification and documentation

Table II

List of Independent and Dependent
Variables for a Methanol Plant

Independent Variables:

x_1	=	Total moles feed to converter
x_2	=	H_2/CO ratio in converter feed
x_3	=	CH_4/CO ratio in converter feed
x_4	=	cold split 1 in converter
x_5	=	cold split 2 in converter
x_6	=	cold split 3 in converter
x_7	=	boiler-feed water rate to heat exchanger
x_8	=	cooling water rate to cooler before separator
x_9	=	compressor speed
x_{10}	=	water rate to compressor intercooler stage 1
x_{11}	=	water rate to compressor intercooler stage 2
x_{12}	=	water rate to compressor intercooler stage 3
x_{13}	=	water rate to compressor intercooler stage 4
x_{14}	=	Steam to gas ratio in Reformer
x_{15}	=	Fuel rate to Reformer
x_{16}	=	Fraction purge gas sent to reformer as fuel
x_{17}	=	Damper control in reformer stack

Dependent Variables:

y_1	=	profit, which is a function of methanol production, heat recovery in boiler feed water heater, H_2 loss in purge, cooling water cost, etc.
y_2	=	purge rate in separator
y_3	=	inlet temperature to converter
y_4	=	cooling water temperature exiting various heat exchangers
y_6, y_7, y_8, y_9	=	constrained temperatures at the exit of catalyst beds 1, 2, 3 and 4
y_{10}	=	product temperature exiting boiler feed water heater
y_{11}	=	water temperature exiting boiler feed water heater
y_{12}	=	temperature of product entering separator
y_{13}	=	pressure at separator
y_{14}	=	pressure at outlet of recycle wheel of compressor
y_{15}	=	total flow in compressor make-up stream
y_{16}	=	H_2/CO ratio in make-up stream
y_{17}	=	CH_4/CO ratio in make-up stream
y_{18}	=	Total horsepower requirement for compressor
y_{19}	=	% CH_4 in reformer exit gas
y_{20}	=	% CO_2 in reformer exit gas
y_{21}	=	Reformer exit gas temperature
y_{22}	=	Reformer Furnace box temperature
y_{23}	=	process gas feed to Reformer
y_{24}	=	steam required for compressor turbines
y_{25}	=	% CO_2 exit CO_2 absorber

1111

VARIABLE INFORMATION SHEET — 1 5: 0 2 0 5 5

FEED CONVERSION (PRODUCT CONCENTRATION)

TARGET VALUE CALCULATED BY G10138

7 8 10

0 0	AR-103 CONVERSN	Identification used for Message Output.
0 1	PCT	Engineering Units used for Console Display.
0 2	2	Process Unit of Variable. (0 = [illegible], 1 = [illegible], 2 = Next Unit)
0 3	3 2 1 1	Grid/Column/Row - Console Grid Position.
0 4	0 5 0 / 0 1 6 7	Routine Processing Interval / Processing Sequence.
0 5	2	Number of Routine Values used to determine Average Variable Value.
0 6		Is Avg. Value used for Reference, Delta and Deviation checks? (1 = Yes)

Current Value Processing:

1 0	1	Is Variable a Measured Input? (1 = Yes, following 7 items applicable)
1 1		Is Input continuously monitored for MAX-MIN Limit Alert? (1 = Yes)
1 2	0 1 3 3	Measured Input Address.
1 3	1 0 0.	Top of Scale.
1 4	0.	Bottom of Scale.
1 5	V	Option in Conversion Eq. (1 = Yes)
1 6	4 1 . 6 7	Bias (B)
1 7	6 6 . 6 7	Coefficient (C)
1 8		Calculate Current Value, Convert TC Input or Adjust Conv. Value.
1 9		Intermediate Special Action.

ADC Value Conversion Equation:

Converted Value = $B + C\sqrt{\frac{\text{optional ADC} - a}{b}}$

Note: Thermocouple input converted by specifying preassigned code in 18

Limit Checking and Special Actions:

2 0	9 0.	Assigned Process MAXIMUM Limit.
2 1	3 0 0 3 2	(No Viol. → Viol.) Special Action taken when passing through
2 2		(Viol. → No Viol.) the Assigned Process Maximum Limit.
2 3	6 0.	Assigned Process MINIMUM Limit.
2 4	1 0 2 0 9	(No Viol. → Viol.) Special Action taken when passing through
2 5		(Viol. → No Viol.) the Assigned Process Minimum Limit.
2 6		Upper Special Action Reference Point.
2 7		(Below → Above) Special Action taken when passing through
2 8		(Above → Below) the Upper Reference Point.
2 9		Lower Special Action Reference Point.
3 0		(Above → Below) Special Action taken when passing through
3 1		(Below → Above) the Lower Reference Point.
3 2	3.	Assigned Delta Limit for Operator Notification.
3 3		(No Viol. → Viol.) Special Action taken when Delta Limit exceeded.
3 4		Delta required for a Predictive Adjustment Action.
3 5		Action taken for a Predictive (Delta) Adjustment.

Target Value and Deviation Processing:

4 0	1	Does Variable have a Target? (1 = Yes, following 10 items applicable)
4 1	1 5	Minimum Time between successive Target Calc. and/or Dev. Adj.
4 2	1 0 1 3 8	Specified Action to evaluate New Target Value.
4 3	5.	Assigned Deviation Limit for Operator Notification.
4 4		(No Viol. → Viol.) Special Action taken when Dev. Limit exceeded.
4 5	.5	Deviation required for Normal Adjustment Action.
4 6	2 2 0 7 6	Action taken for Normal Deviation Adjustment.
4 7		Maximum Set Point Adjustment per pass.
4 8		Set Point Output. (1 = Controller, 2 = DDC, 3 = Operator Message)
4 9		Computer Set Controller / DDC Address.
5 0		Set Point Movement Rate. (1 = Max. rate, 2 = 1/2 Max., 3 = 1/3 Max., etc.)
5 1		Final Special Action.

Figure 3. Variable information sheet.

ADJUSTMENT INFORMATION SHEET — 1 5: 2 2 0 7 6

MAINTAIN CONVERSION AT DESIRED LEVEL

MANIPULATED VARIABLE - FURNACE OUTLET TEMPERATURE

FEEDBACK REQUEST - CONVERSION ; FEEDFORWARD REQUEST - FEEDRATE

Adjustment Equation: $0 = F_1 \Delta T + F_2 \Delta M + F_3 \Delta U + F_4 \Delta V_1 + F_5 \Delta V_2 + F_6 \Delta V_3$

Target - Current(Average) — Calculated Adjustment

Current(Average) - Last Base for U (Uncontrolled) Variable

Interacting Control Information. (Omit if single equation solution)

7 8 10 14 16 20 22 26 28 32 34

0 0	2		2		2		2			Re-solution acceptable? (1 = Yes)

Definition and Development of Terms in Adjustment Equation.

Defines Value of Coefficient F_n. (-nnnn, 1nnnn, 3nnnn or Blank*)

Identifies Process Variable.

7 8 10 14 16 *Const. F_n Coeff. 28 30 33

0 1	-0.191		2 0 5 5	T - Targeted Variable
0 2		-1.0	2 0 8 6	M - Manipulated Variable
0 3	1 0 1 4 3		2 0 6 3	U - Used for Predictive
0 4				V_1 - Adjustment and/or
0 5				V_2 - Simultaneous Equation
0 6				V_3 - Interactions.

Adjustment Equation Output Restrictions.

1 0	1 0 1 9 6	Action taken when Variable M Out of Service.
1 1	1 0 1 7 1	Action taken when Variable M Off Computer (On Operator).
1 2	1 0.	Maximum allowable adjustment to M per pass.
1 3	2 2 0 8 4	Action taken when Target of M already at MAX-MIN Limit.
1 4		Adjustment Reference List for Partial Loop Test. (2nnnn or Blank)
1 5	2 0 9 2	If Entry 14 specifies a Reference List, Entries 15-18 specify sublists
1 6		(-1, -2, -3, -4) referenced for all subsequent M's in control loop.
1 7		If Entry 14 is Blank, Entries 15-18 specify all of the subsequent M
1 8		Variables in control loop.
1 9		Is Target of M set to Average instead of Current for Partial Loop? (1 = Yes)

Manipulated Variable Adjustment (ΔM) Output Control.

2 1	1 0	TIME in minutes (bXXXX) or as developed (-nnnn or 1nnnn).

7 8 10 %T 14 %ΔM

2 2	0	5 0	(1)	%T = Cumulative elasped time from present time
2 3	2 0	7 5	(2)	expressed as a percent of TIME.
2 4	5 0	-2.5	(3)	% ΔM = Percent of ΔM change made after %T time.
2 5			(4)	(Note: Negative % ΔM is permissible.)

Special Action Initiated Upon Significant Change in Manipulated Variable (M).

7 8 10 Initiating M Change 22 21 Action

3 1	8.	1 0 2 2 2
3 2		
3 3		
3 4		
3 5		
3 6		

Action may be specified as follows:

(a) Process Variable Special (0nnnn).

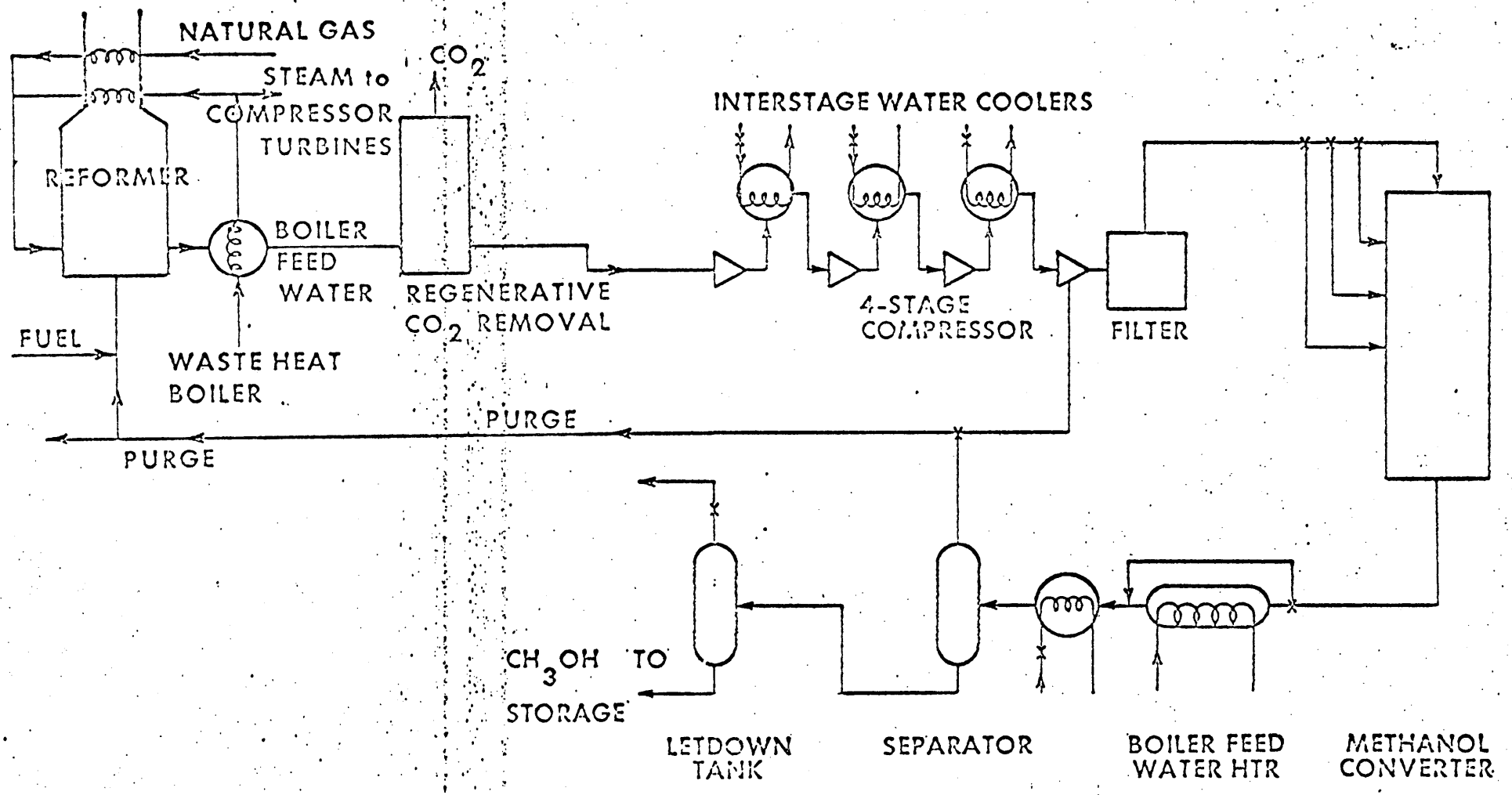
(b) Execute General Action (1nnnn).**

(c) Feedforward Request Call of Adjustment Equation (2nnnn).

(d) Execute Special Program (3nnnn).**

** Calculated ΔM in Result Register.

Figure 4 Adjustment information sheet.


NATURAL GAS
STEAM to COMPRESSOR TURBINES
CO_2
INTERSTAGE WATER COOLERS
REFORMER
BOILER FEED WATER
REGENERATIVE CO_2 REMOVAL
4-STAGE COMPRESSOR
FILTER
FUEL
WASTE HEAT BOILER
PURGE
PURGE
CH_3OH TO STORAGE
LETDOWN TANK
SEPARATOR
BOILER FEED WATER HTR
METHANOL CONVERTER

1143

Simplified Diagram of a Methanol Converter

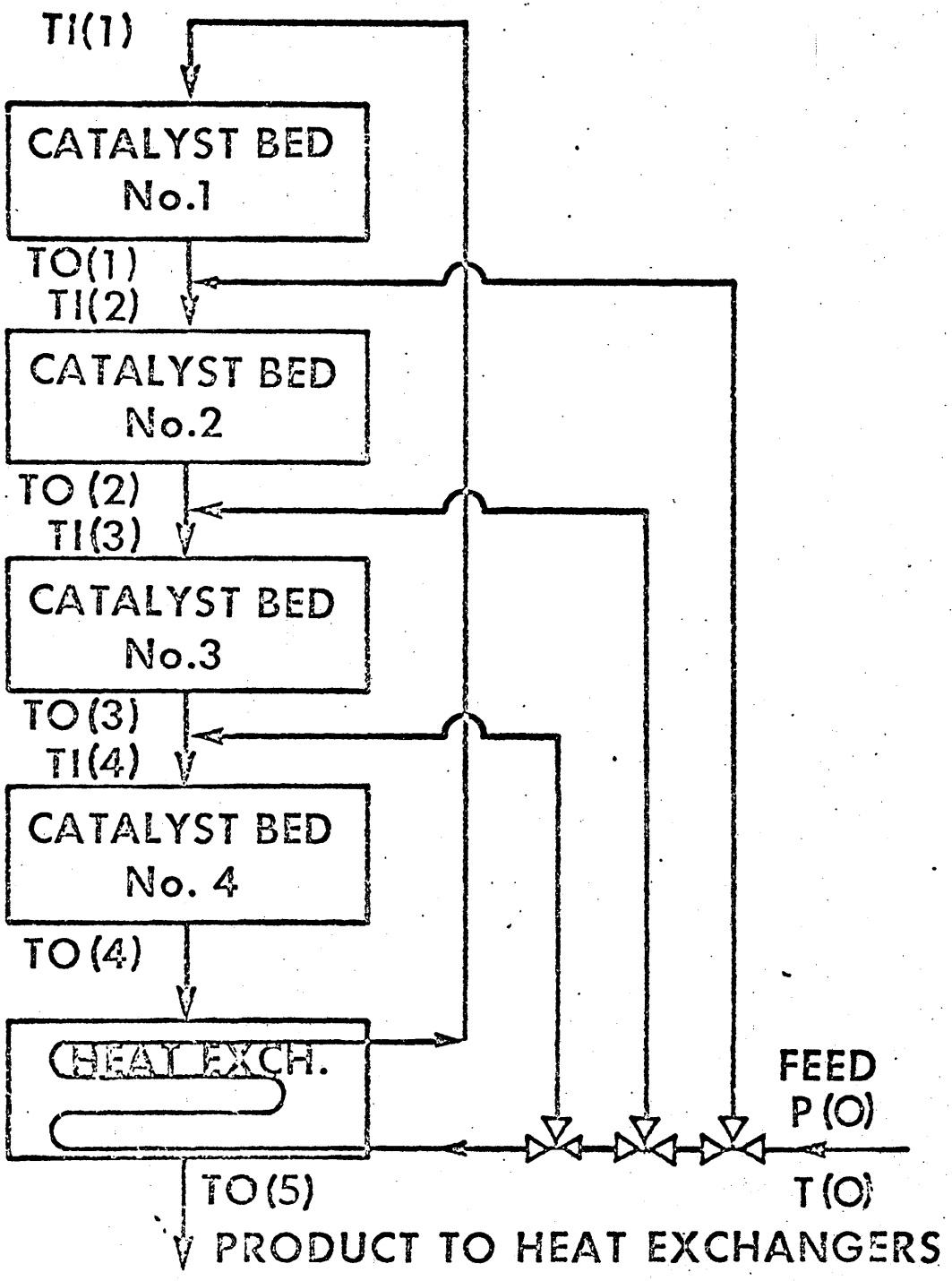


Figure 6

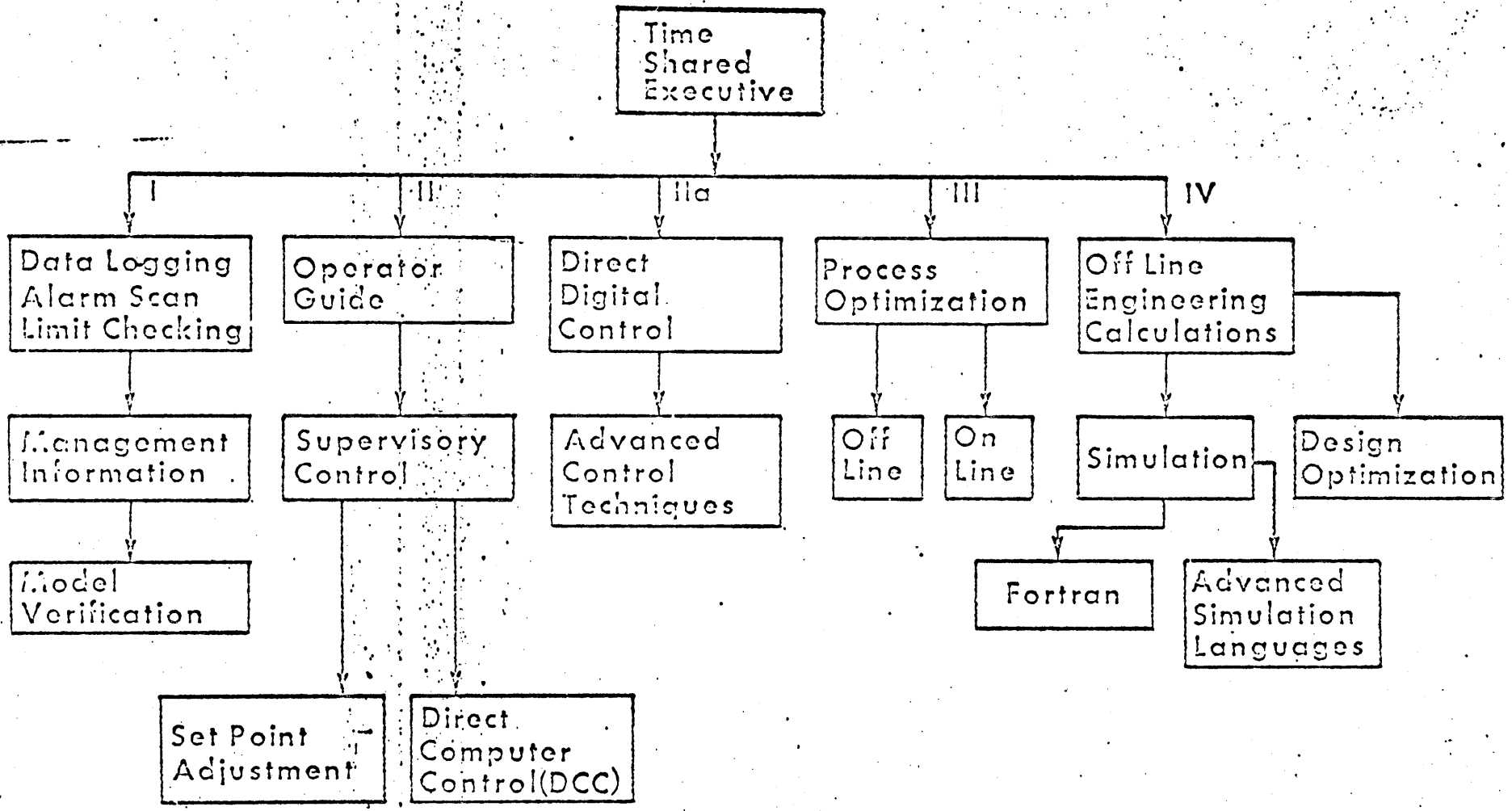


fig. 1 Various functions of process control computer.

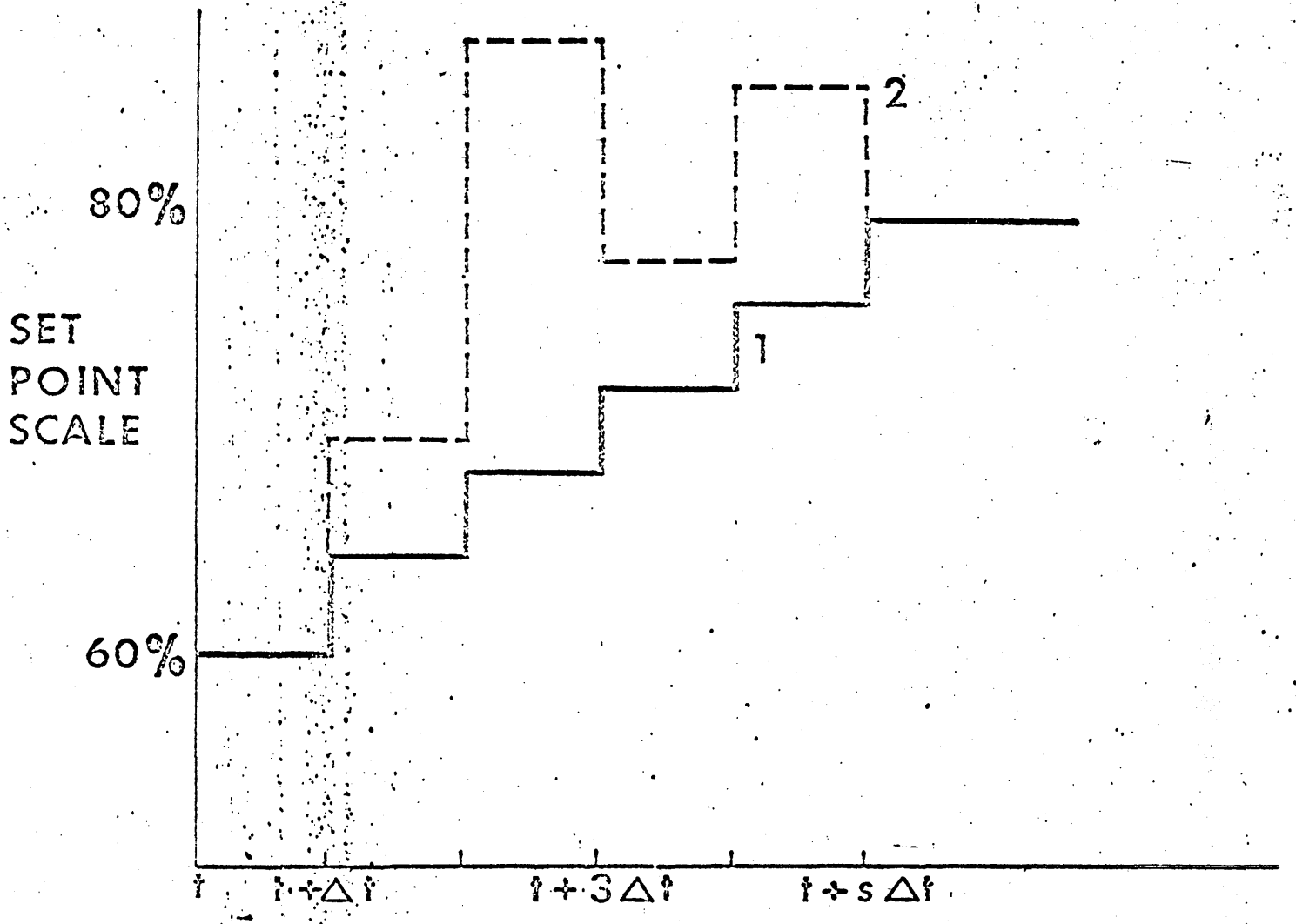


fig.2 Method of adjusting set point.

1. Sensitive loop.
2. Sluggish loop.

9444

T. 3. 5

COMMON User's Meeting

San Francisco Meeting December 11, 12, 13

Laboratory Automation

by Ray Edwards IBM

The session on Laboratory Automation will include a general description of programming and equipment aspects of the application of the 1800 and 1130 to this newly emerging and fast growing field. Following this discussion, an application will be discussed in detail. This will include a description of the type of research to be accomplished, the incentives of the on-line computer, the programming system to run the instrument in a closed loop fashion, and other experiences to date with the system.

T.3.6

DESIGN PHILOSOPHY

- SYSTEM OVERVIEW -

APPLICATION MONITOR

Executes Procedures in Specified Seq.

PROCEDURES

Handles a Specific Task (e.g. Sort)

PROGRAMS and SUBROUTINES

All Written in Fortran

Lowest Level Routines do General Purpose Functions

DESIGN PHILOSOPHY

- SYSTEM OVERVIEW -

CORE ORGANIZATION

Standard Compiler Allocation

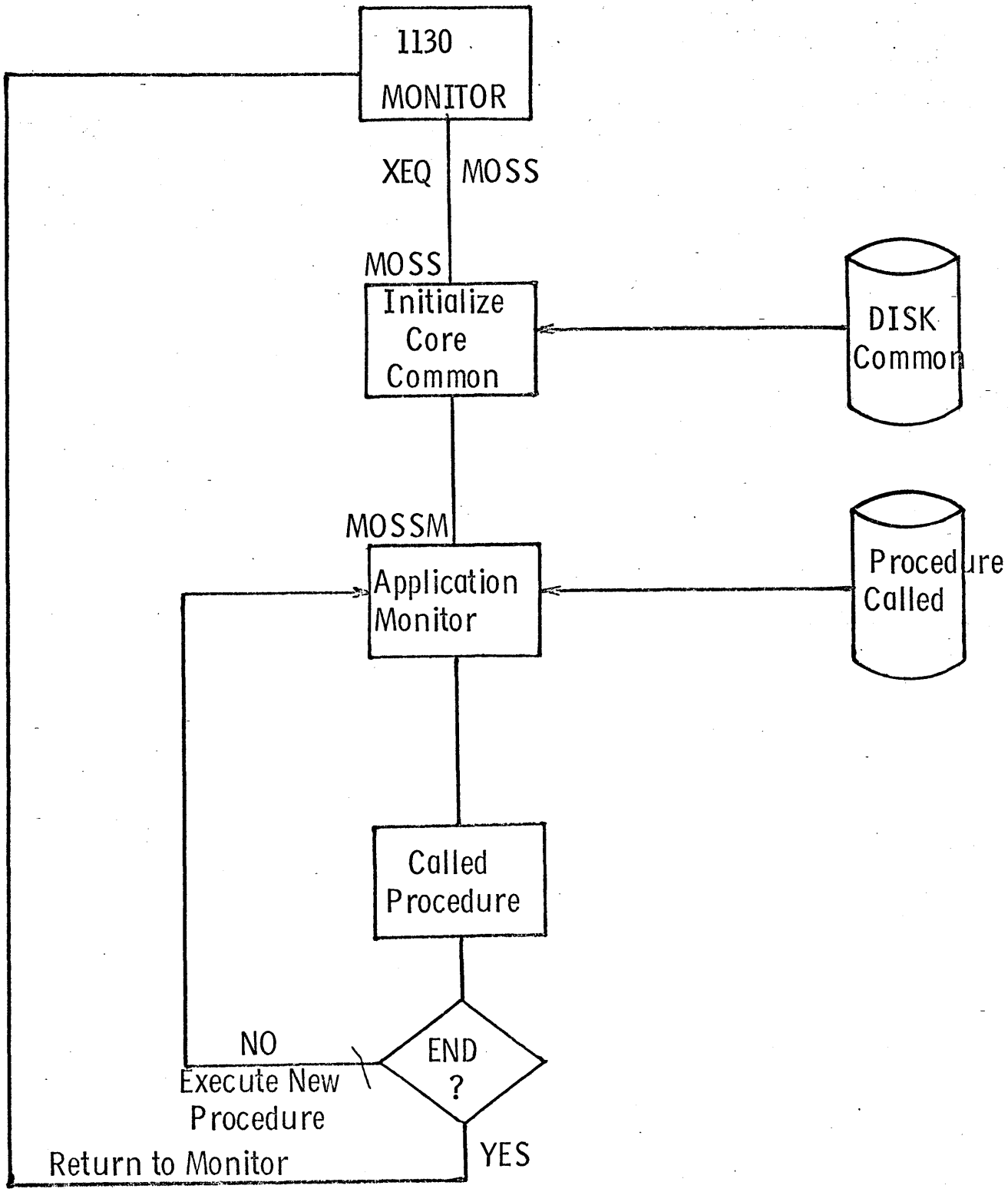
Two Sections: COMMON and WORK REGIONS

DISK ORGANIZATION

Single External File

Each Record One Sector Long

Files Use Simple String Method Via Regions


1130
MONITOR
XEQ
MOSS
MOSS
Initialize
Core
Common
DISK
Common
MOSSM
Application
Monitor
Procedure
Called
Called
Procedure
END
?
NO
Execute New
Procedure
YES
Return to Monitor

DESIGN PHILOSOPHY

- PROGRAMMING STANDARDS -

COMMON...

K = FIXED POINT

Q = REAL

VARIABLES...

I = INDEXES

L = LENGTHS

J = REGIONS

PROCEDURE FLOW

CALL DOWN LIST
(THRU INPUT)

MOSS

INPUT

- OVER-WRITE CALL DOWN LIST

CNV1

CNV2

- OVER-WRITE CALL DOWN LIST

CLEAN

SORT

SORT3

- OVER-WRITE CALL DOWN LIST

CLN2

SORT

SORT3

- OVER-WRITE CALL DOWN LIST

Etc.

CALL EXECU (LINK1, LINKL, LEVEL, KREPT)*

GO TO jjjjj*

GO TO kkkkk*

LINK1 - Position in Call Link Array of First Program to be Moved [(3-4n) where n is control Call Link]

LINKL - Position in Call Link Array of Last Program to be Moved [-(1+4M) where M is Last Link in Array]

LEVEL - Recall Indicator [if Level = 0 must return to main]

KREPT - Provides Re-entry control [Used in Computed GO TO]

jjjjj - Statement Number of First Call Link in ARRAY

kkkkk - Statement Number of Statement Following Control Call Link

- READ-WRITE DISK -

DREAD (NREC, J)

Read a Disk Record (NREC) into Region (J) where

NREC = 2048 * J + Record Number

DWRIT (NREC, J)

Write a Disk Record (NREC) into Region (J) where

NREC = 2048 * J + Record Number

DREAD (KNDCT, J)

Read the First Dictionary Record

OTHER PROCESSING FILES

KMTX - - MATRIX FILE

1. Derived from SETUP
2. Scaled for Processing Accuracy
3. Contains:
   a) Explicit Equation Coefficients
   b) Implicit Row Coefficients

KVLST - - VARIABLE STATUS LIST FROM SETUP

1. Derived from SETUP
2. Contains:
   a) Name
   b) Type
   c) Matrix File Element Seq. Number
   d) Scale Factor
   e) UB and LB For Each Variable

OTHER COMPUTATION FILES

KETA - - ETA FILE

1. Formed by INVERT
2. Represents Inverse of Basis Variables
3. An ETA Matrix is added by Each Minor Iteration
4. On Each Major Inversion KETA is reduced back to Single Entry
   a) Minimize ETA File
   b) Increase Accuracy

KBETA - - BETA FILE

1. Formed by INVERT on Last Step
2. Contains:
   a) Post Inversion Activities of Basis
   b) Post Inversion Activities of Bounds
3. Values are Scaled

OTHER OUTPUT FILES

KSLTN - - SOLUTION FILE

1. Formed by LPSOLUTION

2. Contains:

   a) Status List
   b) Matrix File Objective Entry
   c) Basis Activity
   d) Reduced Cost

3. Values are Scaled

KIANL - - INTERMEDIATE ANALYSIS (LPANAL)

1. Used In Conjunction with KMTX and KVLST for LPANAL

2. Contains:

   a) Increase - Decrease Cost
   b) Basis
   c) Activity Increase - Decrease

3. Values are Scaled

KWRK1-9 - - WORK FILES

1. Used for Transient Data

2. Some Used by System, Some Not

TO RUN LPMOSS
WITH FORTRAN COMPILER AND
ROOM TO WORK IN

```
{ MONITOR
// DUP
* Define Void Assembler
* Define Fixed Area                    0100
* Storedata   CD   FX   LPMSS    1778
```

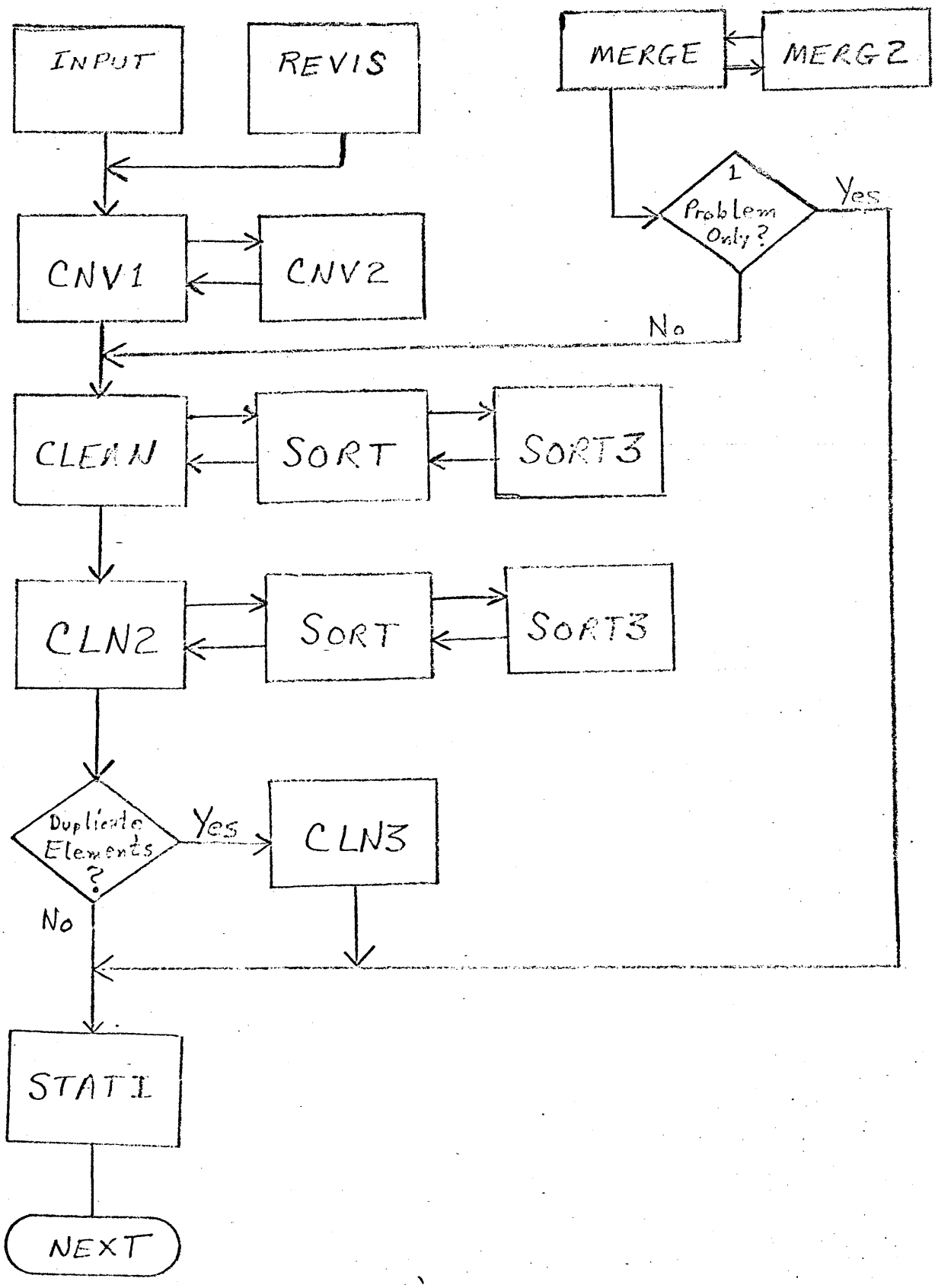
PROGRAMS STORED IN FIXED AREA

ON LPMOSS DISK

BZERO
CHECK
CLEAN
CLN2
CNV1
CNV2
CRASH
INPUT
INVER
INV22
INV44
INV55
INV66
LPSOL
MERGE
MERG2
MOSSM
MOVE
NEWBE
OPTIM
PIVOT
PRIMA
PRLCY
PRL44
REPOR
REVIS
RNG1P
RNG2P
SETBO
SETUP
SET2
SET3
SORT
SORT3
SOUL2
STATI

Procedure Flow

(use Push Down List. for Simultaneous Reference)



FLIP 2

REGION LAYOUT

KRGN (, J)

(KRGIN, J)	REGION INCREMENT
	~~ Not Used ~~
(KRGLN, J)	REGION LENGTH
(KTHRC, J)	THIS RECORD NUMBER
(KTHPY, J)	PREVIOUS RECORD NUMBER
(KTHNX, J)	NEXT RECORD NUMBER
(KFIID, J)	FILE I. D. (Not Used)
(KNOEL, J)	NUMBER OF ELEMENTS IN RECORD
(KFRPX, J)	INDEX OF 1st FIXED VARIABLE (In KRGN)
(KFRFL, J)	INDEX OF 1st REAL VARIABLE (In QRGN)
(KPXLN, J)	LENGTH OF FIXED POINT (WORDS)
(KFLLN, J)	LENGTH OF REAL VARIABLES (WORDS)

ELEMENT 1 – FIXED
ELEMENT 1 – REAL
ELEMENT 2 – FIXED
ELEMENT 2 – REAL
ELEMENT n – FIXED
ELEMENT n – REAL

FLIP 3

DICTIONARY

*PCF ADDRESS (N)	PROBLEM NAME (α)

SUBROUTINES (USING REGION):

DGET ~ GET PCF OF A NAMED PROBLEM.

DSAVE ~ SAVE A PROBLEM PCF and ENTER NAME IN DICTIONARY.

DDLTE ~ DELETE NAME and PCF FROM DICTIONARY.

DUPDT ~ DICTIONARY UPDATE:

```
CALL DDLTE
CALL DSAVE
RETURN
END
```

*PCF = PROBLEM CONTROL FIELD

FLIP 4

DATA MAINTENANCE

KPCFR FILE (PROBLEM CONTROL FILE)

(1,1)	RECORD NUMBER
(KVAIX,J)	HIGHEST INDEX USED THUS FAR
(1,NAME)	PROBLEM NAME (∝)
(,JFI)	FILE-CONTROL-FIELD VARS
(,JFI)	FILE - CONTROL - FIELD DATA

DATA MAINTENANCE FLIP 5

KPCFR FILE (F. C. F. SECTION)

(KFIST,JFI)	FILE STATUS (Now Open or Close)
(KFRRC,JFI)	FIRST RECORD LOCATION
(KLSRC,JFI)	LAST RECORD LOCATION
(KNXRC,JFI)	NUMBER OF RECORDS
(KEXFI,JFI)	*EXTERNAL FILE No.

*Now 1, SINCE 1130/PCS PRESENTLY ONLY SUPPORTS 1 DISK. (To be made variable to support multiple disks later.)

FLIP 6

DATA MAINTENANCE

KDATA FILE (GOES WITH KVARS, HOLDS DATA)

	SET INDEX (I)	VARS INDEX (I)	DATA TYPE (I)	DATA COEFF. (R)
*				
Word #:	1	2	3	4 5

ELEMENT LENGTH:
3 ONE-WORD INTEGERS +
2 WORDS FOR 1 REAL VAR.

SET INDEX REFERS to COLUMN ENTRY.

VARS INDEX REFERS to ROW ENTRY.

IFF. TWO SET ENTRIES ∴ DUPLICATE RECORD AND LAST USED.

FLIP 7

DATA MAINTENANCE

KVARS FILE (1 ENTRY / VARIABLE NAME)

	VARS INDEX (I)	VARIABLE NAME (4A2)	VARS TYPE (I)
*			
Word #	1	2 3 4 5	6

ELEMENT LENGTH:
6 FIXED POINT WORDS

AT INPUT A VARS INDEX IS ASSIGNED A SEQUENCE NO., REGARDLESS OF DUPLICATE ENTRIES. THIS DONE BY ROW. AFTER CLEAN-UP, DUPLICATES ARE REMOVED LEAVING ONE ENTRY FOR EACH X-Y CO-ORDINATE.

(See Flow Chart on "FLIP 1")

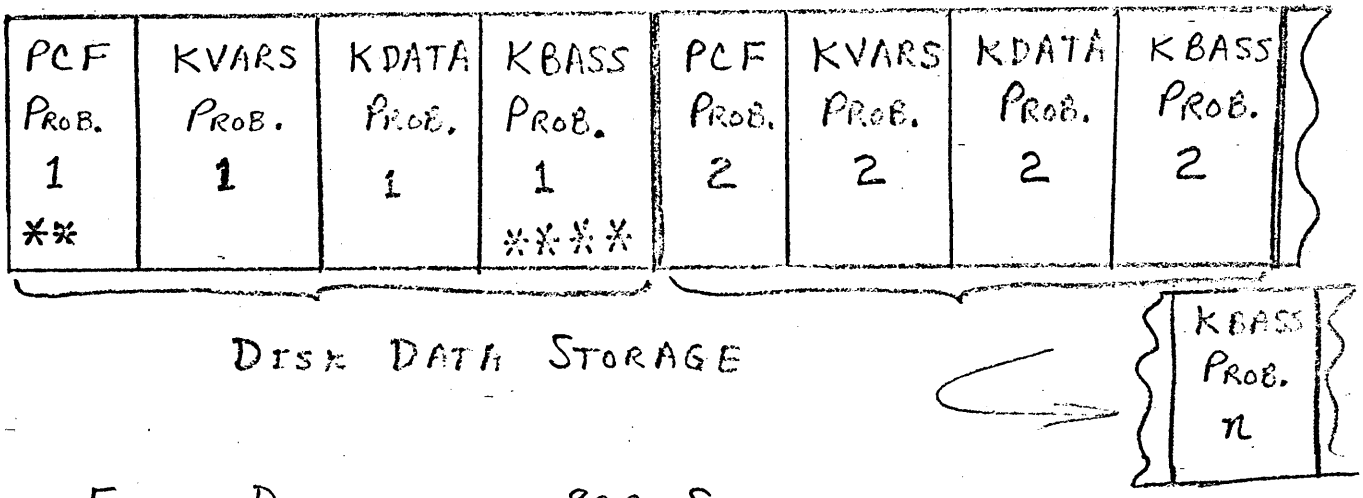
---

* (I) = I FORMAT ; (R) = FORMAT for REAL No. ; (4A2) = A FORMAT

DISK LAYOUT

MOSS COMMON	GET OFF RECD. ***	ALPHABET RECORDS (All of 1130 Set)	DICTIONARY RECORD *

STANDARD DISK HEADER



DISK DATA STORAGE

FILE DEFINED 800 SECTORS

* POINTS TO PROPER PCF.

** POINTS TO PROPER KVARS AND KDATA.

*** USED FOR RESTART PROCEDURE CALL.

**** STORES STATUS OF VARIABLE ON SAVESOLUTION PROCEDURE CALL.

CODE TO READ A FILE.

```
                ①      ②      ③      ④
      CALL INITR(JPCFR,NFILE,JFI,JFI1,JFIN,NDRIN)

10  IF (JFI) 100,100,20

                  ⑤    ⑥    ⑦
20  CALL GETDF(IFI,JFI,LFI,IFIL)        <- Fixed Point Read

25  IF (IFI-IFIL) 50,50,30               <- End of Record Test

                  ⑧
30  CALL DREAD (JFI,NDRIN)
    GO TO 10
50  Process Element
        ~~~~
    IFI = IFI + LFI
    GO TO 25
100 End of Job Routine
        ~~~
```

① Initialize Region & Get Logical Record
② PCF
③ Name of File Containing Index to PCF
④ Region No.
⑤ Fixed Index
⑥ Element length
⑦ Last Element
⑧ Read into Region JFI (No Physical IOCS if Already there).

JFI ~ Returned By Region Currently Used
IF JFI = Ø ∴ Region is Empty ∴ Physical EOJ Test

JFI1 ~ Sent As Region To Use First.

JFIN ~ Sent As Last Region To Use.

NDRIN ~ Indicates: Read Forward when (+1)
Read Backward when (-1)

441172

T.4.2[E]

ESSO RESEARCH LABORATORIES
HUMBLE OIL & REFINING COMPANY - BATON ROUGE REFINERY
BATON ROUGE, LOUISIANA

MEMORANDUM ON

Modification of IBM 1800 TSX to Support Six Disk Drives

DATE	BY	FILE NO.
December 4, 1967	E. H. Spencer	1407-1

INTRODUCTION

This is a report on work at the Esso Research Laboratories of Baton Rouge which added three more disks to the TSX system. The additional drives are treated in an identical fashion as the existing three drives. The new system has been tested for some of the options of TSX and all known bugs have been corrected. It is easy to convert an existing TSX Version 3 system into a six drive system. All that is required is a TASK assembly, the STOREMD of four subroutines, a partial system load, a define operation, and a skeleton rebuild. Although the method of distribution has not been decided, these changes are to be made available to other 1800 users.

DISCUSSION

The Esso Research Laboratories are located only two blocks from the main office building of the Humble Oil Refinery at Baton Rouge. Work at the laboratories is mostly process development. A small computing center is maintained in the laboratory. Through the years this has changed from a C.P.C.,to a 650, to a 1620, and will be an 1800 when program conversion is completed. Large computing jobs are sent to the refinery 360 system. A study of work done by the labs computer showed it to be data acquisition and batch processing of small jobs. This finding led to our 1800 order.

Logic of handling the acquired data indicated that some form of indexing and a random access to the files are desirable. Hence the choice of disks for mass storage. However, data volume is such that three 2310 disks are insufficient. Our system was ordered with five drives. A similar situation exists in the refinery where an 1800 computer is being used for direct digital control of a blending operation. Here the system was ordered with six disk drives. Under the press of two urgent applications, I began the job of modifying TSX to support the three additional disk drives. I was assisted by our I.B.M. representative, Mr. J. A. Albritton (S.E.).

I.B.M. was very cooperative and through Mr. Rob Martin of the Houston DACS Center we were able to get copies of 1401 microfiche tapes of the TSX system.

The work was started in May and completed in September. The modified system has been in use since October. There are no known bugs remaining in the six drive TSX. Although all of the options of TSX have not been tested such things as skeleton build, core load build, compiling of Fortran and assembler language programs, storing and deleting of core loads and relocatable programs, and the execution of process and non-process programs have been done.

Humble has consented to make these results available to other 1800 TSX users once the method of distribution has been selected. The Esso Labs would prefer not to act as agent for handling the distribution.

We started from TSX 3 Mod 1 with corrections through PTF 27. Appendix B shows the source level changes that we made to TSX. It should be noted that card numbers correspond to the microfiche tape for Mod 0. This is because only Mod 0 was available when we started. As Mod 1 and PTF's through 27 became available we added them to our Mod 0 source decks.

Our work was aided greatly by the fact that in some areas of the TSX programs provisions were made for future expansion to six drives. In particular, the Monitor and Disk Utility Programs (DUP) record and test the availability of drives by the use of three bits of a word in DCOM. Since a word has 16 bits, additional bits were available for three more drives. Even more important was the embedding of unused words in DCOM whenever information referring to a particular drive was stored. Without these provisions it would have been necessary to modify and enlarge DCOM. DCOM is the heart of TSX and is referenced absolutely and repeatedly throughout the routines. Furthermore several system programs end only a few words from the start of DCOM and enlargement would require moving them to lower core. Such moves would be difficult to organize in sections of EDP and DUP where an extensive overlay structure has been built. In fact, without the unused words of DCOM, it would not have been practical to change TSX for six drives and because of this it is not feasible to consider support of more than six drives.

It is difficult to convey the size of a job of this kind. It took over five months and involved twenty-seven TSX routines. Fourteen pages of source changes are listed in Appendix B. TSX logic in handling the disks was sufficiently general that only in a few cases was the logic changed and often, even then, just for a gain in efficiency. Most of the changes or additions were minor. But the problem of locating them was not necessarily small. It sometimes involved tracing back through several routines and to do this required a great deal of detail knowledge of the TSX programs. Because no courses on TSX internals are offered by I.B.M., we were forced to learn the system by reading programs.

Instructions outlining the procedure to be used in converting an existing three drive TSX system into a six drive system are given in Appendix A. It is briefly to (1) insert the change cards in source TASK and reassemble, (2) STOREMD the four Fortran I/Ø subroutines, (3) partial system load the changed routines redoing the assignments and the DEDIT, (4) define NDISK and CONFG, and (5) rebuild skeleton. The define and rebuild skeleton must be done with six drive TASK in core. The define CONFG and skeleton build are required because a new cold start and an error decision program have been stored. Word count and sector address of skeleton are stored to cold start by define and to EDP by skeleton builder.

CONCLUSION

Other potential users of our six drive modifications face two problems. They are the problems of distribution and maintenance of an IBM unsupported system. Corrections to TSX have been numerous and many are in the twenty-seven routines which we changed. If updated source programs are available, it is usually easy to make the corrections and recompile. However, most 1800 users do not have source cards, which are available only on magnetic tape.

EHS/gth

APPENDIX A

PROCEDURE FOR UPDATING

TO

TO SUPPORT SIX DRIVES

441172

ESSO RESEARCH LABORATORIES
HUMBLE OIL & REFINING COMPANY - BATON ROUGE REFINERY
BATON ROUGE, LOUISIANA

MEMORANDUM ON

Six Drive Changes to IBM 1800 TSX-Phase 2, 1800-Os-001, Version 3, Modification Level 1, through PTF 27

DATE	BY	FILE NO.
December 4, 1967	E. H. Spencer, J. A. Albritton (IBM)	

I. General Comments

A. These decks represent user modifications to TSX to support six disk drives. Three drives are standard and three are RPQ devices.

B. Area Codes of 20, 24, 25 and IAC codes of 20, 24, 25 are required.

C. The cards consist of source change cards for TASK and object decks for subroutines and for system routines. Control cards are included.

D. After the partial system load, user TASK is required. The system is now incompatible with Sys Gen Task.

E. Assignments must be made for the 3 additional disk drives. A DEDIT is required to clear FLET. A DEFINE CONFG and a skeleton rebuild are required.

F. Three additional fields are used on the cold start card (24, 26, 28) for dive identification and on the JOB (21-25, 26-30, 31-35) for label identification.

II. Procedure

A. Complete the additional TASK EQU's. Insert the change cards replacing cards with equal numbers and including others in the source TASK cards. Reassemble TASK.

B. Copy your system on another pack. Henceforth use the copy.

C. Do the DUP operation required to STOREMD for the four subroutines.

D. Do the partial system load. Redo the ASSIGNMENTS and DEDIT to initialize FLET. If the three additional disks have not been assigned, be sure to assign them. (From here on user's TASK from Step A is required).

E. Load user's TASK from cards. Do a DEFINE CONFG. (Ignore the error message which says TASK in core is not the same as TASK on disk).

F. Rebuild skeleton using the new SKA program and the Level 1 SKB program.

EHS,JAA/gth

470

MATERIAL LIST FOR
SIX DRIVE CHANGES TO I.B.M.
1800 TSX-PHASE 2, 1800-OS-001
VERSION 3, MODIFICATION LEVEL 1, THROUGH PTF 27

USER MODIFIED TO SUPPORT SIX DISK DRIVES
(Esso Research Laboratories, Humble Oil Company, Baton Rouge, La.
and I.B.M., Baton Rouge, La.)

I. WRITE UP OF PROCEDURE

II. CARDS

Item No.	Description	Quantity	Deck Ident. (Col. 73-75)
1	Task (Source change cards to be inserted in source task)	339	TSK
2	Supervisor	80	NPS
3	Core Load Builder	83	CLB
4	Cold Start	51	CLD
5	Disk Utilities (Total - 142 Cards)		
	(1) Control	18	DCT
	(2) Let/Flet Dump	17	LFD
	(3) Store Control Card	43	SCT
	(4) Delete	25	DEL
	(5) Dump-Disk to NPWS	19	DP1
	(6) Write address	9	DWR
	(7) Search Program Name Table	11	SRP
6	Error Programs (Total - 85 Cards)		
	(1) Control Program	38	EDP
	(2) Disk Error Program	11	DSK
	(3) Error Table	4	EUD
	(4) C.E.Routine A	14	CEA
	(5) C.E.Routine B	18	CEB
7	Skeleton Builder	60	SKA
8	Task Utilities		
	(1) Task Card to Disk	10	TRL
	(2) Task Disk to Card	12	TDD
	(3) Task Disk to Patch	7	TDP
	(4) Task Disk Duplication	6	TUP
	(5) Task Disk Load for Off-Line System	13	TDL
	(6) Task Write Address	33	TWA
9	Subroutines (Total - 31 Cards)		
	(1) F-I/O Busy Test	5	BT1
	(2) Disk F-I/O Busy Test	4	BT2
	(3) Disk F-I/O	17	MDI
	(4) Fortran Find Routine	5	MDN

APPENDIX B

SIX DRIVE SOURCE

LANGUAGE CHANGES

TO TSX

```
DORG3  EQU          0 IF THREE DISK DR, ELSE 1                    TSK00081
DORG4  EQU          0 IF FOUR  DISK DR, ELSE 1                    TSK00082
DORG5  EQU          0 IF FIVE  DISK DR, ELSE 1                    TSK00083
PRIL3  EQU          DISK DR 3 INTER. LEVEL * 0/23 *               TSK00084
PRIL4  EQU          DISK DR 4 INTER. LEVEL * 0/23 *               TSK00085
PRIL5  EQU          DISK DR 5 INTER. LEVEL * 0/23 *               TSK00086
       DC     20            ADDITIONS TO IAC TABLE                TSK01671
       DC     IN3                                                 TSK01672
       DC     24                                                  TSK01673
       DC     IN4                                                 TSK01674
       DC     25                                                  TSK01675
       DC     IN5                                                 TSK01676
TADD   DC     TADDS&6       DK DRV TAB ADD TABLE    187           TSK03790
LOGDR  DC     TADDS&1       ADD OF LOGICAL DRV TAB  188           TSK03800
PHYDR  DC     TADDS&7       ADD OF PHYSICAL DR TAB  189           TSK03810
       DC     TADDS&12      END OF PHYSICAL DR TAB  190           TSK03820
DRSUP  DC     5             CODE OF LAST DISK DRV   191           TSK03830
       DC     -6            MINUS MAX NO OF DRIVES  192           TSK03840
       DC     TADDS         FWA OF DEVICE TAB-1     193           TSK03850
TADDS  DC     *&5           DK DRV TAB ADD TABLE                  TSK03901
       DC     TA            LOGICAL DRIVE 0                       TSK03902
       DC     TA1*DORG1                                           TSK03903
       DC     TA2*DORG1*DORG2                                     TSK03904
       DC     TA3*DORG1*DORG2*DORG3                               TSK03905
       DC     TA4*DORG1*DORG2*DORG3*DORG4                         TSK03906
       DC     TA5*DORG1*DORG2*DORG3*DORG4*DORG5                   TSK03907
       DC     TA            PHYSICAL DRIVE 0                      TSK03908
       DC     TA1*DORG1                                           TSK03909
       DC     TA2*DORG1*DORG2                                     TSK0390A
       DC     TA3*DORG1*DORG2*DORG3                               TSK0390B
       DC     TA4*DORG1*DORG2*DORG3*DORG4                         TSK0390C
       DC     TA5*DORG1*DORG2*DORG3*DORG4*DORG5                   TSK0390D
CDRSP  EQU    DRSUP-CON     DISPLACEMENT  FROM CON                TSK04241
TA33   BSS  E  0                                                   TSK07390
FFA    EQU    TA33-TA2                                            TSK07400
       ORG    FFA*DORG2*DORG1&TA2                                 TSK07410
* INSERT THE FOLLOWING 222 CARDS AFTER TSK07410
*                                                                  ERL00001
* TABLE FOR DRIVE 3                                               ERL00002
*                                                                  ERL00003
TA3    DC     0             LAST IOCC  %AREA¤       0           ERL00004
       DC     /A000                    %CONTROL¤      1           ERL00005
       DC     0             NEXT IOCC  %AREA¤       2           ERL00006
       DC     /A000                    %CONTROL¤      3           ERL00007
       DC     0             LAST WD CT              4           ERL00008
       DC     0             LAST DK ADD             5           ERL00009
       DC     0             NEXT WD CT              6           ERL00010
       DC     0             NEXT DK ADD             7           ERL00011
       DC     0             SEEK IOCC  %AREA¤       8           ERL00012
       DC     /A400                    %CONTROL¤      9           ERL00013
       DC     0             SENSE IOCC & SEEK TO    10          ERL00014
       DC     /A700                    %CONTROL¤      11          ERL00015
       DC     TA3&36        CHECK TABLE IOCC %AREA¤   12          ERL00016
       DC     /A600                    %CONTROL¤      13          ERL00017
       DC     0             TOTAL WD CT LEFT          14          ERL00018
       DC     0           FIRST READ/WRITE            15          ERL00019
```

```
         DC    0          SAVE 1                          16 ERL00020
         DC    0                                          17 ERL00021
         DC    0          SAVE 2                          18 ERL00022
         DC    0                                          19 ERL00023
         DC    -2         VARIABLE &-2                    20 ERL00024
         DC    TA3&16     SAVE ADD                        21 ERL00025
         DC    TA3&17     SAVE ADD&1                      22 ERL00026
         DC    0          NEXT AREA ADD & 1               23 ERL00027
         DC    -8         ARM POSITION                    24 ERL00028
         DC    0          ERROR CT                        25 ERL00029
         DC    0          CALL INT LEVEL                  26 ERL00030
         DC    750        NONPROCESS WORK STORAGE         27 ERL00031
* INDICATOR TABLE          NEG-ON          EVEN-OFF           ERL00032
         DC    0          I1              I2              28 ERL00033
         DC    0          SEEK            WRITE IMM       29 ERL00034
         DC    0          CH TAB          CH TAB RET      30 ERL00035
         DC    0          SEEK FUN        READ            31 ERL00036
         DC    0          W/RBC           W/RBC IN OP     32 ERL00037
         DC    0            GENERAL INDICATOR             33 ERL00038
         DC    0          FILE PROT  FILE PRO SW          34 ERL00039
         DC    0          IND OP          FIN OLD OP      35 ERL00040
         DC    1          I/O AREA FOR CHECK TABLE        36 ERL00041
         DC    0                                          37 ERL00042
         DC    0          FIRST IOCC                      38 ERL00043
         DC    0                                          39 ERL00044
         DC    0          FIRST WD COUNT                  40 ERL00045
         DC    0          FIRST DK AD                     41 ERL00046
         DC    0          FIRST TOTAL WD CT LEFT          42 ERL00047
         DC    /A701      SENSE/RESET                     43 ERL00048
         DC    0          RELATIVE ARM POSITION           44 ERL00049
         DC    0          FILE SWITCH                     45 ERL00050
         DC    0          SKIP ARM POSITION CHECK         46 ERL00051
         DC    1600       NONPROC WK ST END ADD&1         47 ERL00052
         DC    0          NOT USED YET                    48 ERL00053
         DC    PRIL3      DISK INT LEVEL                  49 ERL00054
         DC    0          $IO SET INDICATOR               50 ERL00055
         DC    0          CALL ADDRESS                    51 ERL00056
         DC    *&1        TABLE READ IOCC                 52 ERL00057
         DC    /A600                                      53 ERL00058
*DISK DATA TABLE--SECTOR 0                                  ERL00059
         DC    11         TABLE WORD COUNT                54 ERL00060
         DC    0          SECTOR ADDRESS                  55 ERL00061
         DC    0          PACK LABLE                      56 ERL00062
         DC    0          FIRST SEC PRO WK STO            57 ERL00063
         DC    0          LAST&1 SEC PRO WK STO           58 ERL00064
         DC    0          MODIFICATION LEVEL              59 ERL00065
         DC    0          FIRST BAD CYL ADD               60 ERL00066
         DC    0          SECOND BAD CYL ADDR             61 ERL00067
         DC    0          THIRD BAD CYL ADDR              62 ERL00068
         DC    0          NOT USED YET                    63 ERL00069
         DC    0          NOT USED YET                    64 ERL00070
         DC    0          NOT USED YET                    65 ERL00071
TA44     BSS   E  0                                         ERL00072
FAC      EQU    TA44-TA3                                   ERL00073
         ORG    FAC*DORG3*DORG2*DORG1&TA3                   ERL00074
*                                                           ERL00075
* TABLE FOR DRIVE 4                                        ERL00076
```

TSX MODIFICATIONS TO SUPPORT
TSX DISK DRIVES
SOURCE LANGUAGE CHANGE CARDS

```
*                                                                  ERL00077
TA4     DC      0          LAST IOCC  %AREA□          0 ERL00078
        DC      /C000                 %CONTROL□       1 ERL00079
        DC      0          NEXT IOCC  %AREA□          2 ERL00080
        DC      /C000                 %CONTROL□       3 ERL00081
        DC      0          LAST WD CT                 4 ERL00082
        DC      0          LAST DK ADD                5 ERL00083
        DC      0          NEXT WD CT                 6 ERL00084
        DC      0          NEXT DK ADD                7 ERL00085
        DC      0          SEEK IOCC  %AREA□          8 ERL00086
        DC      /C400                 %CONTROL□       9 ERL00087
        DC      0          SENSE IOCC & SEEK TO      10 ERL00088
        DC      /C700                 %CONTROL□      11 ERL00089
        DC      TA4&36     CHECK TABLE IOCC %AREA□    12 ERL00090
        DC      /C600                 %CONTROL□      13 ERL00091
        DC      0          TOTAL WD CT LEFT           14 ERL00092
        DC      0          FIRST READ/WRITE           15 ERL00093
        DC      0          SAVE 1                     16 ERL00094
        DC      0                                     17 ERL00095
        DC      0          SAVE 2                     18 ERL00096
        DC      0                                     19 ERL00097
        DC      -2         VARIABLE &-2               20 ERL00098
        DC      TA4&16     SAVE ADD                   21 ERL00099
        DC      TA4&17     SAVE ADD&1                 22 ERL00100
        DC      0          NEXT AREA ADD & 1          23 ERL00101
        DC      -8         ARM POSITION               24 ERL00102
        DC      0          ERROR CT                   25 ERL00103
        DC      0          CALL INT LEVEL             26 ERL00104
        DC      750        NONPROCESS WORK STORAGE    27 ERL00105
* INDICATOR TABLE          NEG-ON        EVEN-OFF       ERL00106
        DC      0          I1            I2           28 ERL00107
        DC      0          SEEK          WRITE IMM    29 ERL00108
        DC      0          CH TAB        CH TAB RET   30 ERL00109
        DC      0          SEEK FUN      READ         31 ERL00110
        DC      0          W/RBC         W/RBC IN OP  32 ERL00111
        DC      0            GENERAL INDICATOR        33 ERL00112
        DC      0          FILE PROT     FILE PRO SW  34 ERL00113
        DC      0          IND OP        FIN OLD OP   35 ERL00114
        DC      1          I/O AREA FOR CHECK TABLE   36 ERL00115
        DC      0                                     37 ERL00116
        DC      0          FIRST IOCC                 38 ERL00117
        DC      0                                     39 ERL00118
        DC      0          FIRST WD COUNT             40 ERL00119
        DC      0          FIRST DK AD                41 ERL00120
        DC      0          FIRST TOTAL WD CT LEFT     42 ERL00121
        DC      /C701      SENSE/RESET                43 ERL00122
        DC      0          RELATIVE ARM POSITION      44 ERL00123
        DC      0          FILE SWITCH                45 ERL00124
        DC      0          SKIP ARM POSITION CHECK    46 ERL00125
        DC      1600       NONPROC WK ST END ADD&1    47 ERL00126
        DC      0          NOT USED YET               48 ERL00127
        DC      PRIL4      DISK INT LEVEL             49 ERL00128
        DC      0          $IO SET INDICATOR          50 ERL00129
        DC      0          CALL ADDRESS               51 ERL00130
        DC      *&1        TABLE READ IOCC            52 ERL00131
        DC      /C600                                 53 ERL00132
*DISK DATA TABLE--SECTOR 0                                 ERL00133
```

```
            DC      11          TABLE WORD COUNT                54 ERL00134
            DC      0           SECTOR ADDRESS                  55 ERL00135
            DC      0           PACK LABLE                      56 ERL00136
            DC      0           FIRST SEC PRO WK STO            57 ERL00137
            DC      0           LAST&1 SEC PRO WK STO           58 ERL00138
            DC      0           MODIFICATION LEVEL              59 ERL00139
            DC      0           FIRST BAD CYL ADD               60 ERL00140
            DC      0           SECOND BAD CYL ADDR             61 ERL00141
            DC      0           THIRD BAD CYL ADDR              62 ERL00142
            DC      0           NOT USED YET                    63 ERL00143
            DC      0           NOT USED YET                    64 ERL00144
            DC      0           NOT USED YET                    65 ERL00145
TA55        BSS  E  0                                           ERL00146
FAD         EQU     TA55-TA4                                    ERL00147
            ORG     FAD*DORG4*DORG3*DORG2*DORG1&TA4             ERL00148
*                                                               ERL00149
* TABLE FOR DRIVE 5                                            ERL00150
*                                                               ERL00151
TA5         DC      0           LAST IOCC  %AREA□       0       ERL00152
            DC      /C800                  %CONTROL□        1  ERL00153
            DC      0           NEXT IOCC  %AREA□       2       ERL00154
            DC      /C800                  %CONTROL□        3  ERL00155
            DC      0           LAST WD CT                  4   ERL00156
            DC      0           LAST DK ADD                 5   ERL00157
            DC      0           NEXT WD CT                  6   ERL00158
            DC      0           NEXT DK ADD                 7   ERL00159
            DC      0           SEEK IOCC  %AREA□       8       ERL00160
            DC      /CC00                  %CONTROL□        9  ERL00161
            DC      0           SENSE IOCC  & SEEK TO       10  ERL00162
            DC      /CF00                  %CONTROL□        11 ERL00163
            DC      TA5&36      CHECK TABLE IOCC %AREA□     12  ERL00164
            DC      /CE00                  %CONTROL□        13 ERL00165
            DC      0           TOTAL WD CT LEFT            14  ERL00166
            DC      0           FIRST READ/WRITE            15  ERL00167
            DC      0           SAVE 1                      16  ERL00168
            DC      0                                       17  ERL00169
            DC      0           SAVE 2                      18  ERL00170
            DC      0                                       19  ERL00171
            DC      -2          VARIABLE &-2                20  ERL00172
            DC      TA5&16      SAVE ADD                    21  ERL00173
            DC      TA5&17      SAVE ADD&1                  22  ERL00174
            DC      0           NEXT AREA ADD & 1           23  ERL00175
            DC      -8          ARM POSITION                24  ERL00176
            DC      0           ERROR CT                    25  ERL00177
            DC      0           CALL INT LEVEL              26  ERL00178
            DC      750         NONPROCESS WORK STORAGE     27  ERL00179
* INDICATOR TABLE               NEG-ON          EVEN-OFF        ERL00180
            DC      0           I1              I2          28  ERL00181
            DC      0           SEEK            WRITE IMM   29  ERL00182
            DC      0           CH TAB          CH TAB RET  30  ERL00183
            DC      0           SEEK FUN        READ        31  ERL00184
            DC      0           W/RBC           W/RBC IN OP 32  ERL00185
            DC      0             GENERAL INDICATOR         33  ERL00186
            DC      0           FILE PROT  FILE PRO SW      34  ERL00187
            DC      0           IND OP          FIN OLD OP  35  ERL00188
            DC      1           I/O AREA FOR CHECK TABLE    36  ERL00189
            DC      0                                       37  ERL00190
```

```
            DC      0           FIRST IOCC                      38 ERL00191
            DC      0                                           39 ERL00192
            DC      0           FIRST WD COUNT                  40 ERL00193
            DC      0           FIRST DK AD                     41 ERL00194
            DC      0           FIRST TOTAL WD CT LEFT          42 ERL00195
            DC      /CF01       SENSE/RESET                     43 ERL00196
            DC      0           RELATIVE ARM POSITION           44 ERL00197
            DC      0           FILE SWITCH                     45 ERL00198
            DC      0           SKIP ARM POSITION CHECK         46 ERL00199
            DC      1600        NONPROC WK ST END ADD&1         47 ERL00200
            DC      0           NOT USED YET                    48 ERL00201
            DC      PRIL5       DISK INT LEVEL                  49 ERL00202
            DC      0           $IO SET INDICATOR               50 ERL00203
            DC      0           CALL ADDRESS                    51 ERL00204
            DC      *&1         TABLE READ IOCC                 52 ERL00205
            DC      /CE00                                       53 ERL00206
*DISK       DATA TABLE--SECTOR 0                                   ERL00207
            DC      11          TABLE WORD COUNT                54 ERL00208
            DC      0           SECTOR ADDRESS                  55 ERL00209
            DC      0           PACK LABLE                      56 ERL00210
            DC      0           FIRST SEC PRO WK STO            57 ERL00211
            DC      0           LAST&1 SEC PRO WK STO           58 ERL00212
            DC      0           MODIFICATION LEVEL              59 ERL00213
            DC      0           FIRST BAD CYL ADD               60 ERL00214
            DC      0           SECOND BAD CYL ADDR             61 ERL00215
            DC      0           THIRD BAD CYL ADDR              62 ERL00216
            DC      0           NOT USED YET                    63 ERL00217
            DC      0           NOT USED YET                    64 ERL00218
            DC      0           NOT USED YET                    65 ERL00219
TA66        EQU     *                                              ERL00220
FAE         EQU     TA66-TA5                                       ERL00221
            ORG     FAE*DORG5*DORG4*DORG3*DORG2*DORG1&TA5          ERL00222
            S    X1 CDRSP       TEST FOR NO. OF DISKS           TSK10620
CEXTZ       LDX   1 CON                                         TSK11210
            XIO  X2 SNIOC                                       TSK11211
            LD   X1 CDRSP       STO NO. OF DRVS TO COUNT        TSK11370
            A    X1 D1                                          TSK11380
            STO    COUNT                                        TSK11390
            LDX  L1 TADDS&1                                     TSK11400
LOOP9       LD    1 0           LD DEVICE TAB ADDR              TSK11410
            BSC  L  *&5,&-      ZERO-BRANCH IF OFF LINE         TSK11420
            STO  L  *&1                                         TSK11430
BYSTA       LDX  L2 *-*         SET XR2 TO DEVICE TAB ADD       TSK11440
            BSI     BSYTS       TEST I/O AREA BUSY              TSK11450
            MDX   1 1           INCREMENT TO NEXT DEV TAB       TSK11460
            MDX  L  COUNT,-1    DECREMENT COUNTER               TSK11470
            MDX     LOOP9       LOOP FOR NEXT DRIVE             TSK11480
            LDX   1 CON         RESTORE XR1                     TSK11490
            BSC  L  CEXIT       AREA NOT FOUND-NOT BUSY         TSK11500
COUNT       DC      *-*                                         TSK11510
BSYTS       DC      0                                           TSK11520
            LD    2 FIOCC       ARE I/O AREAS THE SAME          TSK11530
            S     3 AREAD                                       TSK11540
            BSC  L  *&3,Z       NO - BRANCH TO TEST NEXT        TSK11550
            LD    2 IND1        YES- TEST FOR BUSY              TSK11551
            BSC  L  CEXTZ,Z     BUSY-BRANCH TO INT TEST         TSK11552
            BSC  I  BSYTS       RETURN                          TSK11553
```

TSX MODIFICATIONS TO SUPPORT
SIX DISK DRIVES
SOURCE LANGUAGE CHANGE CARDS

```
        MDX     GOON                                                TSK15531
IN3     LDX  L2 TA3                                                 TSK15532
        ORG     *-3&3*DORG1*DORG2*DORG3                             TSK15533
        MDX     GOON                                                TSK15534
IN4     LDX  L2 TA4                                                 TSK15535
        ORG     *-3&3*DORG1*DORG2*DORG3*DORG4                       TSK15536
        MDX     GOON                                                TSK15537
IN5     LDX  L2 TA5                                                 TSK15538
        ORG     *-3&3*DORG1*DORG2*DORG3*DORG4*DORG5                 TSK15539
        STO  X3 KYLEV&2                                             TSK25760
        MDX  L2 -DT5&1      DETERMINE WHICH KEYBOARD                TSK26060
        S    X1 CDRSP       TEST FOR LEGAL DRV CODE                 TSK39780
        S    X1 CDRSP       TEST FOR LEGAL DRV CODE                 TSK39840
        LD   L3 TADDS&12    GET PHY DEV TAB ADDR                    TSK39880
        STO  L2 TADDS&6     AND PUT IN LOGICAL TABLE                TSK39890
        DC      TADDS&1                                             TSK44040
        DC      /4000       DISK 3                                  TSK53111
        DC      /A700       SENSE IOCC - DRIVE 3                    TSK53112
        DC      Z1Z&6                                               TSK53113
        DC      20                                                  TSK53114
        ORG     *-4&4*DORG1*DORG2*DORG3                             TSK53115
        DC      /4000       DISK 4                                  TSK53116
        DC      /C700       SENSE IOCC - DRIVE 4                    TSK53117
        DC      Z1Z&7                                               TSK53118
        DC      24                                                  TSK53119
        ORG     *-4&4*DORG1*DORG2*DORG3*DORG4                       TSK5311A
        DC      /4000       DISK 5                                  TSK5311B
        DC      /CF00       SENSE IOCC - DRIVE 5                    TSK5311C
        DC      Z1Z&8                                               TSK5311D
        DC      25                                                  TSK5311E
        ORG     *-4&4*DORG1*DORG2*DORG3*DORG4*DORG5                 TSK5311F
        DC      IN3                                                 TSK54471
        DC      IN4                                                 TSK54472
        DC      IN5                                                 TSK54473
ASTAR   LDX  I3 TADDS&1     RESET DISK DEVICE TABLES                TSK55330
        LDX  I2 TADDS&2                                             TSK55520
        LDX  I2 TADDS&3                                             TSK55560
        LDX  I2 TADDS&4                                             TSK55591
        MDX   2 0                                                   TSK55592
        STO  X2 IND1                                                TSK55593
        ORG     *-4&4*DORG1*DORG2*DORG3                             TSK55594
        LDX  I2 TADDS&5                                             TSK55595
        MDX   2 0                                                   TSK55596
        STO  X2 IND1                                                TSK55597
        ORG     *-4&4*DORG1*DORG2*DORG3*DORG4                       TSK55598
        LDX  I2 TADDS&6                                             TSK55599
        MDX   2 0                                                   TSK5559A
        STO  X2 IND1                                                TSK5559B
        ORG     *-4&4*DORG1*DORG2*DORG3*DORG4*DORG5                 TSK5559C
IDK0    LDX  I2 DRSUP       XR2#DRIVE COUNT                         TSK56280
IDK0A   LD   L2 TADDS&1     GET DEV TABLE ADDRESS                   TSK56290
        STO  L2 TADDS&1     THAT DISK OFF-LINE                      TSK56460
        A     2 LOGDR-CON                                           TSK60590
* DELETE TSK60750
SKLLV   LDX  I2 TADDS&1     TEST TO SEE                             TSK61080
```

```
LDRCD LD    L  JBFLG      IF JOB FLAG IS ZERO SKIP     NPS05230
      A     L  188        ADDRESS OF DEVICE TABLE      NPS22110
* DELETE NPS22370
*DEAC MDX   L  7,-1                                     NPS26820
*     NOP                                              NPS26830
CDEAC LD    2  CDCON&2                                  NPS26840
      MDX      CDOUT                                    NPS26900

      LDX   2  6          NUMBER OF DISK DRIVES        CLB07770
      DC       /B000       DR.3                         CLB08291
      DC       /C000       DR.4                         CLB08292
      DC       /D000       DR.5                         CLB08293
      DC       /B000       DR.3                         CLB08371
      DC       /C000       DR.4                         CLB08372
      DC       /D000       DR.5                         CLB08373
      DC       0           DR.3                         CLB08441
      DC       0           DR.4                         CLB08442
      DC       0           DR.5                         CLB08443
      BSS      345        *THIS BSS MUST BE AT LEAST*   CLB21530
      BSC   L  IN010,-Z    BR. IF NON-PROCESS           CLB23110
      BSC   L  IN010,E     BR. IF NOT AN INTERRUPT      CLB23150
IN010 LDX   I1 192         NO. OF DRIVES                CLB23280
      LDX   I2 189                                      CLB23290
      STX   2  *&1                                      CLB23291
IN016 LD    L1 *-*         GET DEVICE TABLE POINTER     CLB23292
      OR    L1 WSOVF FLAG  BIT%B0#1¤,DRV,CD,%B1-B3¤     CLB23350
      STO   L1 WSOVF MAX,  RELATIVE DISK ADDRESS        CLB23360

EQU00 EQU      0                                        CLD00271
      LD    L  189                                      CLD03605
      STO   L  PRTCL                                    CLD03606
      SLA      16                                       CLD03607
      S     L  192                                      CLD03608
      STO   L  PRTCL&1                                  CLD03609
LDDBL LDD   I  STGPK                                    CLD03610
      LDX   I1 188         INITIALIZE START OF TABLE    CLD05750
      MDX   1  -1            OF ADDRESSES OF DEVICE     CLD05752
      STX   1  HMDRV&1       TABLES                     CLD05754
      LD    L  191         INITIALIZE DRIVE CODE        CLD05756
      SLA      12            NUMBER                     CLD05758
      STO   L  SA                                       CLD0575A
      LDX   I1 191         SET UP LOOP COUNTER          CLD0575C
      MDX   1  1                                        CLD0575E
HMDRV LD    L1 *-*         PICK UP DEVICE TBL. ADD.     CLD05760
      DC       188         ADDRESS WORDS 188-192        CLD14460
      DC       5           WORD COUNT                   CLD14470
PRTCL DC       *-*         ADDRESS WORDS-PHY DRV TAB    CLD14481
      DC       *-*         WORD COUNT                   CLD14482
*                                                       CLD14483
SA    DC       *-*         DRIVE CODE                   CLD14760
* REMOVE CLD15720 AND CLD15730
* REMOVE CLD17140 THRU CLD18330
NXTS9 LDX   I1 188         XR1#ADD OF LOG DRV TAB       CLD17140
      STX   1  SETDR&1                                  CLD17150
```

TSX MODIFICATIONS TO SUPPORT
SIX DISK DRIVES
SOURCE LANGUAGE CHANGE CARDS

```
         STX   1 BOGDW&1                                        CLD17160
         STX   1 NOTYT&1                                        CLD17170
         MDX   1 -1                                             CLD17180
         STX   1 NOTYT&5                                        CLD17190
         LDX  11 190                                            CLD17200
         MDX   1 1                                              CLD17210
         STX   1 SETDR&5                                        CLD17220
         LDX  L1 RDBUF&17   XR1 POINTS TO COL 18                CLD17230
         LDX   2 0          XR2 LOGICAL UNIT COUNTER            CLD17240
NEXCL    LDX   3 -6         XR3 IS COMPARE TAB POINTER          CLD17250
*                                                               CLD17260
* INITIAL THE LOGICAL DRIVES, ERROR NO. 10                      CLD17270
*                                                               CLD17280
REDOA    LD    1 EQU00      LOAD HOLLERITH DRIVE CODE           CLD17290
         EOR  L3 NUMBR&6    COMPARE TO TABLE ENTRY              CLD17300
         BSC  L  SETDR,&-   BRANCH ON EQUAL TO SET DR.          CLD17310
         MDX   3 1          NOT EQUAL, STEP DOWN TABLE          CLD17320
         MDX     REDOA      NOT AT END OF TABLE,REPEAT          CLD17330
         LD    1 EQU00      END OF TABLE,COMPARE BLANK          CLD17340
         EOR     NUMBR&6                                        CLD17350
         BSC  L  BLNAK,&-   EQUAL BRANCH TO TAKE OFF L          CLD17360
ERCRD    MDX  L  ERR10,1    NOT VALID CODE-NOT BLANK            CLD17370
         NOP                GO TO ERROR 10                      CLD17380
         BSC  L  BADCD                                          CLD17390
*                                                               CLD17400
* STORE SELECTED DEVICE TABLE ADDRESS TO LOGICAL DR             CLD17410
*                                                               CLD17420
SETDR    LD   L2 *-*                                           CLD17430
         BSC  L  ERCRD,&-   ERROR IF NOT ON LINE                CLD17440
         LD   L3 *-*        GET DEVICE TABLE ADDRESSES          CLD17450
*                            FROM PHYSICAL UNIT TABLE           CLD17460
BOGDW    STO  L2 *-*        STORE TO LOGICAL UNIT               CLD17470
         MDX   1 2          BUMP TO NEXT CARD COLUMN            CLD17480
         MDX   2 -4         TEST FOR LAST LOGICAL UNIT          CLD17490
         MDX     *&3        LAST UNIT,GO TO CHECK DUPL          CLD17500
         MDX   2 5                                              CLD17510
         NOP                                                    CLD17520
         MDX     NEXCL      PROCESS NEXT COLUMN                 CLD17530
*                                                               CLD17540
* TEST FOR DUPLICATION IN LOGICAL DRIVE CONFGURAT-              CLD17550
*       ION PUNCHED IN THE CARD.                                CLD17560
*                                                               CLD17570
         LDX   1 5          SET COUNTER FOR OUTER LOOP          CLD17580
CHAR1    STX   1 CONEG                                          CLD17590
         LDX  I2 CONFG      SET COUNTER FOR INNER LOOP          CLD17600
         MDX   2 -1                                             CLD17610
         NOP                                                    CLD17611
NOTYT    LD   L1 *-*        LOAD TEST WORD                      CLD17620
         BSC  L  LILJO,&-   IF ZERO, SKIP COMPARE               CLD17630
         EOR  L2 *-*        COMPARE                             CLD17640
         BSC  L  ERCRD,&-   IF EQUAL, BRANCH TO ERROR           CLD17650
         MDX   2 -1         BUMP COMPARE COUNTER                CLD17660
         MDX     NOTYT      BRANCH FOR NEXT COMPARE             CLD17670
LILJO    MDX   1 -1         NO EQUAL, MOVE TO NEXT              CLD17680
         MDX     CHAR1                                          CLD17690
         MDX     BNCRT                                          CLD17700
*                                                               CLD17710
```

```
* PUT THIS DISK DRIVE OFF LINE                                  CLD17720
*                                                              CLD17730
BLNAK SRA     16          ZERO THE ACCUMULATOR                 CLD17740
      MDX     BOGDW                                            CLD17750
*                                                              CLD17760
*                                                              CLD17770
*                                                              CLD17780
BNCRT LDX  I1 SVXR1       RESTORE INDEX REGISTERS              CLD17790
      LDX  I2 SVXR2                                            CLD17800
      LDX  I3 SVXR3                                            CLD17810
      BSC  I  INTPT       RETURN                               CLD17820
*                                                              CLD17830
CONFG DC      *-*                                              CLD17840
NUMBR DC      /2000         0                                  CLD17850
      DC      /1000         1                                  CLD17860
      DC      /0800         2                                  CLD17870
      DC      /0400         3                                  CLD17880
      DC      /0200         4                                  CLD17890
      DC      /0100         5                                  CLD17900
      DC      /0000         BLANK                              CLD17910
      STX  L3 SVZR3&1                                          CLD23401
      LDX  I1 188           XR1 POINTS TO LOG DEV TAB          CLD23440
* REMOVE CLD23640 THRU CLD23840
      LDX  3  0             INIT DRV CODE TO ZERO              CLD23640
      STX  3  DRIVE                                            CLD23650
      LDX  I3 191                                              CLD23660
      MDX  3  1             XR3 IS LOOP MONITOR                CLD23670
DROV2 LD   1  0             LD DEVICE TABLE ADDRESS            CLD23680
      BSC     Z             SKIP SEARCH IF ZERO                CLD23690
      BSI     SEARC         NOT ZERO GO TO SEARCH              CLD23700
      LD      DRIVE         NOT FOUND, INCREASE DR.CODE        CLD23710
      A    2  D1000-MDX03                                      CLD23720
      STO     DRIVE                                            CLD23730
      MDX  1  1             INCREASE DEVICE LIST ADDR.         CLD23740
      MDX  3  -1            TEST FOR LAST LOGICAL DRV.         CLD23750
      MDX     DROV2         NOT LAST, CONTINUE SEARCH          CLD23760
SVZR3 LDX  L3 *-*                                              CLD24131

      LD   L1 ENDLZ&COMMA                                      DCT09110
      MDX  I1 DLOGO                                            DCT09120
      STX  1  *&1                                              DCT09130
      LDX  I3 *-*                                              DCT09140
      SRA     4                                                DCT09150
      STO  3  NPSCC                                            DCT09160
      LD   2  NPWST                                            DCT09170
      SRA     4                                                DCT09180
      STO  3  EFFHM                                            DCT09190

*                                                              LFD04141
      LD   1  9                                                LFD04142
      S    2  7             CMP FOR .3 .%DR 3□                 LFD04143
      BSC  L  LET3,&-                                          LFD04144
*                                                              LFD04145
      LD   1  9                                                LFD04146
      S    2  8             CMP FOR .4 .%DR 4□                 LFD04147
      BSC  L  LET4,&-                                          LFD04148
```

TSX MODIFICATIONS TO SUPPORT
SIX DISK DRIVES
SOURCE LANGUAGE CHANGE CARDS

```
*                                                          LFD04149
        LD    1 9                                          LFD0414A
        S     2 9          CMP FOR .5 .%DR 5¤              LFD0414B
        BSC  L LET5,&-                                     LFD0414C
        MDX    START-2                                     LFD04271
LET3    LD     C3           DRIVE 3                        LFD04272
        STO    DRV                                         LFD04273
        MDX    START-2                                     LFD04274
LET4    LD     C4           DRIVE 4                        LFD04275
        STO    DRV                                         LFD04276
        MDX    START-2                                     LFD04277
LET5    LD     C5           DRIVE 5                        LFD04278
        STO    DRV                                         LFD04279
        S      C1                                          LFD04470
        BSC  L LTDR2,&-     DRIVE 2 REQUESTED              LFD04472
*                                                          LFD04474
        S      C1                                          LFD04476
        BSC    LTDR3,&-     DRIVE 3 REQUESTED              LFD04478
*                                                          LFD04480
        S      C1                                          LFD04482
        BSC    LTDR4,&-     DRIVE 4 REQUESTED              LFD04484
*                                                          LFD04486
        LD     PR5CD        DRIVE 5 REQUESTED              LFD04488
        MDX    LET0&2                                      LFD0448A
LTDR2   LD     DR2CD                                       LFD04511
        MDX    LET0&2                                      LFD04512
LTDR3   LD     DR3CP                                       LFD04513
        MDX    LET0&2                                      LFD04514
LTDR4   LD     DR4CD                                       LFD04515
        MDX    LET0&2                                      LFD04516
DR3CD   DC     /3001        LET SAD DR 3                   LFD04611
DR4CD   DC     /4001        LET SAD DR 4                   LFD04612
DR5CD   DC     /5001        LET SAD DR 5                   LFD04613
C3      DC     3                                           LFD04651
C5      DC     5                                           LFD04661
        DC     /F340        .3 .                           LFD07941
        DC     /F440        .4 .                           LFD07942
        DC     /F540        .5 .                           LFD07943


        A    L DLOGO                                       SCT08170
        STO    *&1                                         SCT08180
        LDX   I2 *-*        AT XEQ TIME,ADDR OF INST       SCT08190
*REMOVE SCT08200
        MDX   I1 DLOGO                                     SCT09660
        STX    1 *&1                                       SCT09670
        LDX   I2 *-*                                       SCT09680
        LD    L1 UADBZ&C    NOT SAME DISK,SO NPWS          SCT09820
        MDX   I1 DLOGO                                     SCT09830
        STX    1 *&1                                       SCT09840
        LDX   I2 *-*        UPDATED                        SCT09850
        A    L DLOGO                                       SCT15380
        STO    *&1                                         SCT15390
        LDX   I2 *-*                                       SCT15400
*REMOVE SCT15410 AND SCT15420
```

```
         DC    ONECN                                          DEL06251
         DC    ONECN                                          DEL06252
         DC    ONECN                                          DEL06253

GWSAD    LDX   I3 DRNUM    XR3 CONTAINS DR CODE               DP106430
         MDX   I3 DLOGO                                       DP106440
         STX    3 *&1      DR WRK LVL ADDR                    DP106450
         LDX   I3 *-*                                         DP106470

         LDX   I2 DLOGO                                       DWR06470

         SLT    1                                             SRP04760
         RTE    17                                            SRP04770

         LD    L  191      LOAD MAXDRIVE CODE                 EDP14941
         SLA      12          MOVE TO DRIVE POSITION         EDP14942
         STO   L  SA           STORE TO I/O AREA             EDP14943
         SLA      4        ZERO THE ACCUM                     EDP14944
         S     L  192      LOAD NO. OF DRIVES                 EDP14945
         OR       LDXIN       OR INTO LDX INSTRUCTION        EDP14946
         STO   L  DKDEV        STORE TO PROGRAM              EDP14947
         STO   L  RPLDK        STORE TO PROGRAM              EDP14948
         STO   L  STODK-1      STORE TO PROGRAM              EDP14949
         LD    L  193      STORE DEVICE TABLE ADD-1           EDP1494A
         STO   L  DKLP&1       TO PROGRAM                    EDP1494B
         STO   L  STODK&1                                     EDP1494C
         A        APTP2                                       EDP1494D
         STO   L  RPLND&1                                     EDP1494E
LDXIN    LDX    1 *-*      BLANK LOAD INSTRUCTION             EDP15311
APTP2    DC       APT&2    START OF I/O AREA PLUS 2           EDP15312
*                                                            EDP15313
LDXIN    LDX    1 *-*      BLANK LOAD INSTRUCTION             EDP15381
APTP2    DC       APT&2    START OF I/O AREA PLUS 2           EDP15382
*                                                            EDP15383
DKDEV    LDX    1 *-*                                         EDP09510
DKLP     LD    L1 *-*      SAVE USERS 2310 ADDRESS            EDP09520
RPLDK    LDX    1 *-*                                         EDP10140
RPLND    STO   L1 *-*      RESTORE THE ENTRY                  EDP10160
         LDX    1 *-*      XR1 # NO. OF DRIVES                EDP10610
STODK    LD    L1 *-*                                        EDP10620
SA       DC       *-*      DRIVE NUMBER                       EDP12080

         BSC      -          IS AREA CODE LESS THAN 16       DSK01040
         MDX      PSTVE      YES                             DSK01042
         LDX    1 5          NO,XR1#5 IF AREA CODE#25        DSK01043
         MDX   L  GETAB&1,11 SET UP FOR ERR CNTR IND         DSK01044
         MDX      *&1                                         DSK01046
PSTVE    LDX    1 2          XR1#2 IF AREA CODE IS 9         DSK01048
         SRA      11         POSITION AREA CODE AT RT        DSK01050
         STO      AC         SAVE HEX AREA CODE              DSK01055
         BSC      E          IS AREA CODE 9 OR 25            DSK01060
         MDX      GETAC      YES                             DSK01070
```

TSX MODIFICATIONS TO SUPPORT
SIX DISK DRIVES
SOURCE LANGUAGE CHANGE CARDS

```
         MDX    1 -1          NO.SET XR1 FOR DR CD#1 OR4 DSK01080
         SRA      2                                       DSK01090
         BSC      E           IS AREA CODE 8 OR 24        DSK01100
         MDX    1 -1          NO.SET XR1 FOR DR CD#0 OR3 DSK01110
         NOP                  YES                         DSK01115
GETAC    STX    1 CVEDP       SAVE DRIVE CODE             DSK01120
         MDX    1 12          SET UP FOR ERR CNTR IND     DSK01125
         LD       CVEDP       LOAD DRIVE CODE             DSK01130
         SLA      12          SET UP DRIVE CODE FOR 2ND   DSK01135
         STO  L   NUMER&1       WORD OF DISK IOCC         DSK01140
         LD       AC          GET HEX AREA CODE           DSK01145
         SRT      4           CONVERT TO EBCDIC           DSK01150
         SLA      5             AREA                      DSK01155
         SLT      4             CODE                      DSK01160
         OR       HF0F0                                   DSK01160
         STO  L   PAREA&6     STORE IN PRINT AREA         DSK01170
GETAB    MDX   L1 *-*         MODIFY XR1 FOR ERR CNTR     DSK01175
         STX    1 CNTR          IND AND STORE IT          DSK01180
HF0F0    DC       /F0F0       EBCDIC MASK                 DSK01565
*REMOVE DSK01730 THRU DSK01800
*DELETE DSK 02290 AND DSK02300
CNTR     DC       0           CONTAINS CNTR NO FOR        DSK02525
*                               HARDWARE ERROR           DSK02526
         STO      PAREA&9       CORELOAD NAME             DSK03590
         STO      CNTR        ZERO ERROR CNTR INDICATOR   DSK03780
         LD       CNTR                                    DSK03850
*REMOVE DSK02830 THRU DSK02850 AND DSK02890
*REMOVE DSK03200 THRU DSK03220 AND DSK03260
ERR6     BSI  I   DISKN       SEEK HOME ON DRIVE TO       DSK02860
ERR4     BSI  I   DISKN       SEEK HOME ON DRIVE TO       DSK03230


         DC       0           4TH  2310 DISK STORAGE     EUD00471
         DC       0           5TH  2310 DISK STORAGE     EUD00472
         DC       0           6TH  2310 DISK STORAGE     EUD00473
         BSS      2           DUMMY LOCATIONS            EUD00480


         LDX    1 28          LENGTH OF ERR CNTR TABLE    CEA01000
         LDX   L2 TOUT-1        MESSAGE BUFFER            CEA01010
         EBC      .2310  20  .                            CEA02011
         EBC      .2310  24  .                            CEA02012
         EBC      .2310  25  .                            CEA02013


ERCT     EQU      /FEF0       OUTPUT AREA                 CEB00540
         LDX   L1 114         TO BE PRINTED OUT           CEB00670
         LDX    1 6             MAX NO OF UNITS THIS DEV  CEB01210
         LDX    I1 193                                    CEB01250
         LDX    I2 187                                    CEB01260
         LDX     2 6                                      CEB01750
         LDX   L1 BUFF&8                                  CEB01760
         LDX    I1 187                                    CEB01780
         LDX    I1 193                                    CEB01810
         LDX     2 7           NO. OF LINES OF OUTPUT     CEB01840
         S        K6                                      CEB02760
         S        K2                                      CEB02780
```

TSX MODIFICATIONS TO SUPPORT
SIX DISK DRIVES
SOURCE LANGUAGE CHANGE CARDS

```
K2      DC      2           TEST IF 1053                    CEB02920
K6      DC      6           TEST IF 2310                    CEB02960
        S       K6                                          CEB03840
        LDX  I1 193                                         CEB03951
        STX   1 *&2                                         CEB03952
        LDX   1 6           SET TO MAX NUMBER               CEB03960
HSEEK   LD   I1 *-*         TEST IF DRIVE ON LINE           CEB03970
        S    L  K6                                          CEB04100
DIV     DC      /1000       USED TO CALC NEXT DRIVE         CEB04191
SA      DC      /5000                                       CEB04220


        DC      /3000                                       SKA04000
        DC      /4000                                       SKA04010
        DC      /5000                                       SKA04020
        A    L  188         ADDRESS OF DEVICE TABLE         SKA16140
*REMOVE SKA16400

        LDX  I1 188         LOAD ADDRESS OF                 TRL01580
        STX   1 *&1         DRIVE ZERO                      TRL01581
        LDX  I1 *-*         DEVICE TABLE                    TRL01582

        LDX  I1 188         GET FILE PROTECT STATUS         TDD01290
        STX   1 *&1                                         TDD01291
        LDX  I1 *-*                                         TDD01292

        MDX   2 -5          BRANCH BACK TO ST1 IF           TDP00810
        MDX     ST1         DRIVE CODE ILLEGAL              TDP00820
        LDX  I1 187         GET LOGICAL DRIVE               TDP00821
        STX   1 *-1         DEVICE TABLE                    TDP00822
        LD   L2 *-*         ADDRESS                         TDP00830
        LD   L  DCON&1                                      TDP01630

        LDX  I2 188         LOAD ADDRESS OF                 TUP00480
        STX   2 *&1         DEVICE                          TUP00481
        LDX  I2 *-*         TABLE                           TUP00482
        LDX  I3 188         SAVE BAD CYLINDER               TUP00840
        MDX   3 1           ADDRESS                         TUP00841
        STX   3 *&1         ON                              TUP00842
        LDX  I3 *-*         THIS                            TUP00843

        S    L  191         AND BRANCH TO ERROR             TDL01890
        LDX  I1 188                                         TDL01985
        STX   1 *&1                                         TDL01986
        LD   L2 *-*         BRANCH TO ERROR ROUTINE         TDL01990
        AND  L  H00FF                                       TDL02340

MESS3   EBC     . DRIVE CODES--HEX 0 1 2 3 4 5 .            TWA00450
        S    L  191         MUST BE 0 1 2 3 4 5             TWA00850
        A    L  187         GET DISKN DEVICE TABLE ADD      TWA00865
        LD   X1 0                                           TWA00890
```

TSX MODIFICATIONS TO SUPPORT
SIX DISK DRIVES
SOURCE LANGUAGE CHANGE CARDS

```
         DC      /1000       2310-3                  BT100731
         DC      /A700                               BT100732
         DC      /1000       2310-4                  BT100733
         DC      /C700                               BT100734
         DC      /1000       2310-5                  BT100735
         DC      /CF00                               BT100736

         DC      /1000       2310-3                  BT200281
         DC      /A700                               BT200282
         DC      /1000       2310-4                  BT200283
         DC      /C700                               BT200284
         DC      /1000       2310-5                  BT200285
         DC      /CF00                               BT200286

         A   L   188         *       *               MD102590
*REMOVE MD102800

         A   L   188         *       *               MDN00440
*REMOVE MDN00970
```

OS/360 FORTRAN REREAD

A. Known Restrictions

1. In its present form, REREAD did not work for a program which the E-level compiler suggested subdividing. We got around this by

```
MAINLINE:   CALL RR
            Etc.

SUBPROGRAM: SUBROUTINE RR
            CALL REREAD
            RETURN
            END
```

This apparently let FORTRAN handle linkage conventions for a module small enough that use and nonrestore of register 8 by REREAD did not hurt us.

2. We have not tried it (REREAD) with data sets organized as direct data sets. Author said it will not work.

3. REREAD will not work with a variable for DSRN:

```
   READ (J, 3) list
 3 FORMAT (etc.)
```

is not possible.

4. As distributed, REREAD is for an E-level system. If your system includes G- or H-level, or is restricted to G- or H-level, statement 3 of the Assembly Language source code should be changed to

```
EXTRN IHCFCOMH
```

5. As distributed, REREAD uses DSRN 19 for REREADing. To choose another DSRN (≤99), change statement 44 of the Assembly Language listing (as distributed) to

```
I19 DC F'xx'
```

where xx is the DSRN you desire to use. Author says this number is independent of number of DSRN's chosen at SYSGEN time, and that a DD DUMMY card does not have to be included.

B. Use

1. It is necessary to issue the statement

```
CALL REREAD
```

once and only once for each FORTRAN program in which REREADing is to occur. This call must be executed before any REREADing takes place.

2. After B.1 has occurred, REREADing is accomplished by

```
READ (19, f) list
```

where f is the FORMAT under which REREADing is to occur.

3. Any number of

```
READ (19, f1) list1
READ (19, f2) list2
...
READ (19, fn) listn
```

can occur for the same buffer load.

4. An application: User reads cards which occur randomly with respect to type, but each card contains its type identification in the same field (say, Col. 56). Let DSRN = 1 stand for the system reader. Then

```
      READ (1, 11) JTYPE
   11 FORMAT (55X, I1)
      GO TO (a1, a2, ..., an) JTYPE
   a1 READ (19, f1) list1
      GO TO b1
   a2 READ (19, f2) list2
      GO TO b2
      ...
   an READ (19, fn) listn
      GO TO bn
```

will accomplish this.

C. Function

CALL REREAD

substitutes the address of ZZZ for the address of FIOCS in the FORTRAN I/O area.

Each subsequent

READ (n, f) list

passes through ZZZ, where the pointers to buffers are saved and it is examined for DSRN = 19. If DSRN ≠ 19, then the control returns immediately to FORTRAN I/O for a physical read. If DSRN = 19, then the buffers are restored to their condition for the previous READ, physical reading is bypassed, and control is returned to FORTRAN I/O for FORMATting only.

D. Author of Program: Tom Vaden
IBM
Baton Rouge, Louisiana

E. Presented by a Successful User: Wade A. Norton
Chief Analyst, Computer Group
Southern Services, Inc.
Birmingham, Alabama 35202
COMMON 1125

F. This paper was presented to the OS Committee during both the Cincinnati and San Francisco meetings, because the audience was substantially different at each.

G. Listing of a D.O.S. version of the REREAD subroutine was distributed to the D.O.S. committee at a Cincinnati session. A copy of this listing is included as Page 17 of this paper. It was supplied by Jim Miller of IBM Birmingham who adapted it for use with D.O.S. Pete Haswell of Southern Natural Gas (1163) plans to use REREAD with D.O.S. FORTRAN.

Rust's version of REREAD as supplied by Max Allen 5 Sept 71

190

```
//REREAD JOB MSGLEVEL=1
//STEP1 EXEC ASMFCL,PARM.LKED=(MAP,LET,NCAL,DC)
//COMP      EXEC PGM=IEUASM                                          00000010
//SYSLIB    DD  DSNAME=SYS1.MACLIB,DISP=OLD                          00000020
//SYSUT1 DD UNIT=DISK,SPACE=(1700,(400,50)),DISP=(NEW,PASS)          00000030
//SYSUT3    DD  UNIT=DISK,SPACE=(1700,(400,50))                      00000040
//SYSUT2    DD  UNIT=(DISK,SEP=(SYSUT1,SYSUT3)),SPACE=(1700,(400,50))  00000050
//SYSPRINT DD  SYSOUT=A                                              00000060
//SYSPUNCH DD  DSNAME=&LOADSET,UNIT=DISK,SPACE=(400,(50,10)),       X00000070
//             DISP=(MOD,PASS),DCB=(,BLKSIZE=400)                    00000080
//COMP.SYSIN DD DATA
IEF236I ALLOC. FOR REREAD   COMP      STEP1
IEF237I SYSLIB    ON 190
IEF237I SYSUT1    ON 190
IEF237I SYSUT3    ON 190
IEF237I SYSUT2    ON 191
IEF237I SYSPUNCH ON 191
IEF237I SYSIN     ON 00C
```

SYMBOL	TYPE	ID	ADDR	LENGTH	LD ID
	PC	01	000000	000002	
REREAD	LD		000000		01
IHCFCOME	ER	02			
IHCFICSH	ER	03			

```
 LOC   OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                F01JAN67  10/29/67

000000                              1          START 0                                              061967WN
                                    2          ENTRY REREAD                                         061967WN
                                    3          EXTRN IHCFCOME                                       061967WN
                                    4          EXTRN IHCFIOSH                                       061967WN
000000 188F                         5 REREAD   LR    8,15                                           061967WN
000000                              6          USING REREAD,8                                       061967WN
000002 5010 80C8            000C8   7          ST    1,SV1                                          061967WN
000006 5810 8070            00070   8          L     1,ADZZZ                                        061967WN
00000A 58F0 8074            00074   9          L     15,ADIBCOM                                     061967WN
00000E 9250 F04A      0004A        10          MVI   74(15),X'50'                                   061967WN
000012 440F 004A            0004A  11          EX    0,74(15)                                       061967WN
000016 9258 F04A      0004A        12          MVI   74(15),X'58'                                   061967WN
00001A 5810 80C8            000C8  13          L     1,SV1                                          061967WN
00001E 07FF                        14          BCR   15,14                                          061967WN
000020                             15          USING *,1                                            061967WN
000020 904F 1058            00078  16 ZZZ       STM   4,15,SAVE                                    061967WN
                                   17          DROP  1                                              061967WN
000024 1881                        18          LR    11,1                                           061967WN
000020                             19          USING ZZZ,11                                         061967WN
000026 1840                        20          LR    4,0                                            061967WN
000028 D501 4000 B0B0 00000 000D0  21          CLC   0(2,4),HOOFO                                   061967WN
00002E 4780 B01C            0003C  22          BC    8,CHK19
000032 5810 80A0            000C0  23          L     1,ADFIOCS                                      061967WN
000036 984F B058            00078  24          LM    4,15,SAVE                                      061967WN
00003A 07F1                        25          BCR   15,1                                           061967WN
00003C D503 E000 B0A4 00000 000C4  26 CHK19    CLC   0(4,14),I19
000042 4780 B03C            0005C  27          BC    8,BYPASS                                       061967WN
000046 182E                        28          LR    2,14                                           061967WN
000048 5810 80A0            000C0  29          L     1,ADFIOCS                                      061967WN
00004C 0501                        30          BALR  0,1                                            061967WN
00004E 00F0                        31          DC    X'00F0'                                        061967WN
000050 5020 80A8            000C8  32          ST    2,SV1                                          061967WN
000054 5030 80AC            000CC  33          ST    3,SV2                                          061967WN
000058 47F0 8044            00064  34          BC    15,LEAVE                                       061967WN
00005C 5820 80A8            000C8  35 BYPASS   L     2,SV1                                          061967WN
000060 5830 80AC            000CC  36          L     3,SV2                                          061967WN
000064 4114 0002            00002  37 LEAVE    LA    1,2(4)                                         061967WN
000068 984F B058            00078  38          LM    4,15,SAVE                                      061967WN
00006C 07F1                        39          BCR   15,1                                           061967WN
00006E 0000
000070 00000020                    40 ADZZZ    DC    A(ZZZ)                                         061967WN
000074 00000000                    41 ADIBCOM  DC    A(IHCFCOME)                                    061967WN
000078 0000000000000000            42 SAVE     DC    18F'0'                                         061967WN
0000C0 00000000                    43 ADFIOCS  DC    A(IHCFIOSH)                                    061967WN
0000C4 00000013                    44 I19      DC    F'19'
0000C8                             45 SV1      DS    F                                              061967WN
0000CC                             46 SV2      DS    F                                              061967WN
0000D0 00F0                        47 HOOFO    DC    X'00F0'                                        061967WN
                                   48          END                                                  061967WN
```

10/29/67

POS.ID	REL.ID	FLAGS	ADDRESS
01	01	0C	000070
01	02	0C	000074
01	03	0C	0000C0

CROSS-REFERENCE PAGE 1

10/29/67

SYMBOL	LEN	VALUE	DEFN	REFERENCES
ADFIOCS	00004	0000C0	0043	0023 0029
ADIBCOM	00004	000074	0041	0009
ADZZZ	00004	000070	0040	0008
BYPASS	00004	00005C	0035	0027
CHK19	00006	00003C	0026	0022
HOOFO	00002	0000D0	0047	0021
IHCFCOME	00001	000000	0003	0041
IHCFIOSH	00001	000000	0004	0043
I19	00004	0000C4	0044	0026
LEAVE	00004	000064	0037	0034
REREAD	00002	000000	0005	0002 0006
SAVE	00004	000078	0042	0016 0024 0038
SV1	00004	0000C8	0045	0007 0013 0032 0035
SV2	00004	0000CC	0046	0033 0036
ZZZ	00004	000020	0016	0019 0040

NO STATEMENTS FLAGGED IN THIS ASSEMBLY
82 PRINTED LINES

```
IEF285I   SYS1.MACLIB                                KEPT
IEF285I   VOL SER NOS= 000017.
IEF285I   AAAAAAAA.AAAAAAAA.AAAAAAAA.AAAAAAAA.00000042 PASSED
IEF285I   VOL SER NOS= 000017.
IEF285I   AAAAAAAA.AAAAAAAA.AAAAAAAA.AAAAAAAA.00000043 DELETED
IEF285I   VOL SER NOS= 000017.
IEF285I   AAAAAAAA.AAAAAAAA.AAAAAAAA.AAAAAAAA.00000044 DELETED
IEF285I   VOL SER NOS= 000011.
IEF285I   SYSOUT                                     SYSOUT
IEF285I   VOL SER NOS=
IEF285I   LOADSET.REREAD                             PASSED
IEF285I   VOL SER NOS= 000011.
//LKED EXEC PGM=IEWL,PARM=(MAP,LET,NCAL,DC),COND=(5,LT,COMP)          00000090
//SYSLIN   DD  DSNAME=&LOADSET,DISP=(OLD,DELETE),UNIT=DISK            00000100
//          DD  DDNAME=SYSIN                                          00000110
//SYSUT1 DD DSNAME=*.COMP.SYSUT1,DISP=(OLD,DELETE)                    00000120
//LKED.SYSLMOD DD DSNAME=SYS1.FORTLIB(REREAD),DISP=OLD
//SYSLMOD DD DSNAME=TEST.LOAD,DISP=OLD                              00000130
//SYSPRINT DD  SYSOUT=A                                             00000140
//
IEF236I ALLOC. FOR REREAD   LKED     STEP1
IEF237I SYSLIN   ON 191
IEF237I SYSUT1   ON 190
IEF237I SYSLMOD  ON 192
```

E-LEVEL LINKAGE EDITOR OPTIONS SPECIFIED MAP,LET,NCAL,DC
IEW0461 IHCFCOME
IEW0461 IHCFIOSH
****REREAD DOES NOT EXIST BUT HAS BEEN ADDED TO DATA SET

DIAGNOSTIC MESSAGE DIRECTORY

IEW0461 WARNING - SYMBOL PRINTED IS AN UNRESOLVED EXTERNAL REFERENCE, NCAL WAS SPECIFIED.

MODULE MAP

CONTROL SECTION			ENTRY							
NAME	ORIGIN	LENGTH	NAME	LOCATION	NAME	LOCATION	NAME	LOCATION	NAME	LOCATION
$PRIVATE	00	D2								
			REREAD	00						

ENTRY ADDRESS 00
TOTAL LENGTH D2

```
IEF285I   LOADSET.REREAD                                  DELETED
IEF285I   VOL SER NOS= 000011.
IEF285I   AAAAAAAA.AAAAAAAA.AAAAAAAA.AAAAAAAA.00000042 DELETED
IEF285I   VOL SER NOS= 000017.
IEF285I   SYS1.FORTLIB                                    KEPT
IEF285I   VOL SER NOS= 000005.
IEF285I   SYSOUT                                          SYSOUT
IEF285I   VOL SER NOS=      .
```

```
//TREREAD JOB MSGLEVEL=1
//STEP1 EXEC FORTECLG
//COMP EXEC PGM=IEJFAAA0                                                    00000010
//SYSPRINT DD SYSOUT=A                                                      00000020
//SYSUT1 DD UNIT=DISK,SPACE=(1024,(200,10)),DISP=(NEW,PASS)                 00000030
//SYSUT2 DD UNIT=DISK,SPACE=(TRK,(30,10))                                   00000040
//SYSLIN DD DSNAME=&LOAD,DISP=(MOD,PASS),UNIT=DISK,SPACE=(TRK,(30,10))      00000050
//COMP.SYSIN DD DATA
IEF236I ALLOC. FOR TREREAD  COMP     STEP1
IEF237I SYSUT1    ON 190
IEF237I SYSUT2    ON 192
IEF237I SYSLIN    ON 191
IEF237I SYSIN     ON 00C


IEJ001I COMPILER OPTIONS IN EFFECT: SOURCE,MAP,LOAD,ADJUST,PRFRM,SIZE=45056,LINELNG=132
```

12

498

LEVEL: 1JUL66 IBM OS/360 BASIC FORTRAN IV (E) COMPILATION DATE: 67.302

```
            C 061967                                                    061967WN
S.0001          DIMENSION A(20)                                         061967WN
S.0002          CALL REREAD                                             061967WN
S.0003          READ (5,1) A                                            061967WN
S.0004        2 FORMAT (' ',20A4)                                       061967WN
S.0005        1 FORMAT (20A4)                                           061967WN
S.0006          DO 3 I=1,10                                             061967WN
S.0007        3 WRITE (6,2) A                                           061967WN
S.0008       10 FORMAT (3F6.2,I4)                                       061967WN
S.0009          READ (19,10) X,Y,Z,J
S.0010          DO 5 I=1,10                                             061967WN
S.0011        5 WRITE (6,7) X,Y,Z,J
S.0012        7 FORMAT (' ',3F12.2,I10)                                 061967WN
S.0013          STOP                                                    061967WN
S.0014          END                                                     061967WN
```

STORAGE MAP VARIABLES (TAGS: C=COMMON, E=EQUIVALENCE)

NAME	TAG	REL ADR	NAME	TAG	REL ADR	NAME	TAG	REL ADR	NAME	TAG	REL ADR
A		000154	I		0001A4	X		0001A8	Y		0001AC
Z		0001B0	J		0001B4						

EXTERNAL REFERENCES

NAME	REL ADR	NAME	REL ADR	NAME	REL ADR	NAME	REL ADR
REREAD	000188						

CONSTANTS

NAME	REL ADR	NAME	REL ADR	NAME	REL ADR	NAME	REL ADR

IMPLIED EXTERNAL REFERENCES

NAME	REL ADR	NAME	REL ADR	NAME	REL ADR	NAME	REL ADR
IBCOM#	0001FC						

STATEMENT NUMBER	REL ADR	STATEMENT NUMBER	REL ADR	STATEMENT NUMBER	REL ADR	STATEMENT NUMBER	REL ADR
00002	0001C8	00001	0001D4	00003	000248	00010	0001DC
00005	000288	00007	0001E8				

SIZE OF COMMON 000000 PROGRAM 000782

END OF COMPILATION MAIN

```
IEF285I   SYSOUT                                          SYSOUT
IEF285I   VOL SER NOS=
IEF285I   AAAAAAAA.AAAAAAAA.AAAAAAAA.AAAAAAAA.00000049 PASSED
IEF285I   VOL SER NOS= 000017.
IEF285I   AAAAAAAA.AAAAAAAA.AAAAAAAA.AAAAAAAA.00000050 DELETED
IEF285I   VOL SER NOS= 000005.
IEF285I   LOAD.TREREAD                                    PASSED
IEF285I   VOL SER NOS= 000011.
//LKED  EXEC PGM=IEWL,PARM=(MAP,LET,LIST,DC),COND=(5,LT,COMP)          00000060
//SYSLIB DD DSNAME=SYS1.FORTLIB,DISP=OLD                               00000070
//SYSUT1 DD DSNAME=*.COMP.SYSUT1,DISP=(OLD,DELETE)                     00000080
//SYSPRINT DD SYSOUT=A                                                 00000090
//SYSLIN DD DSNAME=&LOAD,DISP=(OLD,DELETE)                             00000100
//       DD DDNAME=SYSIN                                               00000110
//SYSLMOD DD DSNAME=&GO(MAIN),DISP=(NEW,PASS),UNIT=DISK,              X00000120
//             SPACE=(3072,(20,10,1))                                  00000130
IEF236I ALLOC. FOR TREREAD  LKED     STEP1
IEF237I SYSLIB    ON 192
IEF237I SYSUT1    ON 190
IEF237I SYSLIN    ON 191
IEF237I SYSLMOD   ON 190
```

E-LEVEL LINKAGE EDITOR OPTIONS SPECIFIED MAP,LET,LIST,DC
****MAIN DOES NOT EXIST BUT HAS BEEN ADDED TO DATA SET

MODULE MAP

CONTROL SECTION			ENTRY							
NAME	ORIGIN	LENGTH	NAME	LOCATION	NAME	LOCATION	NAME	LOCATION	NAME	LOCATION
MAIN	00	30E								
$PRIVATE *	310	02								
			REREAD	310						
IHCFCOME*	3E8	1464								
			IBCOM#	3E8	FDIOCS#	9F4				
IHCFIOSH*	1850	CF2								
			FIOCS#	1850						
IHCUATBL*	2548	138								

ENTRY ADDRESS 00
TOTAL LENGTH 2680

```
IEF285I   SYS1.FORTLIB                                    KEPT
IEF285I   VOL SER NOS= 000005.
IEF285I   AAAAAAAA.AAAAAAAA.AAAAAAAA.AAAAAAAA.00000049 DELETED
IEF285I   VOL SER NOS= 000017.
IEF285I   SYSOUT                                          SYSOUT
IEF285I   VOL SER NOS=
IEF285I   LOAD.TREREAD                                    DELETED
IEF285I   VOL SER NOS= 000011.
IEF285I   GO.TREREAD                                      PASSED
IEF285I   VOL SER NOS= 000017.
//GO EXEC PGM=*.LKED.SYSLMOD,COND=((5,LT,COMP),(5,LT,LKED))          00000140
//FT01F001 DD DDNAME=SYSIN                                          00000150
//FT02F001 DD UNIT=PUNCH                                            00000160
//FT03F001 DD SYSOUT=A                                              00000170
//GO.FT06F001 DD SYSOUT=A
//GO.FT19F001 DD DUMMY
//GO.FT05F001 DD DATA
IEF236I ALLOC. FOR TREREAD  GO       STEP1
IEF237I PGM=*.DD ON 190
IEF237I FT02F001 ON 00A
IEF237I FT05F001 ON 00C

12345678912345678912345678912345678912345678912345678912345678912345678912345678
12345678912345678912345678912345678912345678912345678912345678912345678912345678
12345678912345678912345678912345678912345678912345678912345678912345678912345678
12345678912345678912345678912345678912345678912345678912345678912345678912345678
12345678912345678912345678912345678912345678912345678912345678912345678912345678
12345678912345678912345678912345678912345678912345678912345678912345678912345678
12345678912345678912345678912345678912345678912345678912345678912345678912345678
12345678912345678912345678912345678912345678912345678912345678912345678912345678
12345678912345678912345678912345678912345678912345678912345678912345678912345678
12345678912345678912345678912345678912345678912345678912345678912345678912345678
     1234.56     7891.23     4567.89     1234
     1234.56     7891.23     4567.89     1234
     1234.56     7891.23     4567.89     1234
     1234.56     7891.23     4567.89     1234
     1234.56     7891.23     4567.89     1234
     1234.56     7891.23     4567.89     1234
     1234.56     7891.23     4567.89     1234
     1234.56     7891.23     4567.89     1234
     1234.56     7891.23     4567.89     1234
     1234.56     7891.23     4567.89     1234
```

Jim Miller's Adaption of REREAD to DOS supplied 5 Sep 67 DOS GLZ-3 08/24/6

Changes from O.S. version noted in red (underlined).

```
LOC  OBJECT CODE     ADDR1 ADDR2  STMT  SOURCE STATEMENT
000000                              1  REREAD   START 0
                                    2           ENTRY REREAD
                                    3           EXTRN IJTACOM
                                    4           EXTRN IJTFIOS
000000                              5           USING *,15
000000 5800 F060            00060   6           L     0,ADZZZ       LOAD 0 WITH ADDRESS OF ZZZ
000004 5810 F064            00064   7           L     1,ADIBCOM     LOAD 1 WITH ADDRESS OF IJTACOM
000008 5000 1818            00818   8           ST    0,2072%0,1¤
00000C 07FE                         9           BCR   15,14
00000E                             10           USING *,1
00000E 904F 105A            00068  11  ZZZ      STM   4,15,SAVE
                                   12           DROP  1
000012 18B1                        13           LR    11,1
00000E                             14           USING ZZZ,11
000014 1840                        15           LR    4,0
000016 D503 4000 B0B2 00000 000C0  16           CLC   0%4,4¤,H00F0
00001C 4780 B01C            0002A  17           BC    8,CHK99
000020 5810 B0A2            000B0  18           L     1,ADFIOCS
000024 984F B05A            00068  19           LM    4,15,SAVE
000028 07F1                        20           BCR   15,1
00002A D503 E000 B0A6 00000 000B4  21  CHK99    CLC   0%4,14¤,I99
000030 4780 B03E            0004C  22           BC    8,BYPASS
000034 182E                        23           LR    2,14
000036 5810 B0A2            000B0  24           L     1,ADFIOCS
00003A 0501                        25           BALR  0,1
00003C 00F0                        26           DC    X@00F0@
00003E 0000                        27           DC    X@0000@
000040 5020 B0AA            000B8  28           ST    2,SV1
000044 5030 B0AE            000BC  29           ST    3,SV2
000048 47F0 B046            00054  30           BC    15,LEAVE
00004C 5820 B0AA            000B8  31  BYPASS   L     2,SV1
000050 5830 B0AE            000BC  32           L     3,SV2
000054 4114 0004            00004  33  LEAVE    LA    1,4%4¤
000058 984F B05A            00068  34           LM    4,15,SAVE
00005C 07F1                        35           BCR   15,1
00005E 0000
000060 0000000E                    36  ADZZZ    DC    A%ZZZ¤
000064 00000000                    37  ADIBCOM  DC    A%IJTACOM¤
000068 0000000000000000            38  SAVE     DC    18F@0@
0000B0 00000000                    39  ADFIOCS  DC    A%IJTFIOS¤
0000B4 0000000A                    40  I99      DC    F@10@
0000B8                             41  SV1      DS    F
0000BC                             42  SV2      DS    F
0000C0 00F00000                    43  H00F0    DC    X@00F00000@
                                   44           END
```

SESSION REPORT

Session No. W. 15

Session Title: 1130 PROJECT

Chairman: DAVID A. DUNSMORE

Speaker: BRIAN SWAIN

Topic:: 1130 PRINTER OVERLAP "WRITE" W/CSP

Summary: Rename SF10 from monitor/replace with another w/PRINT1
Delete PRINT2. Add new SF10 and PRNTX
DIMENSION IA(18)
READ(99,101) IA (STORES 2/WORD FROM 101 FORMAT TO IA)
101 FORMAT(1-)

Speaker: Bob Besinaught (2)

Topic: 1130 Commercial Subroutines

Summary; General History of Development of Comercial
Subroutines and some of the functions provided
to the user.

Speaker:

Topic:

Summary:

NO. OF ATTENDEES: approx 60

W. 1. 5

OVERLAPPED PRINTING FOR IBM 1130 COMMERCIAL APPLICATIONS USING FORTRAN WRITE STATEMENT

by

B. J. SWAIN

THE SHAWINIGAN ENGINEERING COMPANY LIMITED

Box 3010 Station B, Montreal, Canada.

DECEMBER 1967

506

TABLE OF CONTENTS

		Page
ABSTRACT		i
DISCLAIMER		ii
TEXT		
I.	Use of Fortran for Commercial Applications on the IBM 1130	1
II.	Programming Philosophy Using Commercial Subroutine Packages	2
III.	Difficulties Presented by the Use of Commercial Subroutine Packages	3
IV.	Use of Fortran WRITE Statement for Output in Commercial Reports	5
	Definition of Alphabetic Literals Using Fortran READ Statement	8
V.	Performance of the Subroutine	9
APPENDIX I		
	Fortran Output Routine, with Overlap Users Guide	12
	WRITE	12
	READ	14

TABLE OF CONTENTS cont'd

Page

APPENDIX II

Loading the Overlapped Fortran Output Routine 16

APPENDIX III

Source Listings 17

ABSTRACT

A method is described for incorporating overlapped printing into IBM 1130 Commercial FORTRAN programs. Communication to the subprogram which performs a printing operation is achieved through the FORTRAN WRITE statement rather than through the CALL statement. The advantage of this method is that limited use can be made of the formatting ability of the FORTRAN language. Headings can readily be incorporated, and the layout of the printed page specifically by use of FORMAT statements.

DISCLAIMER

Although each program has been tested by its contributor, no warranty, express or implied, is made by the contributor or COMMON USERS Group, as to the accuracy and functioning of the program and related program material, nor shall the fact of distribution constitute any such warranty, and no responsibility is assumed by the contributor or COMMON USERS Group, in connection therewith.

OVERLAPPED PRINTING FOR IBM 1130 COMMERCIAL APPLICATIONS USING FORTRAN WRITE STATEMENT

I. Use of Fortran for Commercial Applications on the IBM 1130.

Since the IBM 1130 was conceived as a computer to be used for the solution of moderate sized engineering and scientific problems, Fortran is the only compiler supported on the system. At many installations, it is necessary to use the computer, not only in this prime role, but also in the preparation of commercial reports, such as payrolls, labor distribution, and time control. In order to permit the use of Fortran for these applications, a number of subroutine packages have been developed. IBM's Commercial Subroutine Package, Version II, is an example. Others are the contributed program packages COMET and IDEAL. The purpose of these subroutine packages is to overcome the problems which would otherwise be presented in handling commercial applications in Fortran. Principally these are the following:

1) Moving character strings.
2) Use of floating dollar signs, check protection, etc.
3) Elimination of round-off in crossfooting totals
4) Zone recognition
5) Stacker Selection
6) Overlapped input/output for increasing processing speed.

The first five items in the list above exist on account of the fact that Fortran is not intended as a commercial data processing language. The sixth item occurs on account of the fact that the Fortran compiler for the IBM 1130 has been designed to conserve core space as much as possible. The normal input/output subroutines for use with Fortran use the same area in core as a buffer area to contain the image of a card being read or a line being printed. It is therefore not possible to overlap these operations, although the interrupt hardware provided on the system would otherwise permit this to be done. In the design of the IBM 1130 Commercial Subroutine Package, it has been recognized that for commercial applications, processing speed assumes a greater importance. Overlap of input/output operations has therefore been achieved at the expense of the additional core required for multiple buffers, and for the more complicated subroutines required to service the interrupts.

II. Programming Philosophy Using Commercial Subroutine Packages

The programming philosophy usually employed when commercial subroutine packages are used is as follows:

1) Input data on cards is read and stored as a card image, in EBCDIC form. A SUBROUTINE subprogram is used for this purpose.
2) Tests are made to determine the type of card which has been read.

Programming Philosophy Using Commercial Subroutine Packages cont'd

3) Data is extracted from the card image, and stored for subsequent calculations. If it is to be used in arithmetic operations, numerical data is converted to a code which permits these operations to be performed.

4) At the conclusion of the necessary calculations, an image is built up of the output line to be printed in the report. Numeric information is reconverted to EBCDIC code for inclusion in this output line image.

5) The output line is printed, using a SUBROUTINE subprogram.

The use of this philosophy imposes virtually no restriction on the manner of presentation of data in the input file, or on the appearance of the final report. The subprograms for card reading and printing permit overlap of these operations with each other. Conversion from card code to EBCDIC is also overlapped with card reading.

III. Difficulties Presented by the Use of Commercial Subroutine Packages.

Although the use of commercial subroutine packages in the manner described above overcomes the shortcomings of Fortran for commercial

Difficulties Presented by the Use of Commercial Subroutine Packages cont'd

applications, a number of additional difficulties are introduced. Principally these are the following:

1) The introduction of headings into the output report.

2) Loss of tracing capability. IBM 1130 Fortran has been provided with a very useful trace feature. This cannot be used when the overlapped input/output subroutines of the Commercial Subroutine Package are loaded, as it is not possible to include the Fortran input/output subroutines in the same core load. The Fortran input/output subroutines are required for the use of the trace feature.

3) A considerable amount of manipulation is often required to build an image of a line of the output report. This tends to make the coding of programs longer and more prone to errors.

4) The introduction of masks for editing. The use of Commercial Subroutine Package EDIT subroutine requires that non-numeric information to be mixed into a numeric field (such as $. etc) be introduced in the form of a mask, which is subsequently

Difficulties Presented by the Use of Commercial Subroutine Packages cont'd

applications, a number of additional difficulties are introduced. Principally these are the following:

1) The introduction of headings into the output report.

2) Loss of tracing capability. IBM 1130 Fortran has been provided with a very useful trace feature. This cannot be used when the overlapped input/output subroutines of the Commercial Subroutine Package are loaded, as it is not possible to include the Fortran input/output subroutines in the same core load. The Fortran input/output subroutines are required for the use of the trace feature.

3) A considerable amount of manipulation is often required to build an image of a line of the output report. This tends to make the coding of programs longer and more prone to errors.

4) The introduction of masks for editing. The use of Commercial Subroutine Package EDIT subroutine requires that non-numeric information to be mixed into a numeric field (such as $. etc) be introduced in the form of a mask, which is subsequently

Difficulties Presented by the Use of Commercial Subroutine Packages cont'd

destroyed by the inclusion of the numeric information. In view of the fact that alphabetic literals cannot be directly defined in the IBM 1130 Fortran compiler, this operation is clumsy (Note - Version II of the IBM 1130 Fortran compiler includes the DATA statement which overcomes this difficulty).

IV Use of Fortran WRITE Statement for Output in Commercial Reports.

At The Shawinigan Engineering Company Limited offices at Montreal, Canada, an IBM 1130 computer was installed in January 1967, primarily to perform engineering calculations. However, it was desired to use the machine for processing the commercial reports which would be required for the administration of the Company. A study was made of the available software for commercial processing, and as a result a set of subroutines written, which combined the best features of the IBM 1130 Commercial Subroutine Package, COMET, and IDEAL.

In recognition of the difficulties which would be presented by the use of SUBROUTINE subprograms for printing, after the manner of the IBM Commercial Subroutine Package, it was decided to adopt a different approach. A new subroutine SFIØ (the subroutine which handles in Fortran the input/output functions) was written in

Use of Fortran WRITE statement for Output in Commercial Reports cont'd

1130 assembler language, having more limited capabilities than the version supplied as part of the system, but taking advantages of the overlap feature of the hardware. The advantages of this procedure over the use of the Commercial Subroutine Package subprograms are the following:

1) Headings may be introduced by the use of a FORMAT statement, as is done in normal processing using Fortran.

2) Tracing can be employed.

3) The amount of manipulation required to format an output line is reduced, as the FORMAT statement may be used for this purpose.

It was not considered necessary to incorporate all the features which are supported by the normal SFIØ subroutine as supplied in the system. Indeed to have done so would have resulted in a subroutine occupying a large core space, a considerable portion of which would have been wasted on features which would be rarely used in the commercial area. The following restrictions were therefore introduced:

1) Only integer variables and arrays may be included in the output list.

2) *ONE WORD INTEGERS option must be used.

Use of Fortran WRITE Statement for Output in Commercial Reports cont'd

3) Card reading is not supported, and all writing is done on the 1132 printer, regardless of the device code appearing in the WRITE statement. Card, typewriter and console input/output are handled by SUBROUTINE subprograms as in the IBM Commercial Subroutine Package.

4) Only the following format field specifications are recognized:

A	for alphameric information. The only field widths recognized are A1 and A2 Anything else is interpreted as A1.
I	for integer variables
X	for spaces
H	for printing alphabetic information. The form using quotes may also be used.

5) The group repeat, e.g. 2(1H1,12A2), is not recognized.

6) Carriage control characters are restricted to the following:

Blank	Print on next line
Zero	Skip one line before printing
1- 6	Skip to channels 1 to 6

Use of Fortran WRITE Statement for Output in Commercial Reports cont'd

None of these restrictions prevent the use of the system in association with Commercial Subroutine Package, COMET or IDEAL. An attempt to print a real variable will result in the output field being filled with asterisks. This restricts the use of tracing to integer variables only, real variables appearing as a line of asterisks. It is our experience that this is not a serious restriction, as it is the values of integer variables which determine the logic flow of a program. If it is necessary to determine the value of a real variable for tracing purposes, then this can be done by converting it to EBCDIC, for display using a WRITE statement temporarily inserted in the program.

Definition of Alphabetic Literals Using Fortran READ Statement

In order to permit the definition of alphabetic literal character strings, a feature was introduced into the re-written SFIØ to permit this to be done using the Fortran READ statement. The manner of achieving this is shown by the following example:

```
      DIMENSION IA(13)
      READ(99,101)IA
  101 FORMAT('THIS MESSAGE TO BE STORED')
```

These statements cause the EBCDIC equivalent of the characters contained in the FORMAT statement to be stored in IA. No modification to the compiler is required. Characters are stored

Definition of Alphabetic Literals Using Fortran READ Statement cont'd

two per word. If it is desired to store characters one per word, then appropriate spaces should be introduced in the field. This feature is particularly useful for the definition of masks to be used for editing operations.

V. Performance of the Subroutine

Using the Disk Monitor System, the subroutine is implemented by deleting from the disk the version of SFIØ as supplied with the system. The subroutine PRNTZ must also be deleted. The subroutines SFIØ and PRNXX, which form the overlapped output package are then substituted. Note that SFIØ can be stored as a subtype 3 program, which permits it to be included in SOCAL overlays. The subroutine PRNT1 supplied by IBM is also required as it is used to drive the 1132 printer. Fortran programs should contain the record

```
*IOCS(1132 PRINTER)
```

or

```
*IOCS(1132 PRINTER,DISK)
```

depending upon whether disk input/output is required. The principal benefit to be derived from using the method is the simplicity of coding. Also the processing speed which can be achieved using this subroutine is apparently identical to that achieved using the overlapped output subroutines supplied with

Performance of the Subroutine cont'd

the Commercial Subroutine Package. The core requirement for SFIØ and PRNXX is words 470, which is somewhat in excess of the equivalent subprograms supplied with the Commercial Subroutine Package. However, this is offset by the smaller core requirements of programs which use this method. As an example, one of the sample programs provided with the Commercial Subroutine Package was modified to suit the overlapped input/output subroutines. The processing speed was the same in both instances. Despite the smaller and simpler main program when using the overlapped Fortran I/O, the total core requirement is somewhat larger, as is shown in the table below. For comparison, the core requirements and processing speed using the normal Fortran input/output subroutines as supplied with the disk monitor system are also shown.

Comparison of Typical Small Commercial Application

	CSP I/O	Overlapped Fortran I/O	Standard Fortran I/O.
Main Program Statements	102	94	83
Main Program Core Requirement	978	850	800
Total size of Core Load	5006	5298	4858
Execution Time	1 32 min sec	1 32 min sec.	2 45 min sec.

Performance of the Subroutine cont'd

The problem selected was the invoicing problem which is included as Sample Problem 2 in the IBM Commercial Subroutine Package, Version II. It is considered that for programs to produce large and complicated commercial reports, where the size of a core load becomes important, the total core load requirement of programs using the output method described here would be smaller than would be required for the same programs using the Commercial Subroutine Package.

APPENDIX I

Fortran Output Routine, with Overlap

Users Guide.

This is a program to replace the IBM library program SFIØ, which is called for Fortran input/output operations. Its purpose is to permit overlap of printing, card reading and processing, for commercial applications. For this reason, permissible devices, allowable variable types and format field specifications have been restricted. Only integer variables and arrays are recognized *ONE WORD INTEGERS option must be used.

WRITE

All writing is done on the 1132 printer, regardless of the device code appearing in the WRITE statement. If it is desired to write on the Console Printer or punch cards then an appropriate SUBROUTINE subprograms must be used. Data to be printed must be stored in EBCDIC or integer form as an integer variable or array.

The following FORMAT field specifications are recognized:

A	for alphameric information. The only field widths recognized are A1 and A2. Anything else is interpreted as A1, with no error indication.
I	for integer variables

WRITE cont'd

X for spaces

H for printing alphabetic information.

The form using quotes e.g.

' THIS MESSAGE' may also be used.

Field repeats e.g. 12A2

The group repeat, e.g. 2(1H1, 12A2), will not be recognized.

The maximum number of characters per line is 120, excluding the carriage control character. The use of E or F field specifications will cause the field to be filled with asterisks without further indication of error.

The first character of a line will be treated as a carriage control character. The following characters are recognized:

blank print on next line

0 skip one line before printing

1 - 6 skip to channels 1 to 6

All other carriage control characters are treated as blank.

The following errors will cause a program stop with the error code displayed in the accumulator.

EE01 line exceeds 120 characters

EE02 invalid format type

READ

No read statement, accepting information from an external medium is included. Appropriate SUBROUTINE subprograms must be used for input from the card reader or console keyboard. The following statement is provided in order to generate alphabetic literal arrays.

```
READ(99,n) IA
     or
READ(99,n) IA(I)
READ(99,n) (IA(i),I=K,L)
READ(99,n)((IA(I,J),I=K,L),J=M,N)
```

which transfers H-type fields from format statement 'n' into variable or array IA. The characters will be stored in EBCDIC form, 2 characters per word. Only H-type fields are permitted in the FORMAT statement. If the number of characters in a field is odd, the rightmost character of the last word is filled with a blank. Characters will be moved until the array is filled. If the FORMAT statement is exhausted, control will return to the last open bracket. The following errors will cuase a program stop with the error code displayed in the accumulator.

EE02	format not H-type
EE03	device code not 99

This routine uses the IBM library subroutine PRNT1. Since printing occurs in overlapped mode, a call to IOND must be made before a PAUSE or STOP statement.

READ cont'd

The input/output routine requires the use of *IOCS(1132 PRINTER). Since communication to the card-reader, or typewriter is through CALL statements to subprograms which load the appropriate card or typewriter subroutines, then CARD, TYPEWRITER or KEYBOARD must not appear in the *IOCS record.

Tracing may be used in conjunction with this overlapped I/O routine. Since real variables are not converted by this routine, they will appear as asterisks in the tracing output.

APPENDIX II

Loading the Overlapped Fortran Output Routine

Using the disk monitor system, the following procedure is required.

- Delete SFIØ and PRNTZ

  Load the overlapped version of SFIØ, and PRNXX.

Note that SFIØ can be designated subtype 3 (punch 3 in column 11 of *STORE card), and hence can be included in SOCAL overlays. PRNXX cannot be included in overlays.

Core Requirements:

SFIØ	456	words
PRNXX	14	"
	470	"

The IBM supplied subroutine PRNT1 and IØND from the Commercial Subroutine Package are also required.

```
// ASM
*LIST SOURCE PROGRAM
                        *       LIMITED CAPABILITY I/O SUBROUTINE
                        *       WITH OVERLAP
                                LIBR
0000    22189580                ENT     SFIO
0076    22645100                ENT     SRED
0095    229998C0                ENT     SWRT
0013    22256240                ENT     SIOI
001F    22256267                ENT     SIOIX
0026    22256049                ENT     SIOAI
002F    22256180                ENT     SIOF
0033    220D6517                ENT     SCOMP
01C4    176558E9                ENT     PRNTZ
                        *       MAINLINE PROGRAM INITIALISATION
0000 0  1000            SFIO    NOP
0001 01 66800003                LDX   I2 *              SET XR2 TO LIBF&1
0003 0  7212                    MDX    2 18
0004 0  6A01                    STX    2 *&1
0005 01 4C000007                BSC   L *               RETURN TO START OF PROGRAM
                        *       END OF MAINLINE INITIALISATION
                        *       CALL ROUTINE
                        *       RETURNS TO MAIN PROGRAM TO GET
                        *       NEXT VARIABLE DESCRIPTION
0007 0  C030            CALL    LD      LOOPS           TEST FOR I/O OF COMPLETE
0008 0  9069                    S       ONE1            ARRAY
0009 01 4C080011                BSC   L RETRN,&
000B 01 74FF0111                MDX   L VARAD,-1        INCREMENT TO NEXT WORD IN
000D 01 74FF0038                MDX   L LOOPS,-1        ARRAY
000F 01 4C00008D                BSC   L DECOD           GO TO DECODE ROUTINE
0011 01 4C000013        RETRN   BSC   L *               RETURN TO MAINLINE PROGRAM
                        *   MAINLINE PROGRAM CALLS ONE OF FOLLOWING ENTRIES
0013 0  1000            SIOI    NOP                     ENTRY FOR SINGLE INTEGER
0014 01 66800016                LDX   I2 *              SET XR2 TO LIBF&1
0016 0  C200                    LD     2 0              GET ADDRESS OF VARIABLE
0017 0  7201            UP1     MDX    2 1              COMPUTE RETURN ADDRESS
0018 01 D4000111        SAVAD   STO   L VARAD
001A 0  C057                    LD      ONE1            SET LOOPS TO 1
001B 0  D01C            LPSET   STO     LOOPS
001C 0  6AF5                    STX    2 RETRN&1        SAVE RETURN ADDRESS
001D 01 4C0000BD                BSC   L DECOD           GO TO DECODE ROUTINE
001F 0  1000            SIOIX   NOP                     ENTRY FOR SUBSCRIPTED INT.
0020 01 66800022                LDX   I2 *              SET XR2 TO LIBF&1
0022 0  C200                    LD     2 0              GET ADDRESS OF VARIABLE
0023 00 84000001                A     L 1               MODIFY BY XR1
0025 0  70F1                    MDX     UP1             JOIN SIOI ENTRY
0026 0  1000            SIOAI   NOP                     ENTRY FOR ARRAY
0027 01 66800029                LDX   I2 *              SET XR2 TO LIBF&1
0029 0  C200                    LD     2 0              GET ADDRESS OF VARIABLE
002A 01 D4000111                STO   L VARAD
002C 0  C201                    LD     2 1              GET NO. OF ELEMENTS
002D 0  7202                    MDX    2 2              COMPUTE RETURN ADDRESS
002E 0  70EC                    MDX     LPSET           JOIN SIOI ENTRY
                        *       THIS ENTRY TO SATISFY LIBF SIOF IN
                        *       TRACE ROUTINES. JOINS SIOI
002F 0  1000            SIOF    NOP
0030 01 66800032                LDX   I2 *              SET XR2 TO LIBF&1
0032 0  70E3                    MDX     SIOI&3          JOIN SIOI
0033 0  1000            SCOMP   NOP                     ENTRY FOR END OF OUTPUT
0034 01 66800036                LDX   I2 *              SET XR2 TO LIBF&1
0036 0  1810                    SRA     16              SET LOOPS TO 0
```

```
0037 0  70E3              MDX     LPSET       JOIN SIOI ENTRY
0038    0001      LOOPS   BSS     1           ARRAY COUNTER
                  *       END OF CALL ROUTINE
                  *       HTYPE ROUTINE
                  *       TRANSFERS H-TYPE CHARACTERS TO I/O BUFFER
0039 0  6200      HTYPE   LDX   2 0           SET XR2 AS CHAR-COUNTER
003A 0  7201      HLOOP   MDX   2 1           INCR. COUNT
003B 01 44000114          BSI   L LINE
003D 00 C4000002          LD    L 2           BIT 15 OF XR2 TO CARRY
003F 0  1010              SLA     16
0040 01 C4800112          LD    I IFMT        GET CHAR FROM FORMAT
0042 0  4802              BSC     C           SKIP IF EVEN CHAR
0043 0  1808              SRA     8           IF ODD, SHIFT RIGHT
0044 0  E030              AND     MASKR       CLEAR GARBAGE
0045 01 44000134          BSI   L IOFIL       MOVE TO IOBUF
0047 00 C4000002          LD    L 2           MOVE FORMAT POINTER IF
0049 01 4C04004D          BSC   L HODD,E      SECOND CHAR IN WORD IS
004B 01 74010112          MDX   L IFMT,1      MOVED
004D 0  9026      HODD    S       FWIDE       TEST FOR LAST CHAR.
004E 01 4C28003A          BSC   L HLOOP,&Z    LEAVE WHEN DIFF  0
0050 0  C023              LD      FWIDE       END OF TRANSFER. IF ODD NO
0051 01 4C040054          BSC   L HEND,E      OF CHARACTERS, MOVE FORMAT
0053 0  7069              MDX     DECOD       POINTER
0054 01 74010112  HEND    MDX   L IFMT,1
0056 0  7066              MDX     DECOD
                  *       END OF HTYPE ROUTINE
                  *       XTYPE ROUTINE
                  *       SKIPS OVER CHARACTERS IN I/O BUFFER
0057 0  C01C      XTYPE   LD      FWIDE       SET XR2 TO FIELD WIDTH
0058 00 D4000002          STO   L 2
005A 01 44000114  XLOOP   BSI   L LINE
005C 0  72FF              MDX   2 -1          TEST FOR LAST CHARACTER
005D 0  70FC              MDX     XLOOP
005E 0  705E              MDX     DECOD
                  *       END OF XTYPE ROUTINE
                  *       ATYPE ROUTINE
                  *       TRANSFERS ONE OR TWO CHARACTERS FROM VARIABLE
                  *       TO I/O BUFFER
005F 01 44000114  ATYPE   BSI   L LINE
0061 01 C4800111          LD    I VARAD       TRANSFER LEFT CHARACTER
0063 0  1808              SRA     8
0064 01 44000134          BSI   L IOFIL       STORE
0066 0  C00D              LD      FWIDE
0067 0  900A              S       ONE1
0068 0  4808              BSC     &
0069 0  709D              MDX     CALL
006A 01 44000114          BSI   L LINE        TRANSFER RIGHT CHARACTER
006C 01 C4800111          LD    I VARAD
006E 0  E006              AND     MASKR
006F 01 44000134          BSI   L IOFIL
0071 0  7095              MDX     CALL
0072 0  0001      ONE1    DC      1
0073    0001      FTYPE   BSS     1           STORES FORMAT TYPE
0074    0001      FWIDE   BSS     1           STORES FIELD WIDTH
0075 0  00FF      MASKR   DC      /00FF
                  *       END OF ATYPE ROUTINE
                  *       INITIALISATION ENTRIES
                  *       SRED CALLED BY READ STATEMENT
0076 0  1000      SRED    NOP
```

```
0077 01 66800079          LDX  I2 *          SET XR2 TO LIBF&1
0079 00 C6800000          LD   I2 0          TEST DEVICE NO.
007B 0  901E              S       NINTN
007C 01 4C180084          BSC  L  ROK,&-
007E 30 09595100    DERR  CALL    IOND       ERROR. DISPLAY /EE03
0080 0  C01B              LD      EE3
0081 0  3000              WAIT
0082 00 4C000038          BSC  L  /38        EXIT
0084 0  1810        ROK   SRA     16         SET READ SWITCH
0085 01 D400010A    SAVSW STO  L  READS
0087 0  C201              LD   2  1          GET FORMAT ADDRESS
0088 01 D4000112          STO  L  IFMT
008A 0  7202              MDX  2  2          COMPUTE RETURN ADDRESS
008B 0  6A86              STX  2  RETRN&1
008C 0  1810              SRA     16         SET INITIAL CHARACTER
008D 01 D400014F          STO  L  IOCHR      COUNT
008F 01 C4000113          LD   L  ONE
0091 0  D077              STO     MULT       SET MULTIPLE FIELD COUNT
0092 0  D0A5              STO     LOOPS      SET ARRAY COUNTER
0093 01 4C000007          BSC  L  CALL
0095 0  1000        SWRT  NOP
0096 01 66800098          LDX  I2 *          SET XR2 TO LIBF&1
0098 0  C07A              LD      ONE        SET WRITE SWITCH
0099 0  70EB              MDX     SAVSW
009A 0  0063        NINTN DC      99
009B 0  0003        THREE DC      3
009C 0  EE03        EE3   DC      /EE03
                    *     END OF INITIALISATION
                    *     FIELD ROUTINE
                    *     TESTS FIELD TYPE
                    *     TRANSFERS TO APPROPRIATE ROUTINE
                    *     TO MOVE TO I/O BUFFER
009D 0  C0D5        FIELD LD      FTYPE
009E 0  906D              S       HCODE      TEST FOR HTYPE
009F 01 4C180039          BSC  L  HTYPE,&-
00A1 0  8018              A       HXCOD      TEST FOR XTYPE
00A2 01 4C180057          BSC  L  XTYPE,&-
00A4 0  9016              S       SXCOD      TEST FOR SLASH
00A5 01 4C2000AA          BSC  L  ATEST,Z
00A7 01 44000150          BSI  L  PRINT
00A9 0  7013              MDX     DECOD
00AA 01 74000038    ATEST MDX  L  LOOPS,0    TEST FOR SCOMP CALLED
00AC 0  7004              MDX     ATTST
00AD 01 44000150          BSI  L  PRINT
00AF 01 4C000011          BSC  L  RETRN
00B1 0  800A        ATTST A       SACOD      TEST FOR A TYPE
00B2 01 4C18005F          BSC  L  ATYPE,&-
00B4 0  8005              A       HXCOD      TEST FOR I TYPE
00B5 01 4C180179          BSC  L  ITYPE,&-
00B7 01 4C0801BF          BSC  L  FETYP,&
00B9 0  7034              MDX     INVAL      INVALID FIELD TYPE
00BA 0  0001        HXCOD DC      /0001      HCODE-XCODE
00BB 0  0003        SXCOD DC      /0003      SLASH-XCODE
00BC 0  0004        SACOD DC      /0004      SLASH-ACODE
003C                IOBUF EQU     /003C      FORTRAN I/O BUFFER
                    *     END OF FIELD ROUTINE
                    *     DECOD ROUTINE
                    *     EXTRACTS FIELD TYPE AND WIDTH FROM
                    *     FORMAT STATEMENT
```

530

```
                        *       IF MULTIPLE FIELD, SETS MULTIPLE
                        *       FIELD COUNTER
                        *       IF END OF FORMAT  STATEMENT, PRINTS
                        *       AND RESETS FORMAT SCAN POINTER
                        *       IF READ 99,XXX ,TRANSFERS CHARACTERS
00BD 0  C04B    DECOD  LD       MULT        TEST FOR MULTIPLE FIELD
00BE 0  9054           S        ONE         PREVIOUSLY FOUND
00BF 01 4C300101       BSC  L   MULTF,-Z
00C1 01 C4800112 NOMLT LD   I   IFMT        NOT MULTIPLE FIELD, GET
00C3 0  E04C           AND      MASK2       FORMAT TYPE AND DECODE
00C4 0  D04A           STO      WIDEC       STORES WIDTH OR REPEAT
00C5 01 C4800112       LD   I   IFMT
00C7 0  180C           SRA      12          MOVE TO RIGHT DIGIT
00C8 0  D045           STO      TYPEC       STORES TYPE CODE
00C9 01 74010112       MDX  L   IFMT,1      INCREMENT FORMAT POINTER
00CB 0  903F           S        FREPT       TEST FOR FIELD REPEAT
00CC 01 4C2000D1       BSC  L   REDTS,Z
00CE 0  C040           LD       WIDEC       FIELD REPEAT, STORE
00CF 0  D039           STO      MULT        NO. OF REPEATS
00D0 0  7030           MDX      MULTF
00D1 0  C03C    REDTS  LD       TYPEC       TEST FOR REDO CODE
00D2 0  9034           S        REDO
00D3 01 4C2000E3       BSC  L   SETF,Z
00D5 0  C03C           LD       IFMT        REDO CODE FOUND
00D6 0  903C           S        ONE         RESET FORMAT SCAN POINTER
00D7 0  9037           S        WIDEC
00D8 0  D039           STO      IFMT
00D9 0  C030           LD       READS       TEST FOR READ
00DA 01 4C0800C1       BSC  L   NOMLT,-
00DC 0  4073           BSI      PRINT       READ SWITCH NOT SET
00DD 01 C4000038       LD   L   LOOPS       PRINT, AND CHECK FOR
00DF 01 4C2000C1       BSC  L   NOMLT,Z     SCOMP CALLED
00E1 01 4C000011       BSC  L   RETRN
00E3 0  C02A    SETF   LD       TYPEC       SET FIELD WIDTH AND TYPE
00E4 0  D08E           STO      FTYPE
00E5 0  C029           LD       WIDEC
00E6 0  D08D           STO      FWIDE
00E7 0  C022    ENDDC  LD       READS       END OF DECODING FORMAT ST.
00E8 01 4C20009D       BSC  L   FIELD,Z
00EA 0  C088           LD       FTYPE       READ SWITCH SET, TEST TYPE
00EB 0  9020           S        HCODE
00EC 01 4C1800F4       BSC  L   REPT,&-
00EE 30 09595100 INVAL CALL     IOND        INVALID TYPE
00F0 0  C01C           LD       EE2
00F1 0  3000           WAIT
00F2 00 4C000038       BSC  L   /38
00F4 01 C4000074 REPT  LD   L   FWIDE       SET MULTIPLE FIELD COUNTER
00F6 0  801C           A        ONE         TO NO.OF WORDS IN H FIELD
00F7 0  1801           SRA      1
00F8 0  D010           STO      MULT
00F9 01 C4800112 TRANF LD   I   IFMT        TRANSFER CHARACTERS TO
00FB 01 D4800111       STO  I   VARAD       VARIABLE
00FD 01 74010112       MDX  L   IFMT,1
00FF 01 4C000007       BSC  L   CALL        TO CALL TO SEE IF DONE
0101 01 74FF0109 MULTF MDX  L   MULT,-1     REDUCE MULTIPLE FIELD COUNT
0103 0  C006           LD       READS
0104 01 4C20009D       BSC  L   FIELD,Z
0106 0  70F2           MDX      TRANF
0107 0  000B    REDO   DC       /000B       TEST FOR REDO INDICATOR
```

```
0108     0001        TEMP  BSS     1
0109     0001        MULT  BSS     1           MULTIPLE FIELD STORAGE
010A     0001        READS BSS     1           0 FOR READ 1 FOR PRINT
010B 0   0009        FREPT DC      /0009       TEST FOR FIELD REPEAT
010C 0   0005        HCODE DC      /0005       TEST FOR H TYPE FIELD
010D 0   EE02        EE2   DC      /EE02       ERROR DISPLAY-INVALID TYPE
010E     0001        TYPEC BSS     1           TEMPORARY STORAGE FOR TYPE
010F     0001        WIDEC BSS     1           TEMPORARY STORAGE FOR WIDTH
0110 0   0FFF        MASK2 DC      /0FFF
0111     0001        VARAD BSS     1           POINTS TO VARIABLE
0112     0001        IFMT  BSS     1           POINTS TO FORMAT STATEMENT
0113 0   0001        ONE   DC      1
                     *     LINE ROUTINE
                     *     CHECKS IF LINE IS FULL
                     *     DISPLAYS /EE01 IF FULL, OTHERWISE
                     *     INCREMENTS LINE COUNTER
                     *     IF START OF LINE, CHECKS PRINTER
                     *     READY, AND CLEARS I/O AREA
0114     0001        LINE  BSS     1
0115 0   C039              LD      IOCHR       CHECK FOR START OF LINE
0116 01  4C200124          BSC  L  NOTST,Z
0118 20  176558F1  TEST3   LIBF    PRNT1       START OF LINE, CHECK
0119 0   0000              DC      /0000       PRINTER NOT BUSY
011A 0   70FD              MDX     TEST3
011B 0   6A06              STX  2  SAV2&1
011C 0   62C3              LDX  2  -61         CLEAR I/O AREA
011D 0   C015              LD      BLNKS
011E 0   D279        STRBL STO  2  IOBUF&61
011F 0   7201              MDX  2  1
0120 0   70FD              MDX     STRBL
0121 01  66000123    SAV2  LDX  L2 *           RESTORE XR2
0123 0   7009              MDX     LINBK
0124 0   900D        NOTST S       LIMIT       CHECK FOR LINE OVERFLOW
0125 01  4C28012D          BSC  L  LINBK,&Z
0127 30  09595100          CALL    IOND        OVERFLOW  CLEAR INTERRUPTS
0129 0   C007              LD      EE1         AND WAIT WITH ERROR DISPLAY
012A 0   3000              WAIT
012B 00  4C000038          BSC  L  /38         EXIT
012D 01  7401014F  LINBK  MDX  L  IOCHR,1      NO OVERFLOW  INCREMENT
012F 01  4C800114          BSC  I  LINE        CHARACTER POINTER
0131 0   EE01        EE1   DC      /EE01
0132 0   0079        LIMIT DC      121         MAX. NO. OF CHARACTERS
0133 0   4040        BLNKS DC      /4040       BLANKS
                     *     END OF LINE ROUTINE
                     *     IOFIL ROUTINE. ENTERS CHARACTER FROM
                     *     ACCUMULATOR IN FORM 00XX INTO IOBUF
                     *     IN POSITION GIVEN BY IOCHR
0134     0001        IOFIL BSS     1
0135 0   D015              STO     HOLD        SAVE ACCUMULATOR
0136 0   C018              LD      IOCHR       CALCULATE ADDRESS TO
0137 0   1881              SRT     1           STORE CHARACTER
0138 0   8015              A       IOADD
0139 0   D006              STO     AND&1
013A 0   1091              SLT     17          MOVE REMAINDER INTO CARRY
013B 0   C011              LD      MASKL
013C 01  4C02013F          BSC  L  AND-,C      IF CHARACTER NO. IS EVEN
013E 0   1808              SRA     8           MOVE MASK TO RIGHT SIDE
013F 01  E4000141  AND     AND  L  *           DELETE UNWANTED CHARACTER
0141 0   D00A              STO     TEMP1       FROM DESTINATION WORD
0142 0   C008              LD      HOLD        GET CHARACTER TO INSERT
```

```
0143 01 4C020146          BSC  L  OR,C        IF CHARACTER NO. IS EVEN
0145 0  1008              SLA     8           MOVE CHARACTER TO LEFT
0146 0  E805        OR    OR      TEMP1       COMBINE
0147 01 D4800140          STO  I  AND&1       STORE IN DESTINATION WORD
0149 01 4C800134          BSC  I  IOFIL       RETURN
014B    0001        HOLD  BSS     1
014C    0001        TEMP1 BSS     1
014D 0  FF00        MASKL DC      /FF00
014E 0  003C        IOADD DC      IOBUF
014F    0001        IOCHR BSS     1           CHARACTER IN LINE
                    *     END OF IOFIL ROUTINE
                    *     PRINT ROUTINE
                    *     GETS CARRIAGE CONTROL CHARACTER AND
                    *     SETS UP APPROPRIATE SKIP
                    *     PRINTS LINE. RESETS CHARACTER POINTER
                    *     IF START OF LINE, TESTS PRINTER READY
                    *
0150    0001        PRINT BSS     1
0151 0  C0FD              LD      IOCHR
0152 01 4C080162          BSC  L  TEST2,&
0154 00 C400003C          LD   L  IOBUF       NOT START OF LINE
0156 0  901F              S       BLZ         CARRIAGE CONTROL CHARACTER
0157 01 4C20015B          BSC  L  NUM,Z
0159 0  C01D              LD      SKP1        IT IS ZERO, SKIP 1 LINE
015A 0  7003              MDX     SKIP
015B 01 4C080165    NUM   BSC  L  OUTLN,&
015D 0  1008              SLA     8           IT IS A CHANNEL NO
015E 0  E819        SKIP  OR      CNTWD       SKIP AS REQUIRED
015F 0  D001              STO     CNTRL
0160 20 176558F1          LIBF    PRNT1
0161 1  0162        CNTRL DC      *
0162 20 176558F1    TEST2 LIBF    PRNT1       WAIT UNTIL SKIP COMPLETE
0163 0  0000              DC      /0000
0164 0  70FD              MDX     TEST2
0165 0  C0E9        OUTLN LD      IOCHR       PRINT LINE
0166 0  1801              SRA     1
0167 00 D400003C          STO  L  IOBUF
0169 01 4C200171          BSC  L  P1CAL,Z     TEST FOR ZERO WC
016B 0  C0A7              LD      ONE         IF COUNT IS ZERO
016C 00 D400003C          STO  L  IOBUF       MAKE COUNT 1 AND
016E 0  C0C4              LD      BLNKS       PUT BLANKS IN IOBUF & 1
016F 00 D400003D          STO  L  IOBUF&1
0171 20 176559E7    P1CAL LIBF    PRNXX       PRINT WITH OVERLAP
0172 0  1810              SRA     16
0173 0  D0DB              STO     IOCHR
0174 01 4C800150          BSC  I  PRINT
0176 0  40F0        BLZ   DC      /40F0       BLANK AND ALPHA ZERO
0177 0  0D00        SKP1  DC      /0D00       CONTROL DIGIT FOR SKIP
0178 0  3000        CNTWD DC      /3000       CONTROL FUNCTION
                    *     END OF PRINT ROUTINE
                    *     ITYPE ROUTINE
                    *     CONVERTS INTEGER VARIABLE TO DECIMAL
                    *     TRANSFERS DIGITS TO I/O BUFFER
                    *     IF FIELD WIDTH TOO SMALL, FILLS FIELD
                    *     WITH ***
0179 0  C0D5        ITYPE LD      IOCHR       SAVE POSITION OF
017A 0  D03C              STO     SAVCR       CHARACTER POINTER
017B 01 C4000074          LD   L  FWIDE       MARK POSITION OF RIGHT
017D 00 D4000002          STO  L  2           END OF FIELD, ALSO TEST
```

```
017F 01 44000114    ILP   BSI  L  LINE       FOR SUFFICIENT SPACE
0181 0  72FF              MDX   2 -1         AVAILABLE IN LINE
0182 0  70FC              MDX     ILP
0183 0  C0CB              LD      IOCHR
0184 0  D033              STO     FEND
0185 01 C4800111          LD   I  VARAD       GET ABSOLUTE VALUE OF
0187 01 4C10018C          BSC  L  POSV,-      VARIABLE
0189 0  1810              SRA     16          IF NEGATIVE,CHANGE SIGN
018A 01 94800111          S    I  VARAD
018C 0  D02C        POSV  STO     VABS
018D 0  C02B        DIVL  LD      VABS        DIVIDE BY 10
018E 0  1890              SRT     16
018F 0  A82A              D       TEN
0190 0  D028              STO     VABS
0191 0  1090              SLT     16          CONVERT REMAINDER TO EBCDIC
0192 0  E828              OR      FZ
0193 0  40A0              BSI     IOFIL       STORE
0194 0  C0BA              LD      IOCHR       MOVE CHARACTER POINTER LEFT
0195 0  9028              S       ONE2
0196 0  D0B8              STO     IOCHR
0197 01 740001B9          MDX  L  VABS,0      TEST FOR ZERO QUOTIENT
0199 0  700E              MDX     NEXTD
019A 01 C4800111          LD   I  VARAD       TEST FOR NEGATIVE VARIABLE
019C 01 4C1001A4          BSC  L  RESET,-
019E 0  C0B0              LD      IOCHR       TEST FOR SPACE FOR SIGN
019F 0  9017              S       SAVCR
01A0 01 4C0801AB          BSC  L  FILL,&
01A2 0  C019              LD      MINUS       ENTER MINUS SIGN
01A3 0  4090              BSI     IOFIL
01A4 0  C013        RESET LD      FEND        RESET CHARACTER POINTER
01A5 0  D0A9              STO     IOCHR
01A6 01 4C000007          BSC  L  CALL
01A8 0  900E        NEXTD S       SAVCR       NEXT DIGIT. TEST FOR SPACE
01A9 01 4C30018D          BSC  L  DIVL,-Z     AVAILABLE
01AB 01 C4000074    FILL  LD   L  FWIDE       NO SPACE. FILL FIELD WITH
01AD 00 D4000002          STO  L  2           ***
01AF 01 44000114    FLP   BSI  L  LINE
01B1 0  C00B              LD      ASTER
01B2 0  4081              BSI     IOFIL
01B3 0  72FF              MDX   2 -1
01B4 0  70FA              MDX     FLP
01B5 01 4C000007          BSC  L  CALL
01B7    0001        SAVCR BSS     1           RIGHT END OF PREV. FIELD
01B8    0001        FEND  BSS     1           RIGHT END OF THIS FIELD
01B9    0001        VABS  BSS     1           ABSOLUTE VALUE OF VARIABLE
01BA 0  000A        TEN   DC      10
01BB 0  00F0        FZ    DC      /00F0       CONVERTS DIGITS TO EBCDIC
01BC 0  0060        MINUS DC      /0060       MINUS SIGN
01BD 0  005C        ASTER DC      /005C
01BE 0  0001        ONE2  DC      1
                    *     END  OF ITYPE ROUTINE
                    *     FETYP ROUTINE
                    *     F OR E TYPE.  FILLS FIELD WITH ***
01BF 01 C4000074    FETYP LD   L  FWIDE
01C1 0  E001              AND     NOD         MASK OUT D PART OF SPEC.
01C2 0  70EA              MDX     FILL&2      JOIN NO SPACE BRANCH
01C3 0  007F        NOD   DC      /007F
                    *     END OF FETYP ROUTINE
01C4 0  1000        PRNTZ NOP                 DUMMY ENTRY NOT USED
```

PA

```
01C5 0  70FE            MDX     *-2
01C6    0001            BSS     1
01C8                    END
```

NO ERRORS IN ABOVE ASSEMBLY.

```
// ASM
*LIST SOURCE PROGRAM
                        *       PRNXX PRINTS IN OVERLAPPED MODE
                        *       THIS SUBROUTINE NOT OVERLAID DURING SOCALS
                                LIBR
0000     176559E7               ENT     PRNXX
0000 0   1000           PRNXX  NOP
0001 01  C4000003              LD    L *
0003 0   D005                  STO     BACK&1
0004 20  176558F1              LIBF    PRNT1
0005 0   2000                  DC      /2000
0006 0   003C                  DC      IOBUF
0007 1   000A                  DC      ENDPG
0008 01  4C00000A       BACK   BSC   L *
000A 1   000B           ENDPG  DC      *          SKIP TO CHANNEL 1 IF
000B 01  4C80000A              BSC   I ENDPG      END OF PAGE
003C                    IOBUF  EQU     /003C      FORTRAN I/O BUFFER
000E                           END

    NO ERRORS IN ABOVE ASSEMBLY.
```

```
// ASM                                                                    IDEAL48A
*LIST                                                                     IDEAL48B
0000     09595100          ENT     IOND      SUBROUTINE NAME            IDEAL48C
                  *CALL    IOND    NO PARAMETERS                        IDEAL48D
                  *CALL    IOND    ALLOWS I/O OPERATIONS TO END BEFORE A IDEAL48E
                  *                PAUSE OR STOP IS ENTERED             IDEAL48F
0000     0001     IOND     BSS     1         ARGUMENT ADDRESS           IDEAL48G
0001 00  74000032 IOPND    MDX  L  50,0      ANY INTERRUPTS PENDING     IDEAL48H
0003 0   70FD             MDX     IOPND                                 IDEAL48I
0004 01  4C800000 BACK     BSC  I  IO,ID                                 IDEAL49
0006              END                                                    IDEAL49A
```

NO ERRORS IN ABOVE ASSEMBLY.

THE SIG

```
// JOB T
// FOR                                                                  CSP12820
** SAMPLE PROBLEM 2                                                     CSP12830
*IOCS(1132 PRINTER)
*LIST ALL
* NAME SMPL2                                                           CSP12840
* ONE WORD INTEGERS                                                    CSP12850
* EXTENDED PRECISION                                                   CSP12860
C  DEMONSTRATE USE OF OVERLAPPED PRINT SUBROUTINE
C  THIS PROGRAM IS IDENTICAL TO COMMERICAL SUBROUTINE PACKAGE
C  SAMPLE PROGRAM NO. 2
C  NOTE INPUT DATA CARDS CSP13960 TO CSP14030 ARE NOT REQUIRED
C-----THE INPUT IS MADE UP OF A MASTER CARD FOLLOWED BY THE TRANSACTION CSP12880
C-----CARDS FOR EACH CUSTOMER. WE WANT TO PRINT AN INVOICE AND PRINT A  CSP12890
C-----NEW MASTER CARD FOR EACH CUSTOMER.                               CSP12900
      DIMENSION  INCRD(82),IPRNT(6),IOTCD(80),ISTOP(5),IWK(13),ISUM(8),
     1IEROR(6)
      READ(99,101) ISTOP
  101 FORMAT('I S T O P')
      READ(99,106) IEROR
  106 FORMAT('E R R O R   ')
      J=2                                                              CSP13010
      INCRD(81)=16448                                                  CSP13020
      INCRD(82)=5440                                                   CSP13030
1     I=0                                                              CSP13040
      L=0                                                              CSP13050
      M=0                                                              CSP13060
      CALL READ(INCRD,1,80,J)                                          CSP13070
      IF(J-1)  22,2,2                                                  CSP13080
2     IF(NCOMP(INCRD,1,5,ISTOP,1))   3,22,3                            CSP13090
3     CALL NZONE(INCRD,70,5,K)                                         CSP13100
      IF(K-1)   26,4,26                                                CSP13110
    4 WRITE(3,102) (INCRD(II),II=1,60)
  102 FORMAT('1',20A1/(' ',20A1))
      READ(99,103) IWK
  103 FORMAT('        ,     $ .     C R ')
      CALL EDIT(INCRD,61,68,IWK,1,13)
      WRITE(3,104) IWK
  104 FORMAT(///10X,'QTY',10X,'NAME',46X,'AMT'/23X,'PREVIOUS BALANCE',
     128X,13A1)
      LINES=8
40    CALL A1DEC(INCRD,61,68,L)                                        CSP13250
      IF(L)   5,5,23                                                   CSP13260
5     CALL MOVE(INCRD,61,68,ISUM,1)                                    CSP13270
      CALL MOVE(INCRD,1,80,IOTCD,1)                                    CSP13280
6     CALL READ(INCRD,1,80,J)                                          CSP13290
      IF(J-1)   22,7,7                                                 CSP13300
7     CALL NZONE(INCRD,70,5,K)                                         CSP13310
      IF(K-1)   18,19,8                                                CSP13320
8     IF(K-2)   18,9,18                                                CSP13330
9     IF(NCOMP(INCRD,1,20,IOTCD,1))   18,10,18                         CSP13340
   10 READ(99,110) IPRNT
  110 FORMAT('    ,     0 ')
      CALL EDIT(INCRD,49,52,IPRNT,1,6)
      READ(99,103) IWK
      CALL EDIT(INCRD,41,48,IWK,1,13)
      IF(LINES-55) 31,31,17
   31 WRITE(3,105)  IPRNT,(INCRD(II),II=21,40),IWK
  105 FORMAT(7X,6A1,10X,20A1,24X,13A1)
      LINES=LINES+1
```

SAMPLE PROBLEM 2 PAGE 02

```
11    CALL A1DEC(INCRD,41,48,L)                                    CSP13430
      IF(L)   12,12,14                                             CSP13440
12    CALL ADD(INCRD,41,48,ISUM,1,8,M)                             CSP13450
      IF(M)   13,6,13                                              CSP13460
13    CALL IOND                                                    CSP13470
      STOP 777                                                     CSP13480
14    CALL NZONE(INCRD,L,4,N1)                                     CSP13490
      N1=0                                                         CSP13500
      CALL A1DEC(INCRD,L,L,N1)                                     CSP13510
      IF(N1)   16,16,15                                            CSP13520
15    CALL IOND                                                    CSP13530
      STOP 666                                                     CSP13540
16    CALL DECA1(INCRD,41,48,L)                                    CSP13550
      L=0                                                          CSP13560
      GO TO 11                                                     CSP13570
   17 WRITE(3,109)
  109 FORMAT('1',9X,'QTY',10X,'NAME',46X,'AMT')
      LINES=1
      GO TO 31
18    CALL TYPER(IEROR,1,5)                                        CSP13620
      CALL TYPER(INCRD,1,82)                                       CSP13630
      GO TO 6                                                      CSP13640
19    CALL DECA1(ISUM,1,8,L)                                       CSP13650
      IF(L)   20,21,20                                             CSP13660
20    CALL IOND                                                    CSP13670
      STOP 555                                                     CSP13680
   21 READ(99,103) IWK
      CALL EDIT(ISUM,1,8,IWK,1,13)
      CALL MOVE(ISUM,1,8,IOTCD,61)
      CALL TYPER(IOTCD,1,80)
      WRITE(3,107) IWK
  107 FORMAT(//23X,'TOTAL',39X,13A1)
      CALL TYPER(INCRD,81,82)
      GO TO 1
   22 READ(99,108) IWK
  108 FORMAT('E N D   O F   J O B ')
      CALL TYPER(IWK,1,10)
      CALL IOND
      STOP 111
23    CALL NZONE(INCRD,L,4,N1)                                     CSP13820
      N1=0                                                         CSP13830
      CALL A1DEC(INCRD,L,L,N1)                                     CSP13840
      IF(N1)   25,25,24                                            CSP13850
24    CALL IOND                                                    CSP13860
      STOP 444                                                     CSP13870
25    CALL DECA1(INCRD,61,68,L)                                    CSP13880
      L=0                                                          CSP13890
      GO TO 40                                                     CSP13900
26    CALL TYPER(IEROR,1,5)                                        CSP13910
      CALL TYPER(INCRD,1,82)                                       CSP13920
      GO TO 1                                                      CSP13930
      END                                                          CSP13940
```

VARIABLE ALLOCATIONS

```
 INCRD=0051  IPRNT=0057  IOTCD=00A7  ISTOP=00AC  IWK  =00B9  ISUM =00C1  IEROR=00C
 M    =00CB  K    =00CC  II   =00CD  LINES=00CE  N1   =00CF
```

STATEMENT ALLOCATIONS

```
 101  =00F8  106  =00FF  102  =0107  103  =0111  104  =0120  110  =013E  105  =014
```

```
SAMPLE PROBLEM 2                                                    PAGE
1    =01A5  2    =01BD  3    =01C6  4    =01D2  40   =0202  5    =020C  6
10   =0243  31   =0265  11   =0288  12   =0292  13   =029F  14   =02A3  15
19   =02DD  20   =02E7  21   =02EB  22   =0313  23   =0322  24   =0336  25

FEATURES SUPPORTED
 ONE WORD INTEGERS
 EXTENDED PRECISION
 IOCS

CALLED SUBPROGRAMS
 READ    NCOMP   NZONE   EDIT    A1DEC   MOVE    ADD     IOND    DECA1   TYPER
 SFIO    SIOAI   SIOIX   SUBSC   STOP    PRNTZ

INTEGER CONSTANTS
    99=00D2      2=00D3   16448=00D4   5440=00D5      0=00D6      1=00D7     80
    60=00DC     61=00DD      68=00DE     13=00DF      8=00E0     20=00E1     49
    48=00E6     55=00E7      21=00E8     40=00E9     777=00EA      4=00EB    666
    10=00F0    111=00F1     444=00F2   1911=00F3   1638=00F4   1365=00F5    273

CORE REQUIREMENTS FOR SMPL2
 COMMON       0  VARIABLES     210  PROGRAM     640

END OF COMPILATION
```

// XEQ L CSP13950

R 47 0B4E (HEX) WORDS AVAILABLE

CALL TRANSFER VECTOR

CARRY	1123
NSIGN	10D7
FILL	10B2
TYPER	0C05
DECA1	0B5A
IOND	0B3E
ADD	0A7E
MOVE	0A40
A1DEC	09D2
EDIT	08AF
NZONE	07FD
NCOMP	07AF
READ	075D

LIBF TRANSFER VECTOR

HOLL	13AE
PRTY	135E
EBPA	130E
TYPE0	11E6
EBPRT	1180
RPACK	1050
SUBIN	1090
ARGS	1022
SWING	106F
SPEED	0ED6
CARD1	0DE0
PRNXX	0DD2
PRNT1	0C50
STOP	0B44
SCOMP	0547
SIOIX	0533
SUBSC	087C
SWRT	05A9
SIOAI	053A
SRED	058A
ESTO	06E0
ELD	06F6
PRNTZ	06D8
SFIO	0514
DISKZ	00F4

SYSTEM ROUTINES

ILS04	1401
ILS02	141D
ILS01	142F
ILS00	1441

0338 (HEX) IS THE EXECUTION ADDR.

DAVES MARKET
1997 WASHINGTON ST.
NEWTOWN, MASS. 02158

QTY	NAME	AMT
	PREVIOUS BALANCE	$111.29
8	SUGAR - BAGS	$21.02
11	CHICKEN SOUP - CASES	$38.76
10	TOMATO SOUP - CASES	$30.11
8	SUGAR RETURNED	$21.02CR
6	COOKIES - CASES	$45.21
17	GINGER ALE - CASES	$52.37
17	ROOT BEER - CASES	$52.37
17	ORANGE ADE - CASES	$52.37
17	CREME SODA - CASES	$52.37
17	CHERRY SODA - CASES	$52.37
17	SODA WATER - CASES	$52.37
25	DOG FOOD - CASES	$101.26
25	CAT FOOD - CASES	$101.26
10	SOAP POWDER - CASES	$72.89
10	DETERGENT - CASES	$72.89
12	HAM - TINS	$36.75
12	HAM - LOAF	$33.75
12	SALAMI	$33.75
12	BOLOGNA	$33.75
12	CORNED BEEF	$33.75
12	ROAST BEEF	$33.75
1,000	BREAD - LOAF	$150.00
4,000	ROLLS	$150.00
200	MILK - QUARTS	$57.42
100	MILK - HALF GALS	$57.42
50	MILK - GALS	$57.42
100	POTATOES - BAGS	$11.23
100	TOMATOES - LOOSE	$11.23
100	CARROTS - BUNCHES	$11.23
10	DETERGENT - CASES	$72.89
12	HAM - TINS	$36.75
12	HAM - LOAF	$33.75
12	SALAMI	$33.75
12	BOLOGNA	$33.75
12	CORNED BEEF	$33.75
12	ROAST BEEF	$33.75
1,000	BREAD - LOAF	$150.00
4,000	ROLLS	$150.00
200	MILK - QUARTS	$57.42
50	MILK - GALS	$57.42
100	MILK - HALF GALS	$57.42
100	POTATOES - BAGS	$11.23
100	TOMATOES - LOOSE	$11.23
100	CARROTS - BUNCHES	$11.23
10	DETERGENT - CASES	$72.89
12	HAM - TINS	$36.75
1,000	BREAD - LOAF	$150.00
4,000	ROLLS	$150.00

QTY	NAME	AMT
200	MILK - QUARTS	$57.42
100	MILK - HALF GALS	$57.42
50	MILK - GALS	$57.42
100	POTATOES - BAGS	$11.23
100	TOMATOES - LOOSE	$11.23
100	CARROTS - BUNCHES	$11.23
10	DETERGENT - CASES	$72.89
12	HAM - TINS	$36.75
12	HAM - LOAF	$33.75
12	SALAMI	$33.75
12	BOLOGNA	$33.75
12	CORNED BEEF	$33.75
12	ROAST BEEF	$33.75
1,000	BREAD - LOAF	$150.00
4,000	ROLLS	$150.00
200	MILK - QUARTS	$57.42
100	MILK - HALF GALS	$57.42
100	MILK - HALF GALS	$57.42
100	POTATOES - BAGS	$11.23
100	TOMATOES - LOOSE	$11.23
100	CARROTS - BUNCHES	$11.23
10	DETERGENT - CASES	$72.89
12	HAM - TINS	$36.75
	TOTAL	$3,893.25

STANDISH MOTORS
10 WATER STREET
PLYMOUTH, MASS.02296

QTY	NAME	AMT
	PREVIOUS BALANCE	$2,356.36
20	AIR CLEANERS - CASES	$200.03
6	GREASE - BARRELS	$165.24
20	TIRES - 650 X 13	$260.38
50	TIRES - 750 X 14	$900.53
50	TIRES - 800 X 14	$1,012.00
100	GASOLINE CAPS	$99.68
	TOTAL	$4,994.22

FOR PROCEEDINGS

BILL LANE

W.1.5b 1130 COMMERCIAL SUBROUTINES

ABSTRACT:

Version II of the IBM 1130 Commercial Subroutines will be the primary emphasis of this session. The general problem of commercial programming in pure FORTRAN and the history of partial solutions available at this date will be discussed. Version II of 1130 CSP will be compared to Version I and other available commercial routines. Performance in execution time and core requirements as well as current limitations will be presented.

W.I. 7(a)

A Comparison between 2310 and 2311 Disk Storage Systems

by

S. F. Seroussi

Thomas J. Watson
Research Center
P. O. Box 218
Yorktown Hts., N. Y.

A series of programs has been written to allow 1800 users to fully utilize the capabilities of the IBM 2841-2311 disk storage system within the frame work of the Time-Sharing Executive -- TSX, Version 3, Mod 1.

A comparison is made between the 2311 and the 2310 disks in programming techniques and timing.

Physical Characteristics

2310

Capacity - Oxide coated disk with 200 primary, 3 alternate 2 track cylinders capable of storing 521,304 16 bit words (1,042,304 bytes) of which about 512K words are available to the user the rest being used as alternates to defective tracks. Each track is divided into four sectors. A sector is the basic addressable unit with a capacity of 321 words one of which is usually used for sector addressing and error checkout.

Timing - The disk rotates at a speed of 1500 rpm; a revolution taking 40 ms. Word transfer rate is 36 K words/sec.

Cylinder to cylinder access is 15 ms[1] in single steps or 20 for double cylinder steps and a delay of 20 ms must be added to allow the carriage to stabilize itself. Average access time 530 ms.

Between sectors the shortest time available is 235 μsec (plus 27μsec for each word not used). As an example, if we write 310 words in a sector and then continue to write on the next sectors the delay between writes will be a minimum of 235 + 27 x 11 or over half a millisecond.[1]

Programming - The following hardware (machine language) operations are available:

---

[1] Models A1, A2, and A3.

CE Mode

INITIALIZE READ - READ TO MEMORY
READ - CHECK

INITIALIZE WRITE

CONTROL (SEEK FORWARD OR BACKWARD)

SENSE

These operations must be executed one at a time with no chaining allowed and with several restrictions imposed such as the need for a 20 millisecond delay between seeks or an error will result. Only one sector can be processed per I/O command.

Interrupts - Sensing the DSW for each disk unit provides the user with a series of indicators. The occurrence of an Operation Complete, causes an interrupt in the 1800. All other indicators will only be detected together with Operation Complete.

Data Status Word

Bit	Interrupt
0	Any Error
1	Operation Complete
2	Disk Not Ready
3	Disk Busy
4	Carriage Home
5	Parity Check Error
6	Storage Protect Error
7	Data Error
8	Write Select Error
9	Data Overrun
10	Not Used
11	CE Not Ready
12	CE Busy
13	Not Used
14-15	Sector Count

Two or more indicators will be on when an interrupt occurs.

Number of Drives Per System - Standard up to three. RPQ three more.

Typical I/O Command:

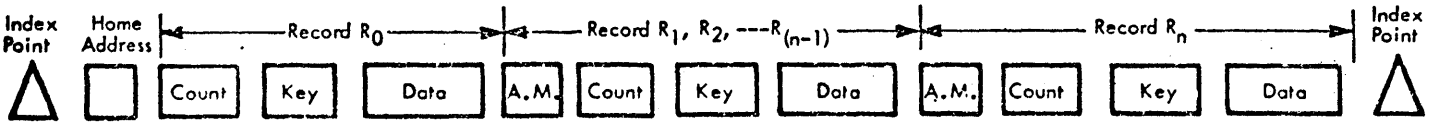
```
            XIO       IOCC1
             .
             .
             .
IOCC1       DC        TABLE     DATA TABLE
            DC        /4D01     WRITE ON DRIVE 1,
                                SECTOR 1
             .
             .
             .
TABLE       DC        321       COUNT
            BSS       321
```

A previous XIO would have moved the arm to the proper cylinder.

2311

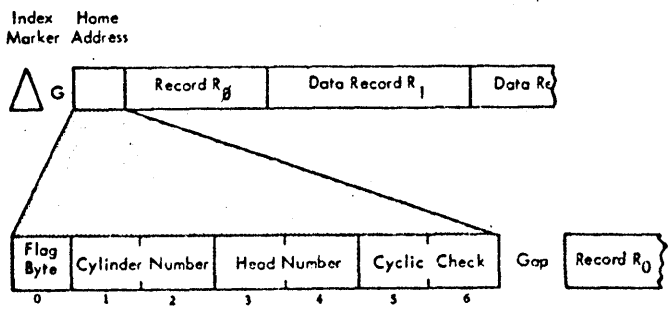
Capacity - 12 oxide coated disks with the ten inside surfaces used for recording data up to 7.5 million 8 bit bytes.

A pack is divided into 200 primary, three alternate cylinders and ten tracks (one for each recording surface) of 3694 words each. Each track can be subdivided into records of variable length as defined by the user. The home address which identifies the track and the first record on the track, RO, usually used to define the condition of the track (and its alternate if needed), are generated with special IBM provided programs with a standard format. A track would have the following configuration:



2311 Track Format

Each of the above areas contains the following information:



Home Address

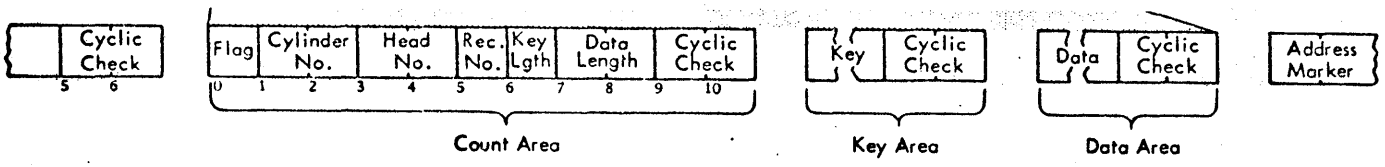
Home Address Flag Byte: The flag byte in the home address is transferred automatically to the using system by performing a read home address operation on the track. The bit significance is:

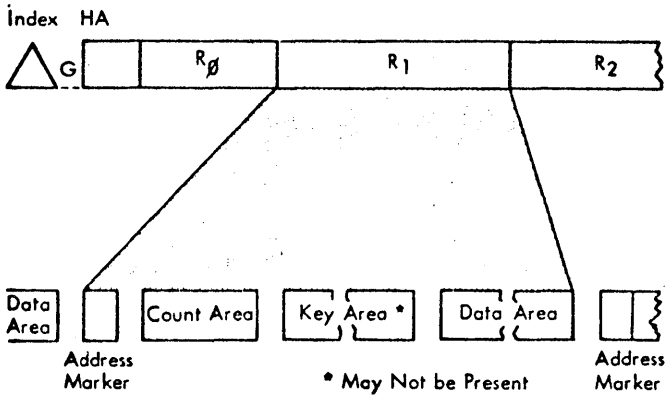
	Bit	Function or Setting	
Flag Byte	0	Zero	
	1	Zero	
	2	Zero	
	3	Zero	
	4	Zero	
	5	Zero	
	6	Track Condition	0 indicates operative track 1 indicates defective track
	7	Track Use	0 indicates primary track 1 indicates alternate track

Flag: Byte 0 of the count area is generated by the 2841 as each record is written. It is not sent from the CPU. Bit 7=1 indicates an alternate track.

	Bit	Function	
Flag Byte	0	0 for even-count records (R0, R_2, R_4, R_6) 1 for odd-count records (R_1, R_3, R_5....) Used by the 2841 to ensure that all address markers (and records) are present. The 2841 signals a missing address marker when two consecutive, identical bits are encountered (unless an index point intervenes).	
	1	Used only with record overflow feature. 0 for all non-overflow records and for the last segment of an overflow record. 1 for each segment, except for the last segment of an overflow record.	
	2	Zero	
	3	Zero	
	4	Zero	
	5	Zero	
	6	Track condition	0 indicates operative track 1 indicates defective track
	7	Track Use	0 indicates primary track 1 indicates alternate track

Bits 6 and 7 are transmitted to the flag bytes of all records on the track from the flag byte of the Home Address of that track by the 2841.





Record $R_1 - R_n$ Format

Storage Units	Track Capacity in Bytes When R_0 is Used as Specified By IBM Programming Systems.	Track Capacity When R_0 is Used for Application Data	Bytes Required by Data Records (KL = Key Length, DL = Data Length)			
			Data Records (except for last record)		Last Record	
			Without Key	With Key	Without Key	With Key
2311	3625	3694	$61 + \frac{537}{512} D_L$	$81 + \frac{537}{512} (K_L + D_L)$	D_L	$20 + (K_L + D_L)$

Number of Bytes per Record when Record R_0 used as specified by IBM Programming Systems.	Number of Equal Length Records Per 2311 Track																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Without Key	3625	1740	1131	830	651	532	447	384	334	295	263	236	213	193	177	162	149	138	127	118
With Key (Includes key bytes + data bytes - $K_L + D_L$)	3605	1720	1111	811	632	512	428	364	315	275	244	217	194	174	158	143	130	119	108	99

Record gaps are of a length determined by the size of the record. Track capacity then, varies with record length as shown in the above figure.

Timing - The disk rotates at 2400 rpm with a revolution time of 25 ms. Access time from cylinder to adjacent

cylinder is 25 ms; maximum access time (from cylinder 000 to 200) is 135 ms and average time for a random access is 75 ms. Once the cylinder is accessed, an average of 12.5 ms is needed to reach the desired record. No delay is encountered in going from track to track because the access mechanism includes one read/write head per track.

Programming - A series of commands each taking 3 1800 words, are available to the user which allow for manipulation of both the control unit and the 2311 drives. These commands can be classified into four groups as recognized by the selector channel:

Cutput Forward (Write, Control)

Input Forward (Read, Sense)

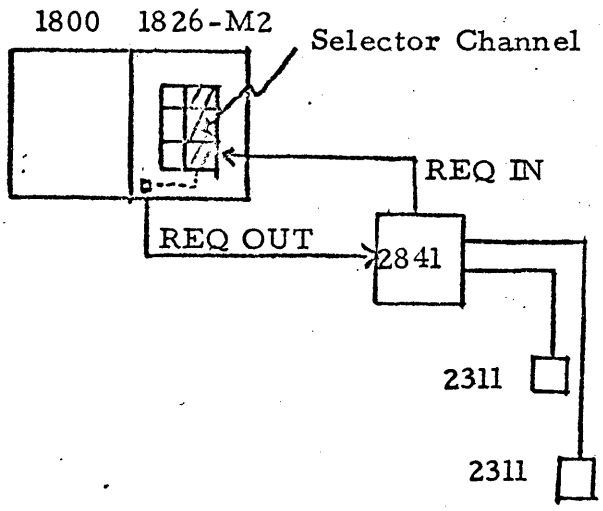
Branching (Transfer in Channel)

Test I/O

There are forty possible command codes[1] some of which allow the user to operate on any of the elements contained in a track or in a whole cylinder. There is no capability to cross cylinder boundaries. This has to be programmed for and is very simple.

Interrupts - Before considering the interrupt capabilities of the 2841-2311 system let us look at a simple diagram of one of these devices connected to an 1800.

1. See Appendix I, pp I.2 for table of commands



From this diagram, it is evident that there are two interfaces of importance; (1) Selector Channel/Control Unit (2841) and (2) Control Unit/I/O Devices (2311).

Selector Channel/Control Unit

(a) If more than one control unit is attached to the selector channel, SENSE ILSW will determine which one of them has interrupted. The machine configuration under consideration has one control unit so there is no need to execute this instruction.

(b) To determine the status of the selector channel, a SENSE CHANNEL STATUS WORD can be executed. It is a similar step to sensing DSW on the 2310. It provides the programmer with the following information in the accumulator:

BIT	STATUS	MEANING
0	Not Operational	Device Addressed not ready or not connected
1	Unit Status Pending	Control Unit and Device Status Waiting for Further Action
2	Program Control Interrupt	Requested by Programmer
3	Program Check	Bad Parity on IOCC - Request Out of Bounds
4	Data Check	Parity Error - Write to Protected Area
5	Interface Check	Hardware Error
6	Incorrect Length	Requested Length and Length in Disk Unequal
7	Channel Busy	Busy - Do not Request Another I/O or Card Check will Occur
8	Unit Operational	Unit Up - Actually In Use

The first four conditions generate an interrupt. Further information can be obtained from the status of the control unit and devices.

Control Unit/Devices

Up to eight 2311s can be attached to a single control unit. To obtain their status, a SENSE UNIT STATUS can be executed. If Unit STATUS PENDING occurred, it is necessary to do this sense in order to clear the interrupt. The 16 bits

of information placed in the accumulator are:

BITS	STATUS	MEANING
0 - 3		Address of Control Unit
4 - 7		Device Address (0 - 7)
8	Attention	Not Used
9	Status Modifier	With 11, Control Unit Busy
10	Control Unit End	Control Unit Available
11	Busy	With 9
12	Channel End (CE)	Indicate Status
13	Device End (DE)	Of Operation
14	Unit Check	Error
15	End Exception	End of File Detected (A Record with Zero Count)

The above bits can be on in several combinations depending upon the operation in progress. In general, CE - DE together and alone, indicate a successful end of operation. Unit Check on the other hand, indicates an error. Three more sense operations are available to assist the programmer in determing the error source:

(a) SENSE CCW ADDRESS - will give the address plus three of the last commands used.

(b) SENSE BYTE COUNTER - will give the residual byte count.

(c) SENSE BYTES - will transfer six bytes of information concerning the unit check from the 2841 to the CPU at an

address specified by the user. The first two bytes contain most of the needed information as shown below:

Byte	Bit	Designation	Significance of "1"
0	0	Command Reject	Indicates that the 2841 has received an invalid operation code, an invalid sequence of commands, an invalid Seek Address, The write portion of the file mask has been violated. (See Set File Mask.)
0	1	Intervention Required	Indicates that the specified file is not physically attached to the system or, if physically attached to the system, it is not available for use because the file motor is not on, a cover interlock is open, etc.
0	2	Bus Out Parity Check	Indicates that the 2841 has detected a parity error during the transfer of a command or data from the channel to the 2841. A parity error detected during command transfer signals a Parity Check.
0	3	Equipment Check	Indicates that an unusual condition is detected in the control or storage unit. Conditions covered by this bit are defined by Sense Byte 2 (See Appendix B).
0	4	Data Check	Indicates that a data check error has been detected in the information received by the 2841 from the storage unit.
0	5	Overrun	Indicates that a chained CCW was issued but it was received too late to be properly executed; or that a byte was received too late (during write operation) to be transferred properly; or the channel did not respond fast enough during a read or search. When writing, the remaining portion of the record area is filled with zeros and the overrun check is generated. When reading or searching, the remaining portion of the record is ignored.
0	6	Track Condition Check	Indicates defective track. A Track Condition check is generated under the following conditions: 1. If an overflow record is being read, written, or searched which overflows to a defective track, the interrupt occurs after the last byte on the previous track has been operated on and before the first byte for the defective track is requested from or sent to the channel. In this case overflow incomplete is also set. 2. If a single track command other than a Search HA, Read HA, or Read R0 is executed on a defective track. 3. If a multiple track command or an overflow operation attempts to switch from an alternate or defective track after an operation has been executed.
0	7	Seek Check	Indicates that the file has been unable to complete a Seek because: 1. The Seek address is outside the valid address boundaries of the storage device. Unused seek address bytes must be a valid address for the device selected. Command Reject is also set. 2. Less than six seek address bytes were sent. Command Reject is also set. 3. The equipment failed, which resulted in the access mechanism going to either the inner or outer stop.
1	0	Data Check in the Count Field	Indicates that cyclic check error has been detected in a Count Area read from the storage device. Data check (bit 4) in byte 0 is also turned on.
1	1	Track Overrun	Indicates that writing has not been completed by the time the index point is detected.
1	2	Cylinder End	Indicates that Cylinder End has been detected, but the CCW Command Chain has not been completed.
1	3	Invalid Sequence	Indicates that an attempt has been made to execute an invalid sequence of CCWs or that two Set File Mask commands appear in the same command chain. Valid command sequences are defined in the Write and Erase command descriptions. Command Reject (Byte 0 bit 0) is also set when an Invalid sequence is detected.
1	4	No Record Found	Indicates that while executing a chain of CCWs the 2841 has detected two Index Points without completing an intervening command to read or write or search the Data Area, Read Home Address, or Read R0. It is also set in conjunction with Missing Address Marker if there is no data on the track. No Record Found is never set if the Multi-Track bit in the command (Bit 0) is on. No Record Found will be posted if the address marker in front of the last physical record on the track is not detected.
1	5	File Protected	Indicates that a command was issued contrary to the file mask. The Command Reject bit is also set by this condition, if the operation violates the write portion of the file mask.
1	6	Missing Address Marker	A missing Address Marker, which may indicate a missing record is detected during the execution of command or chain of commands which operates on successive Count Areas on a track. The condition detected is two successive records on a track with equal bit conditions in bit 0 of the Flag bytes, with no intervening Index Point. Missing Address Marker is set in conjunction with No Record Found if there is no data on the track.
1	7	Overflow Incomplete	This bit is used with the Record Overflow special feature. It is set with other indicators to signal conditions as follows: **Condition** — **Sets Overflow Incomplete and Other Indication** Overflow to a defective track — Track Condition (Byte 0, bit 6) Overflow from an alternate track — Track Condition (Byte 0, bit 6) Data check in data area of overflow record other than last segment. — Data check (Byte 0, bit 4) Overflow to File Protected boundary — File Protected (Byte 1, bit 5) Overflow to wrong track (Head number unequal) — Seek Check (Byte 0, bit 7)

Sense Bytes 0 and 1

Typical I/O Command: READ HOME ADDRESS; CYL 180-HEAD 8

```
          .
          .
          XIO   SIO       START I/O
          .
          .
          .
SIO       DC    CCW       START I/O - IOCC
          DC    /9502     AREA CODE 18 - UNIT 2
          .
          .
          .
CCW  ⎧    DC    5         SEEK AND CHAIN
     ⎨    DC    /4007     FOR RECORD SPECIFIED
     ⎩    DC    SEEK      AT SEEK ADDRESS

     ⎧    DC    6         READ HOME
     ⎨    DC    /001A     ADDRESS INTO
     ⎩    DC    HOME      THIS ADDRESS

SEEK      DC    0
          DC    180       CYLINDER
          DC    8         HEAD
HOME      BSS   3
```

Software

The available software for the 2310 is well known to most users. In brief, there are programs to initialize the disk cartridges, test them for errors, label and define length of LET and FLET tables, and dump their contents when needed. Both FORTRAN and ASSEMBLY LANGUAGE routines are available to read and write to almost any area on the 2310 disks. System routines will define and label at your request buffer areas and will assign sector addresses. All the user has to do is punch a define file card, or a STOREDATA card and the system will do the rest. Moreover, all of this is usually done with re-entrant subroutines enabling the user to be free of worries about using a device from different interrupt levels. The only penalty paid, as the sample program will show, is time and overhead costs. All 2310 routines eventually call an I/O subroutine, DISKN which has been charged with the responsibility of controlling all I/O to the 2310s and servicing the resulting interrupts. It should be noted that FORTRAN calls are not overlapped, i. e., control is returned to the user after the operation has been completed successfully or not.

The software available for the 2311 as provided

is restricted to diagnostic programs and a disk initialization program. TSX does not support the 2841-2311 but programs have been written which allow the 1800 FORTRAN and ASSEMBLY LANGUAGE user to fully utilize the 2841-2311 capabilities. These routines are at the present time being submitted as Type IV programs.(1) Work in switching TSX to the 2311 is also being contemplated. The structure of these routines is similar to those used to service the 2310s but three differences must be mentioned:

(1) The routines are not re-entrant. Given the speed of data transmission, it is possible to mask any interrupts while the 2311 routine is generating a chain of commands prior to executing the I/O. The time delay is comparable to the time it would take to save and restore all needed parameters if made re-entrant.

(2) The I/O can be overlapped. The return to the user is made immediately as soon as the I/O is initiated. The user has the option to examine the final interrupt to determine if the operation was successful or not or to ignore this fact completely and continue with his program.

(3) The different I/Os are requested through calling sequences. There is no special WRITE or READ as in the

1. See Appendix II for further explanation and examples.

2310, and most of the parameters in FORTRAN must be fixed integers. There are 10 calls that can be used to generate a sequence of instructions that will perform a certain I/O on the 2311 and two auxiliary calls; one that allows the user to write his own command chain and the other a busy test routine. In machine language, a third special call allows the user to get to a table of parameters and indicators saved by the 2311 ISS. As in the case of the 2310, the 2311 calling sequences execute all I/O through a subroutine, DISKZ.

Having provided a brief sketch of the software available for both 2310 and 2311 disk storage devices, it is possible now to look at the programs run to obtain some timing comparisons. The philosophy used was very simple; write and read back a fixed length record to both disk units under similar conditions and record the lapsed time between start and finish.

TIMING

Routine	WRITE-READ I/O	Time 2310	Time 2311	Language
TIME1	3200 words	7.188 sec.	.322 sec.	FORTRAN-TSX
TIME2	3200 words	1.609 sec.		ASM-TSX
TIME3	3200 words		.192 sec.	ASM-TSX
$TIME	320 words		.262 sec.	ASM-ABS
TIME4	320 words	.918 sec.		ASM-ABS

It should be pointed out that a data area for use of all these routines was set up in FLET and unprotected by means of DWRAD. This time was not included in the computations. Also, the time shown for $TIME does not correspond to a single write of 320 words but to five repeated attempts to write the same record. This was caused by an error in one of the routines used and shows the timing effect of retrying an operation that failed with a recoverable error (in this case unrecoverable).

It is evident that the use of FORTRAN for disk I/O instead of Assembly Language is costly for the 2310 but that in the case of the 2311 storage devices, at the present time, the difference is quite tolerable.

PHYSICAL CHARACTERISTICS

2310		2311
1.02	STORAGE CAPACITY (MILLION BYTES)	7.5
530.00	HIGH SPEED ACCESSIBILITY (milliseconds average)	75.0
72.00	DATA TRANSFER (KILO BYTES)	156.0
3	MULTIPLE UNIT GROWTH	8
203	NUMBER OF CYLINDERS	203
2	NUMBER OF TRACKS/CYLINDER	10
5.1	CYLINDER CAPACITY (THOUSAND BYTES)	37.0
SECTOR	BASIC ADDRESSABLE UNIT	RECORD
640	MAXIMUM CAPACITY OF BASIC UNIT (BYTES)	3900

I.1

APPENDIX II

II.1

The 2841 and 2311 Subroutine Package

The 2841-2311 subroutines were written to allow FORTRAN and/or Assembly Language users to avail themselves of these high speed storage devices.

The aim was to organize the routines along the lines of the 2310 support programs. To accomplish this, a set of three subprograms were developed:

1) The interrupt service subroutine (ISS), DISKZ, that handles all I/O requests through its subroutine entry point and takes care of all resulting interrupts informing the user, if he cares to know, of the status of the last I/O executed.

2) Ten calling sequences which operate upon all or any element of a record, track or cylinder (see Table I).

3. One additional subroutine with three entry points:
   (a) One entry point, DISKX, which allows the user to write his own command chain and calls DISKX to execute the I/O and handle the interrupts.
   (b) An entry point T2311 to a routine busy test.

An indicator is set to zero or one according to the busy status of DISKZ

4) A dummy entry point X8Y9ϕ to a table of parameters and indicators kept by DISKZ.

DISKZ performs the same functions for the 2311 as DISKN does for the 2310. The other subroutines determine which operation is to be performed by the 2311.

TABLE I

2311 CALLS AVAILABLE UNDER 1800 TSX

CALL WR	WRITE COUNT, KEY AND DATA
CALL WRO	WRITE SPECIAL COUNT KEY AND DATA
CALL WKD	WRITE KEY AND DATA
CALL WD	WRITE DATA
CALL RHA	READ HOME ADDRESS
CALL RRO	READ TRACK DESCRIPTOR RECORD
CALL RCKD	READ COUNT, KEY AND DATA
CALL RC	READ COUNT
CALL RKD	READ KEY AND DATA
CALL RDD	READ DATA
CALL DISKX	DIRECT CALL TO DISKZ
CALL T2311	DISKZ BUSY TEST
CALL X8Y9ϕ	ENTRY POINT AT BEGINNING OF TABLE KEPT BY DISKZ (DUMMY CALL SHOULD NEVER BE EXECUTED)

All these calls have a variable number of parameters. For further information contact:

R. C. Yens
The Mitre Corporation
Command and Management Systems
Planning and Engineering, D-74
P. O. Box 208
Bedford, Massachusetts 01730

APPENDIX III

```
C       EXAMPLE OF 2311 CALLING SEQUENCE IN FORTRAN
C
        EXTERNAL RET
        INTEGER A(453),B(453)
C       ESTABLISH ADDRESS OF RECORD TO BE WRITTEN BY APPENDING TO DATA
C       THE CYLINDER,HEAD AND RECORD NUMBER DESIRED (KEEPING IN MIND
C       THAT ARRAYS ARE STORED FROM HIGH TO LOW CORE IN THE IBM 1800)
C
C       CYLINDER
C
        A(453)= 100
C
C       HEAD
C
        A(452)= 6
C
C       RECORD NUMBER AND KEY LENGTH IN THE FORMAT RRKK MUST BE GIVEN
C       IN THE DECIMAL EQUIVALENT OF THE HEXADECIMAL NUMBER
C       IN THE EXAMPLE THERE WILL BE NO KEY SO THAT FOR RECORD NUMBER 1,
C       THE HEXADECIMAL NUMBER WILL BE 0100 OR DECIMAL 256
        A(451)= 256
C
C       WRITE COUNT KEY AND DATA
C
        CALL WR(RET,1,1,A(453))
C
C       READ COUNT KEY AND DATA
C
        CALL RCKD(RET,1,100,6,1,B(453))
C
C       WAIT FOR END OF OPERATION
C
 10     CALL T2311(1)
        IF(1)10,15,10
 15     CALL DMPP(A(1), B(453))
 20     CALL EXIT
        END
```

```
C
C       CALL RET (KK) IS A USER WRITTEN SUBROUTINE TO HANDLE ALL NORMAL
C       AND ERROR ROUTINES - KK IS AN INDICATOR PASSED TO THE USER
C       BY THE INTERRUPT HANDLING SUBROUTINE, DISKZ
C
C
      SUBROUTINE RET(KK)
      CALL DMPP(KK)
      RETURN
      END
```

```
0000    045175C0            ENT     DMPP
                    *       DMPP SHOULD BE A USER WRITTEN SUBROUTINE
                    *       TO DUMP IN HEXADECIMAL THE ARRAYS DELIMITED
                    *       BY THE GIVEN PARAMETERS
0000 0  0000        DMPP    DC      *-*
0001 01 74020000            MDX  L  DMPP,&2
0003 01 4C800000            BSC  I  DMPP
0006                        END     DMPP
```

References

1. I/O Support Routines for the IBM 2311 under IBM 1800 TSX, S. F. Seroussi, November 24, 1967, The Mitre Corporation, Bedford, Massachusetts.

2. IBM Selector Channel - Principle of Operations, RPQ CO8037, J. B. Sampson and W. L. Gillette, Jr., IBM, Special Systems Development, San Jose, California, Form # L26-2034-0.

3. IBM System/360 Component Descriptions, Form # A26-5988-4.

4. IBM 1800 Data Acquisition and Control System Data Processing Input/Output Units, Form # A26-5969-3.

W.1.1.
(b)

PROCESS CONTROL IN NATURAL GAS TRANSMISSION

A.A. DOULOFF
Manager, Systems & Data Processing
Trans-Canada Pipe Lines Limited

Presented to the COMMON meeting
San Francisco
December, 1967

ABSTRACT

Process Control in Natural Gas Transmission

This paper will present the progress made by Trans-Canada Pipe Lines Limited, Toronto, in the field of process control computers. A brief description is given of the feasibility study prior to ordering the computer, the organization of the implementation team and the methods of implementation.

The purpose of installing the process control computer at Trans-Canada is to save fuel by more efficient operation of compressor stations, and to guide the dispatcher into better control of the line, thus allowing more throughput at the same or better operating cost. The computer is not closed loop, but accepts telemetered data from all compressor stations on a priority interrupt basis, optimizes the line by means of an on-line simulation program and then informs the dispatcher by typewriter output what changes, if any, to make to achieve optimal operation.

A description is given of the telemetering system that feeds the computer and of the simulation/optimization program that is used to control the line. The computer, the IBM 1800, was installed in June 1967.

PROCESS CONTROL IN NATURAL GAS TRANSMISSION

1. Introduction

As pipelines become more and more complicated, it is becoming increasingly difficult for the dispatcher to run his line with a high level of efficiency. This, of course, does not reflect on the dispatcher's ability but rather is a direct result of the tremendous number of variables that have to be controlled, especially in a line that is running at a high load factor. To make an intelligent analysis of the line the dispatcher has to know the flow through each compressor station, the horsepower available, the number of units running at each station, RPM of the engines, surge restrictions and the sales pattern.

The answer to this problem has to lie with the on-line process control computer. There are many fast, reliable and relatively inexpensive computers that, when connected to a good telemetering system, can provide answers quickly and accurately to the dispatcher. I would like to discuss briefly the results of having installed a 1710 in September 1965, and later an 1800 in June 1967, at Trans-Canada Pipe Lines Limited.

2. Description of T.C.P.L.'s System

Trans-Canada Pipe Lines Limited owns and operates one of the longest natural gas transmission lines in the world. It stretches 2,300 miles from the Alberta-Saskatchewan border to Montreal and to export

markets in the United States. In the domestic market it sells gas to 11 distribution companies.

Trans-Canada's main activity is the purchase, transportation and sale of natural gas. The original system was completed on October 10, 1958. As a result of the wide acceptance of this fuel, the system has been constantly expanded, and as at December 31, 1966 it consisted of 2,900 miles of pipeline including loop-line. The investment in facilities is close to $600 million.

The present system can deliver up to 1.4 billion cubic feet of gas per day. About 95% of the gas enters the system at the Alberta-Saskatchewan border, and takes about three days to reach Montreal. To give you an idea in terms of equivalent B.T.U.'s this is equal to over 200,000 barrels of fuel oil or about one-fifth of the current Canadian production and consumption.

The power to transport the gas is provided by 111 compressor units totalling about 550,000 horsepower. To give you an idea:

An outside view of reciprocating station.

An inside view of a reciprocating station, high 30,000 H.P., $7.5 mm.

An Orenda industrial gas turbine.

A compact, fully automated jet station, 9-12,000 H.P., $1.5 mm.

These units, most of which burn natural gas, are grouped into 45 highly automated compressor stations. This makes for a bewildering

combination of power units and compressors, each with its own operating characteristics and restrictions.

3. Statement of the Problem

The problem has generally been outlined in the "Introduction", that is, there are too many variables in a large pipeline for the dispatcher to effectively control the line. Specifically at T.C.P.L. the aim of the process control computer is to save fuel by more efficient operation of compressor stations and to guide the dispatcher into better control of the line thus allowing more throughput at the same or better operating cost. With the fuel consumption being in the order of 100 million cubic feet per day, it would only take a one percent saving to make the system pay for itself. In addition, from five to twenty million cubic feet of additional throughput can be achieved on some days that the line has upset conditions, by giving the dispatcher better operating data quickly.

Since Trans-Canada Pipe Lines is designed such that all the gas that is moved can be sold, the above figures can be translated into direct additional revenue and are not merely paper savings.

4. Feasibility Study

In the early part of 1963 management approved the starting of a feasibility study for the installation of a process control computer. A task force was formed consisting of the Supervisor of Programming,

together with two part time representatives from Engineering and Operations departments. The task force reported to the Manager, Systems and Data Processing, and regular monthly reports were issued to management. Outside consultants were also introduced to work with the task force and advise primarily on selection of hardware.

The task force concentrated on:

I. Definition of the problem, including current practices.

II. Visits and discussions with other transmission companies.

III. Intensive discussions with equipment manufacturers.

IV. Selection of equipment.

Around November 1963 it was established that through increased automation in the field, the instantaneous transmission of such information, by teletype, into a process control computer was possible and economically feasible. In our case the economic justification was one of the most difficult assignments. Though theoretically no one could argue with the mathematics and the fact that all technical people agreed that the savings should easily equal the expenditure, it was hard to show it. In this business it is impossible to establish a controlled experiment whereby the line is run with and without the computer, under the same conditions. The alternative was as follows:

Part of the feasibility study was the creation of a program simulating the pipeline. It was a crude version of the eventual process control program. We then selected certain events in the past and the decision made

by the dispatcher. Then by running the program on an off-line computer, we develop the solution. This we found could then be compared with the actual facts and its effect on the operation translated into dollars and cents.

We had little difficulty in showing that even if the computer was not used for anything other than such occasional events, we would recover the investment. Any additional application would represent extra benefits.

In spring of 1964 management approved the program and a full time implementation group was established. The hardware was selected and the delivery set for September 1965.

The computer selected was the IBM 1710. We had been using a 1620 up to this point so changing to a 1710 on a time-shared basis eased our programming effort. The computer was a Model II, 60K disk system with card I/O and typewriter output to dispatch. In addition, the dispatcher could communicate with the 1710 by a manual entry device.

This 1710 was replaced by an 1800 after only 19 months, since the latter had about three times the core, five times the speed and only rented for three quarters of the 1710 system.

The conversion was simple since the 1800 programs were all in FORTRAN while the 1710 programs were written in SPS.

5. Description of Telemetering & Automation

All 45 compressor stations and three large meter stations are fully telemetered and most are remotely controlled from the central dispatch office in Toronto.

The telemetering works on an exception basis, i.e. no information comes in from a station unless an alarm or upset is present at that station. An alarm is a change in the status of a unit or the recycle valve. An upset is defined as a change of 10 p.s.i. in either the suction or discharge pressure. The master telemeter control at Toronto, continuously polls each station sequentially to see if there is an alarm or upset there and if not, passes on to the next station. If there is, an alarm or upset message consisting of all the variables for that station, comes in to the 1800.

An hourly log is automatically produced on a logger showing the suction and discharge pressure, number of units on-line, horsepower and flow for each compressor station.

The dispatcher will eventually have full unit stop/start control at all compressor stations. When a unit is turned on or off by the dispatcher, the station automatically adjusts itself to this new condition.

At present there are about 1,000 alarms and upsets coming in to the 1800 in a 24-hour period. It is obvious that this amount of data does not allow the dispatcher sufficient time to analyze each message. One of the functions of the process control computer is to filter these messages and print out only the significant changes.

6. Description of Computer Guide System

The IBM 1800 is composed of a 32K Model II memory, card reader/punch, three disk drives, two remote 1053 typewriters and a decade switch

manual entry box. In addition, a special interface/buffer has been designed by IBM that accepts the telemetered alarm and upset messages as they come in serially by bit, converts them from Baudot Code to internal code, stores them in a special buffer and creates an interrupt to the computer when a digit has been assembled.

The system is operated on a time-shared basis in the sense that all existing computer applications are run on the 1800 at the lowest priority level. These applications include engineering design, gas sales, sales forecasting, deliverability of gas wells, etc.

We are using TSX-3, Mod. 3 which takes about 11K of core. We have had very little trouble operating with this system except for minor troubles in earlier versions.

All the process control and most of the off-line programs are stored on the disk. In total, there are ten levels of priority within these programs, (see list in Appendix). The non-process or off-line programs have the lowest priority. Level 4 has been assigned to the manual entry box which allows the dispatcher to ask for any data or program from the computer. The timed interrupts from the real time clock every 35 seconds, is level 1, and the telemetered alarms or upsets, level 0. Hardware interrupt for the typewriters, digital inputs and disks fill the intervening levels.

The following is a detailed description of the various programs servicing the system.

A. .Alarm/Upset Program

This program handles the assembled alarm, upset and demand log messages which have been assembled into complete messages by an in-core routine called CAEPG, and operates at level 7. CAEPG itself operates at level 0 and calls in the Alarm/Upset program after it has assembled a complete message. An alarm is defined as a change in the status of a unit, recycle valve, generator lock-out or local/remote indicator. As soon as the message stack located in the computer memory contains an input message to be processed the Alarm/Upset program is called in after any appropriate lower level has been saved. This program then checks the incoming message to determine if it is an alarm or upset. If it is an upset it takes all the data that has come in on the upset message, i.e. pressures, temperature, flow, number of units on-line and calculates horsepower and if it is a turbine station, RPM, Head, ACFM and Surge condition. At this point the program checks these variables against operating restrictions and if any are violated, types a message to the dispatcher telling him the time, station number and the particular offended condition. If no violation has occurred, no advice is given to the dispatcher; in this way many of the needless upset messages are eliminated.

If the interrupt has been caused by an alarm, the above procedure is repeated, but in addition, the particular data that comes in for alarm, the status of each unit and position of recycle valve, are checked and the reason for the alarm is printed out in English.

B. Timed Interrupts

There are several programs that can be called in this level by the timed interrupt that occurs every 35 seconds. The interrupt is ignored until fifteen minutes to the hour at which time a program is called in to check if there are any stations that have not produced an alarm, demand or upset in the past three hours. If there are, these stations are printed out for the dispatcher to call up via a demand log. Although it is perfectly acceptable for a station not to alarm or upset in a three hour period, this acts as a check to make sure that the station's alarming equipment is working, and ensures that these stations will have their computer-stored data updated at least every three hours.

Every hour, an hourly log is printed from data stored internally. The suction pressure, discharge pressure, number of units, set point, flow, and B.H.P. are logged for each of the stations on the logging typewriter. After this has finished, another program is called in that computes horsepowers, flows, flowing temperature and pressure, and line pack for the entire line. This information is used during the following hour in the handling of alarms and upsets.

At eight o'clock in the morning a complete statistical report is produced showing the horsepower hours, operating and standby hours for each station, the number of alarms, upsets and false alarms and change in line pack for 24-hours.

C. Manual Entry Unit

The dispatcher can call for either data or a program from the computer or enter data into the computer via his 1077 manual entry device consisting of twelve rotary decade switches and an interrupt button. For example, if he wishes to know what the horsepower is at a station, he dials the station number and a two-digit code specifying horsepower, presses the interrupt button, and if higher level programs are not running, will receive the horsepower within three seconds on the typewriter.

If he wishes to change say a sales volume at a particular point in the data stored in the computer, he will dial a sales code, pipeline section number and the new sales volume, press the interrupt button and the data will be entered into the disk.

He can also call the simulation/optimization program at any time he wishes, by dialing the particular code for that program. He would probably do this at the occurrence of a major disturbance to the line such as a station or large horsepower unit suddenly dropping off-line. This program may take up to five minutes to run depending on the number of alarms and upsets which occur and also the complexity of the pipeline due to the non-availability of horsepower.

D. Simulation and Optimization Program

The heart of the process control system is the simulation and optimization program mentioned above. This program is called in by the

dispatcher via the manual entry device to scrutinize pipeline conditions and determine if there is any way to improve its operation.

The simulation program uses an iterative procedure that increments the flow into the line by 5 million cubic feet, using the existing flow as the starting point. With each increment in flow, the program proceeds down the line computing pressure drops, and determines if there is adequate horsepower available to handle the additional flow. The initial increment of 5 million cubic feet is doubled with each iteration. Eventually after several iterations, there comes a point at which the line is incapable of handling additional flow. The program then decrements the flow by one-half of the last increment and tries this amount down the line. If this proves successful, another half again is added and so on, until the difference is down to 2 million cubic feet. This process is quickly convergent, and the whole process takes about six minutes on a Model II 1800. When the final flow has been determined, a check is made to ascertain if any minimum RPM or Surge restriction has been violated. If so, corrections are made to the stations in question and a final pass determines if the flow has to be reduced slightly to compensate for these adjustments.

The final output to the dispatcher consists of a list of compressor station's settings to be used to achieve efficient running line conditions. The dispatcher has to use his judgement as to what order to change the horsepower, and whether to do so at all, if he knows of a particular line condition coming up that is unknown to the computer which would negate its instructions.

The dispatcher can call in the simulation/optimization program to solve a hypothetical problem such as: how should the line be set up to handle the shutdown of a particular station that is going to occur tomorrow due to maintenance? In this case, he can dial in the station that is going to be shutdown, plus any other pertinent information, call in the program and get his answer within five minutes.

7. Conclusions

Here is what we have achieved. For a relatively small incremental cost we have successfully entered the process control field. Without any reservation we are satisfied that we have met the original target. We know that we have saved in fuel requirements. The computer has provided guidance to the dispatcher which in turn resulted in higher throughput. It would be rather difficult to pinpoint exactly how much of this extra throughput is due directly to the process control system due to many inter-related factors. Whatever the inter-relationship we are satisfied that it is higher. In addition we have derived the following extra benefits:

-- We have a hardware system capable of accommodating large and sophisticated programs to be used by other departments in the company. Our Engineering department uses it several hours a day for pipeline design.

-- We have a competent and active group trained in the disciplines required for a successful development of advanced scientific application.

-- Protection of facilities against hazardous operation by scanning and informing the dispatcher when a station does not respond.

-- We learned new facts about our system. Installing a computer system forces one to think more deeply about your operations.

-- Faster preparation of reports on daily operating conditions and immediate corrective action.

-- Management can reach decisions on quantitative facts presented in real time, instead of after the fact.

-- Plan a better maintenance shutdown schedule.

Closed loop which is theoretically possible now, is a possibility. For the time being we are not investigating this aspect for a number of reasons, the most important of which is safety and the fact that reaction time is not critical in our operations and hence closed loop would be difficult to justify financially.

Thank you.

APPENDIX

Interrupt Levels

Levels	Bit Position	Device	Type	
0	0	Tel. Interface (CAEPG)	User	Recognize interrupts - Call Level 7 for all reading and analysis.
1	0	Time Interrupt	TSX	
2	0	2310-1	TSX	
2	1	2310-2	TSX	
2	2	2310-3	TSX	
3	0	TYP-1	TSX	
3	1	TYP-2	TSX	
3	2	TYP-3	TSX	
3	3	1442-1	TSX	
3	4	1442-2	TSX	
3	5	DAO	TSX	
3	6	DI	TSX	
4	0	1077	User	Recognize interrupts - Call Level 9 for reading and analysis.
4	1	Console	TSX	
7	0	LEVLV	User	Called by Level 0.
8	0	Console	User	Call QUEUE Program(CONSL)
9	0	Manual Entry	User	Called by Level 4/0 - for reading dials and all analysis.
10	0	Out of Core Interrupt	User	Type message.
11	0	Keyboard	User	Call Intex.

W. I. 9.

SYSTEMS AND PROGRAMMING PROJECT MANAGEMENT

The allocation of limited systems and programming resources to the highest payoff areas is becoming more important as data processing installation costs continue to rise. Long and short-range plans properly approved by top management and formalized systems project management are vital tools to assure that systems capability is focused on the major areas of the enterprise and adequately controlled to attain the stated objectives. To work effectively in this environment, additional demands are placed onto the systems group to develop systems plans in terms easily understood by top management and onto data processing management to bring about fulfillment of the approved plans on time and as economically as possible.

Ralph B. Sackman, Jr.

RECEIVED
OCT 25 1967
C.S.C. COMPUTER CENTER

ORGANIZING FOR MAXIMUM COMPUTER RESULTS

By Ralph B. Sackman, Jr.

A typical goal of many organizations today is to use a computer to provide information for more effective management of the enterprise. In the eyes of management the computer has passed through the trial stage, which has tended to restrict its usage to record keeping activities in the past, and third generation equipment has been developed with far more capacity and flexibility than was previously available. But more than management approval and computer equipment capacity is needed to achieve maximum computer results for the entire organization.

The first step is to eliminate "fire fighting" that can absorb most of the available system's and programming resources. This can be accomplished by formalizing responsibilities within the data processing area and by establishing a standard method for computer users to request program changes and other services. The timely and accurate reporting that results from a well organized data processing function builds confidence in the ability of systems and programming people to assist in solving problems in other parts of the organization.

The next step toward achieving maximum computer results is to assure that new systems are relevant to the interests of management. An approach toward this end would include a top level management steering committee to direct the overall data processing program. Effective communication between the steering committee and the systems group, including development of a long range plan, assures mutual understanding of management interest and expected system performance.

588

An example follows of the specific actions that could be taken to formalize responsibilities in an existing computer operation and to expand computer usage.

EXISTING OPERATION

A large number of program maintenance tasks could remain after the conversion of a large application. An inventory should be taken of the maintenance items and a short term schedule published citing the task description, the requester and the approximate time to complete. The schedule would enable the supervising programmer to ascertain priorities and assign the work on a systematic basis to his staff. A request for data processing services form should be put into effect simultaneously with the short term schedule for use by interested parties in requesting additional maintenance programming work. Verbal requests for programming work should be discontinued.

Other procedures should be put into effect to make formal the communication between the programming unit and computer operations. Verbal requests for services from or instructions to computer operations personnel from the programming unit should be discontinued by means of two rather simple forms. One is a transmittal letter for requesting computer room services, including the keypunching, assembling and testing of programs and other related tasks. The second transmittal letter is to send new or revised object programs to the computer operations supervisor. He would utilize this document to assure adequate program library maintenance and to inform the computer operators of changes to programs. Formalizing the communication in this manner would reduce the misunderstandings between the programming unit and computer operations.

Computer operations could meet the servicing needs of the programmers while maintaining production schedules by establishing priorities for compilations and testing. Only vital production jobs should carry a priority higher than program compilations and testing. Programmers should receive the results within a few hours and carry on with their work without lost time. Through this procedure the computer users should receive good service on their requests for modifications and difficulties should not be experienced in meeting existing production schedules. The computer operators should run all program tests and assemblies utilizing written instructions submitted by the programmers. Keypunching of programs should also be handled under a priority procedure where a priority one would require immediate attention of the keypunch operator and a priority two would require return of the punched cards and coding sheets within forty-eight hours of receipt.

The librarian functions within the computer operations unit should be assigned to a single individual. That person would maintain the program object deck library utilizing the object deck transmittal forms from the programming unit. Programmers or other individuals should not be permitted to insert or delete object programs from the library. The librarian should update the computer setup and run instructions to notify the computer operators of changes in operating procedures brought about by the program revisions. The single responsibility concept should also be utilized in maintaining the magnetic tape library. Only the librarian should be permitted to select tapes from and return tapes to the library.

A reporting system for computer utilization and scheduling should be developed. The reports should include program efficiency statistics which would show the number of program tests and compilations provided to each programmer. The results would indicate the problem of excessive computer runs caused by inaccurate coding. Utilizing this information, the supervising programmer could discuss performance in specific terms with each of his programmers. Other reports should show computer time spent on production runs by application area. Computer time trends should be charted to foresee any need for additional equipment or a modified configuration. Machine loading would be leveled by means of a monthly forecast of computer time which would portray days that are overloaded. The computer operations supervisor would utilize this report to rearrange schedules, where possible.

Programming standards should be developed to assist in maintaining the quality of work ~~in~~ and a high level production in the programming unit. Standard methods and documentation permit the assignment of program maintenance to junior programmers, thus releasing for other new work the programmers who originally wrote the programs. Approval for departure from the standard methods should be obtained from the supervising programmer. The standards should be reviewed regularly to insure that they are up to date, realistic and useful. Preparation of standard program documentation prevents misunderstandings by maintenance programmers and by computer operators responsible for the proper usage of the program. Thus, the program maintenance and program running tasks should become routine functions that do not require substantial time of systems and programming personnel assigned to new areas.

NEW AREAS

Attention should next be focused on the way to organize the systems effort for work on new long term projects. Definition and selection of future projects often is not as obvious as the original application that justified the computer. The justification of certain projects would be based largely upon intangible benefits rather than reduced clerical workload, reduced accounts receivable float or improved customer service. To provide continuing assurance that the systems and programming group is working on the right problem, a management steering committee should be appointed by the president. The committee's broad function should be to evaluate planned systems projects and establish priorities.

Soon after appointment of the steering committee, the data processing group should prepare a tentative long range schedule showing the major areas of computer usage. The long range schedule should be submitted to the steering committee with the understanding that systems planning reports would be furnished to the committee prior to final approval on any new project. The systems planning reports would show the changes required to improve the flexibility of existing systems, the recommended sequence of projects and the approximate costs. The reports would be written in business terminology and make little, if any, reference to the computer itself. The steering committee would review the reports to assure that the proposed results are relevant to needed management information and to satisfy itself that the results are worth the conversion costs. Upon approval the steering committee would request that appropriate persons be assigned to a project team to work with the systems analyst to develop a detailed systems design.

With multiple systems projects in progress, effective project management is essential. The data processing group should operate under the project leader concept where one person is responsible for a project from the systems planning phase through to the time that all programs are operating successfully in production. The project leader would be responsible for estimating the completion date, scheduling all required activities, and recommending the precise boundaries between his project and others. Each project leader would report monthly as to his progress against plan and would state problems he has encountered that are beyond his control. Programmers would be assigned to the project leaders based upon demonstrated need. Project backlog should be documented at all times.

The project leaders would work in partnership with designated operating personnel toward realistic and economical systems design in new areas. The operating personnel would approve the systems design as it affects their operations and the systems project leader would develop the detailed plan to meet the desired criteria. The management steering committee should be kept informed of progress through minutes of meetings and should review the objectives and their values prior to implementation.

A significant advantage in reducing "fire fighting" and establishing a reporting relationship with top management is the salutary effect upon the esprit de corps of the systems and programming group. Personnel in the group know what projects are ahead and realize that consideration of the backlog and work in progress is considered thoroughly prior to a decision involving a change in direction. Each of them would be assured of participating in a worthwhile project. In addition there is little chance that personnel would be pulled off a long term project to correct a day to day reporting or control problem.

In summary, formalizing the present operation would make intelligent assignment of personnel possible. Appointment of a management steering committee and the systems planning concept would assure that new long term projects attack the largest pay off areas first. There would be a mutual understanding between management and the data processing group as to where the data processing effort stands today and where it is going. Ultimately, the customers and stockholders would benefit because effective computer usage would enable management and employees to do a better job.

December 20, 1967

To: Members of 1620 Project Section COMMON Users

From: Tony A. Ross and Richard D. Ross, Co-chairman 1620 Project Section

As I promised in San Francisco, enclosed you will find a copy of the names and addresses of all 1620 USERS registered at the San Francisco meeting along with the comments that each of you put on the sheet that Richard passed out at the first meeting. Please read these comments carefully and if you can help any of your brother-members in any way, please contact them on an individual basis.

I noticed that many of you were interested in some items and upon compiling the list I found that someone else had been doing work in this particular area. So maybe with a lot of correspondence back and forth, we can be of help to each other.

I just finished a letter to Dr. Jardine suggesting that he send a copy of the Kingston compiler directly to me and I, in turn, will send each of you a copy if you are interested. As soon as he answers my letter and sends a copy of the processor, I will then write to each of you, and you may request a copy from me. Do not let this deter you from writing directly to Dr. Jardine if you so desire.

I am also in the process of completing the copying of the 1620 Newsletter and as soon as these are completed, I will mail these to each of you that specifically requested the remainder of the Newsletter.

Keep in mind our next COMMON meeting to be held in Chicago April 8-10 at the Pick-Congress Hotel. I will be glad to accept any abstracts or papers you would like to present now. I think the meeting in San Francisco was very informative and useful to each 1620 User that attended. Let us all work together to make the meeting in Chicago even more useful and this can only be done with the cooperation of everyone.

Since it was suggested that some papers be given on how to write or modify compilers, I am especially interested in papers of this nature. Although, any useful paper will be accepted. If any of you are doing work in the CAI area, I am sure that this would be of interest to the Users that attend the Chicago meeting.

I want to use this letter to personally thank each of you for attending the meeting and making it the success that it was. I especially would like to thank the ones of you who presented the very useful papers that were given.

Again, let me remind you to be thinking about papers and send any information that you have that might help for the Chicago meeting.

I hope that each of you had a Merry Christmas and a very Happy New Year.

Yours truly,
Tony A Ross, Richard D Ross

Tony A. Ross and Richard D. Ross
Co-chairman 1620 Project Section

P.S. Don't tell me about "sunny" California anymore. I almost froze to death the whole time I was there.

A. Moore, W. T.
Presently awaiting delivery of Monitor I system
1620 Mod-1, 40K, 2-tapes
1443 Printer and Disk

B. Streeter, T. D.
Model 1, 40K

C. Sloboda, J.
1620 Mod-1, 20K
Interested in obtaining help on modifying processors

D. Brummer, Bro. Elmer
1620 Mod-1, 20K
Interested in COBOL compiler for 1620

E. Schrader, H. C. Mrs.
1620 Mod-1, 20K
Have made modification to PDQ compiler
Have precompiler for PDQ FORTRAN

F. Sward, R. W.
1620 Mod-1, 20K, 2-1311, 1443, Teleprocessing working on
1. Computer assisted instruction
2. Computer based councelling
3. Student scheduling
4. Student transcript processing

Interested in DATA SPS for 20K - 1620

G. LaRue, Richard

1620 Model I, 40K, 1 Disk, Printer
1. Interested in obtaining:
   a. Kingston FORTRAN
   b. DATA SPS

2. Can Contribute
   a. Test analysis program using 1230 optical scanner
   b. Tabulation and cross tabulation program for Questionnaire and Survey Analysis.
   c. Medical (or other graduate college) school admissions evaluation system
   d. Student Scheduling Programs
   e. Alumni Processing
   f. SPS SORT routines available for FORTRAN

H. Fricke, E. C.
1620 Model 1, 20K
1. Interested in
   a. Student use of 1620
   b. Student scheduling on 1620
   c. Batch processing
   d. Precompiler for PDQ

I. Goldstein, J. H.
1620 Model II, Disk Drive, 40K, Paper Tape Input
1. Need
   a. Information on Data SPS for 40K

J. Lahners, E. L.
1620, Model I, 20K, 1621, 1622, 1624, 1626, 1627
1. Need PDQ for plotter
2. Processor Available
   a. FORTRAN
   b. PDQ FORTRAN
   c. FORCOM
   d. PDQ-FORCOM
   e. NCE
3. Programs Available
   a. Payroll
   b. Engineering manpower, cost analysis
   c. Pharmacy drug cost accounting (patient, service)
   d. Statistical programs
   e. Neutron Activation Analysis (Simplex Method)
   f. Registration (patient Record Information Retrieval)

K. Jones, C. E.
1620 Model I, 40K, 1443 Model 2 (on order), 1 Disk
1. Need information on Kingston FORTRAN for Disk with and without printer

L. Jones, C. E.
Model I, 40K, with p Disk and Printer
1. Interested in re-read for FORTRAN II and save paper for Monitor I

M. Emerson, J. T.
Model II, 20K, with 1 Disk and no printer
1. Interested in Data SPS, 20K, Version and Compiler modification (10 cards)
2. Can contribute:
   a. Program that reads Porta punched cards for FORTRAN II D program
   b. Program to punch a contour map in 9 densities
   c. Program interrupt - runs large program until start button is depressed, then the large program is stored on the disk and the smaller program is processed. Upon completion of the smaller program, the larger program is then brought off the disk and continues to run.

4. For information on items 1-3 contact
H. Miller
Fresno State computer center

1620 PROJECT

Mrs. L.B. Austin
General Motors Inst.
Flint, Michigan

Paul A. Bickford
DePauw University
Greencastle, Indiana

Bro. E. Brummer
St. Edwards University
Austin, Texas

J.C. Cakin *

J.C. Caslin
Aerospace Research
Dayton, Ohio

J.T. Emerson
Fresno State College
Fresno, California

R.L. Ferral
U.S.W.B. Sacramento RFC
Sacramento, Calif.

D.W. Forsyth
Continental Can Co.
Chicago, Ill.

E.C. Fricke
Linfield College
McMinnville, Oregon

J. H. Goldstein
Emory University
Atlanta, Ga.

L.L. Hoffman
Princeton University
Princeton, N.J.

S. Iwasaki
Palo Alto UNFD School
Palo Alto, Calif.

C.E. Jones
Tennessee A & I State Univ.
Nashville, Tenn.

W. G. Lane
Chico State College
Chico, Calif.

E. L. Lahners
VA Hospital
Omaha, Nebraska

R. H. LaRue
Univ. of South Dakota
Vermillion, S.D.

E.S. Lee
Univ. of Toronto
Toronto, Canada

J.A. N. Lee
Univ. of Mass.
Amherst, Mass.

S. MacGregor *

E. N. Michalowski *

W. T. Moore *

John Powell *

Mrs. H. L. Schrader
CSCH-Hayward
Hayward, Calif.

J. Slobda
Flint Jr. College
Flint, Mich.

R. F. Steinhart
IBM
White Plains, N.Y.

T. D. Streeter
Rock Island Arsenal
Rock Island, Ill.

R. W. Sward
Palo Alto UNFD School
Palo Alto, Calif.

Dr. Jose A. Treuino *

T. Watson
Stanford Electric Labs
Stanford, Calif.

J.E. Wilson
Rock Island Arsenal
Rock Island, Ill.

P.C. Wood
Lowell Tech. Inst.
Lowell, Mass.

O.G. Wright
Pioneer Natural Gas
Amarillo, Texas

Hugh Kerr
Tenn. Tech. University
Cookeville, Tenn.

* Address not availabe. As soon as these addresses are available, they will be sent to you.

W. 2.5

1620 Project

A Batch Processing FORTRAN System for a Minimal Configuration 1620

by

Gaylord Glenn Henry

California State College at Hayward

Hayward, California

1145 Carlsbad Drive #1

San Jose, California

(408) 266-4863

Session M.3.3

Text: 9 pp.

6 Figures

Acknowledgement

My work on this project was done under a Western Data Processing Center Research Assistant Grant. Two other persons also contributed to the development of the project. Mr. Jerry Rose, Assistant Director of the California State College Computer Center, was instrumental in the implementation, and Mrs. L. H. Schrader, Director of the Computer Center, aided in planning and guiding the project.

INTRODUCTION

This paper describes a batch processing FORTRAN system, written at California State College at Hayward, for a 20K, card system IBM 1620 computer. The system is designed to increase throughput and introduce larger system concepts in a student oriented installation.

The primary characteristics of the computing environment at Cal-State are

1. A large number of fairly short FORTRAN programs which have a fast turnaround requirement during periods of heavy computer usage. (These are computer class assignments.)

2. A lesser number of FORTRAN programs for classes such as chemistry and physics. These can be run during off-hours in a "closed shop" environment.

The most important factors contributing to efficient utilization of computer time and student time during the prime hours of class usage are

1. Source Language Error Detection

   Most 1620 card system compilers have limited error checking and will not detect source language errors. These execution errors are costly in terms of student and assistant time. Most students require assistance in diagnosing errors, and machine debugging in such cases is both difficult and time consuming.

2. Decreased Deck Handling and Loading

   In an environment where there are many short programs to be compiled and executed, the time spent in loading the compiler and subroutines can easily become the most significant machine time factor. Efficient use of batch processing techniques can greatly increase the total throughput and also provide fast turnaround for any particular job.

Another important consideration in a system oriented toward programming instruction is the compatibility of the system with larger computer systems. It is desirable that the language used be consistent with FORTRAN's found on larger systems, and that it contain as many basic FORTRAN features as possible.

The C.S.C.H. FORTRAN system, designed to satisfy the requirements of fast throughput and larger system compatibility, is based on the PDQ FORTRAN system. The PDQ compiler was chosen because it has the shortest compile time and produces smaller and faster executing object programs than any 20K card system compiler available at the start of the project. In addition, the PDQ FORTRAN language appears to be the most advanced of the available 1620 FORTRANS with features such as extended FORMAT capabilities and COMMON and PROCEDURE statements.

The PDQ system has two major drawbacks, however. Source language error detection is minimal, and is unsatisfactory for use in a teaching system. In addition, the subroutines must be separately loaded with each object deck. This constant loading becomes quite expensive when a large number of short programs are to be run.

The C.S.C.H. FORTRAN maintains the speed and language attributes of PDQ while reducing these disadvantages. The three principal components in the system are:

1. FORTRAN Precompiler:

   Since the level of error detection required could not be provided in the compiler, a separate precompiler was written which provides a high level of error detection. The total time -- both student and machine -- required to produce a working program is greatly reduced in this manner, since the previous compile-execute-debug phases necessary to find one error is replaced with a precompiler phase which detects multiple errors.

2. FORTRAN Compiler:

   Some additional features have been added to the PDQ language and some modification and deletion of existing statements made. Statements are included that provide communication with the batch processing execution monitor.

3. Subroutines and Batch Processing Monitor:

   A monitor has been added to the subroutine library which allows batch execution with only one loading of the subroutines for any number of stacked execution jobs. Job card control, FORTRAN calls to the monitor, output line counts, program timing, program linking facilities, and continuous no halt execution of programs are features provided by the monitor.

Both the compiler and precompiler operate in a batch processing mode under control of JOB cards. This means that any number of programs can be processed to execution in three phases with only one loading of the precompiler, compiler, and subroutines, and no halts aside from the loading process. The procedure is indicated in figures 1, 2, and 3.

The entire system is card oriented, and no typewriter input or listing of source statements is allowed. It is also possible to suppress typewriter output at execution (by using a monitor option) in order to achieve faster throughput.

COMPILER PHASE

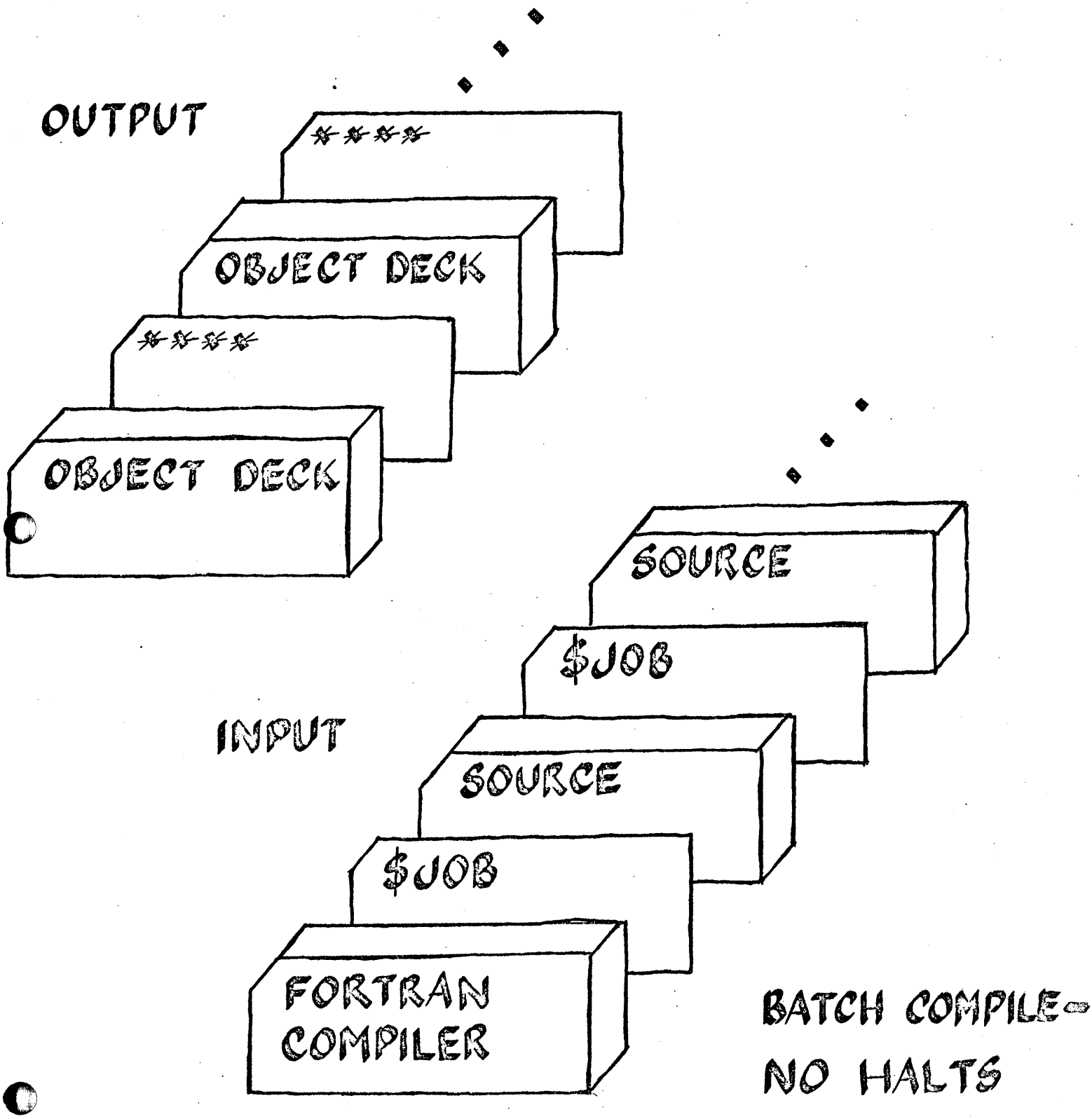
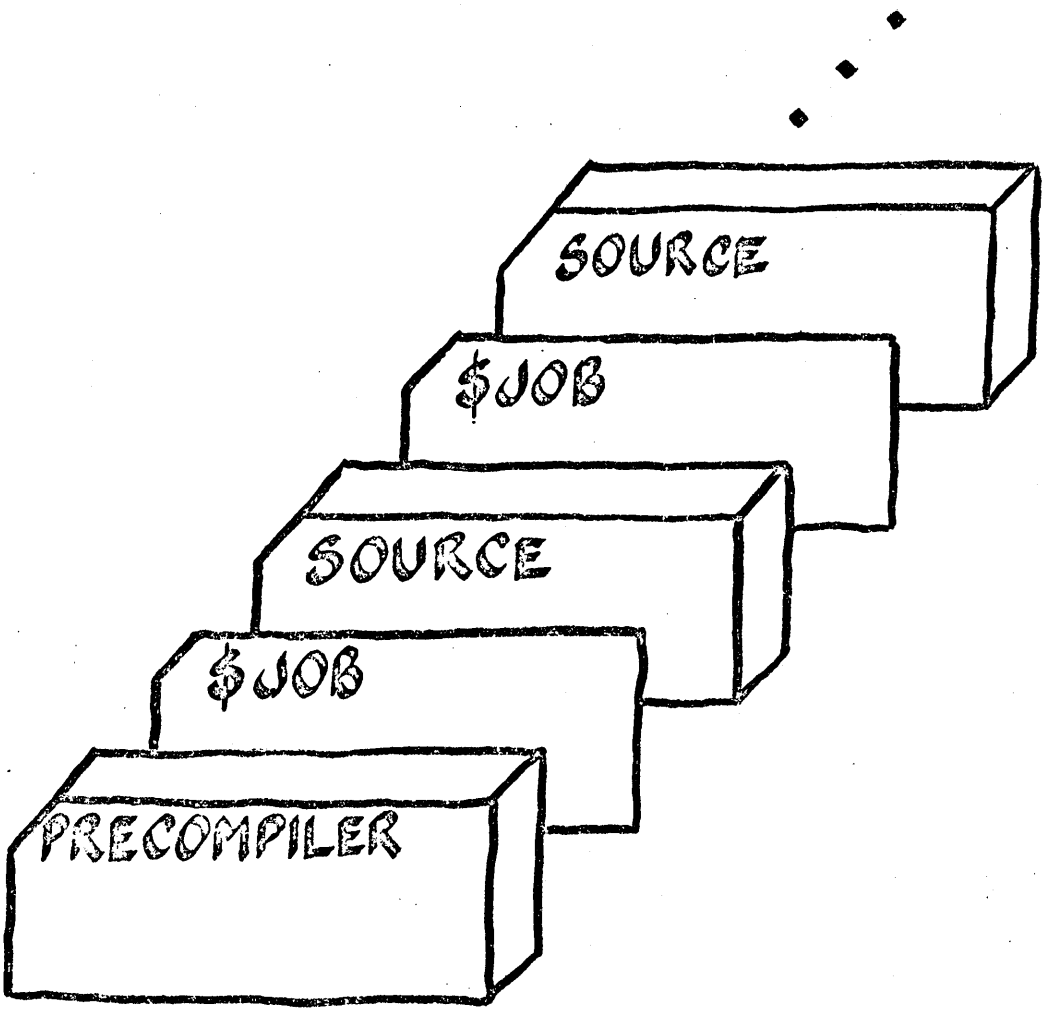


Fig. 2

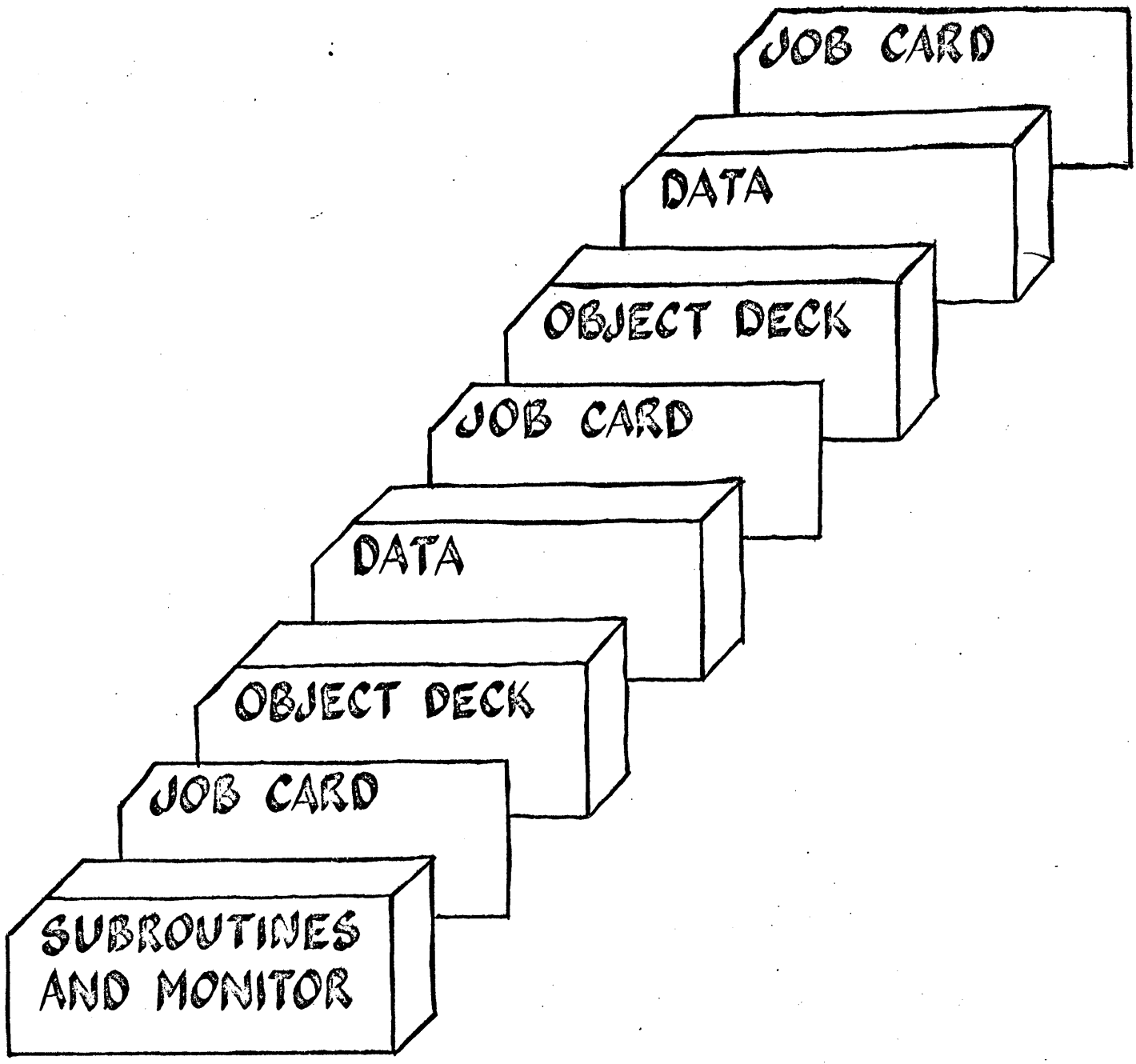
PRECOMPILER PHASE



BATCH PRECOMPILE –
NO HALTS

Fig. 1

EXECUTION PHASE



BATCH EXECUTION —

NO HALTS EXCEPT CHECKSTOPS

Fig. 3

Execution Subroutine Configuration

One of the undesirable features of the PDQ system, as well as other 1620 FORTRAN systems, is the requirement that the subroutines be reloaded for each object deck. This procedure is necessary so that the functional subroutines (SIN, COS, etc.) used by the program can be relocated behind the object program. The execution configuration is illustrated in Figure 4. In the environment discussed, however, the optimization of core storage provided by including only the subroutines required for a particular program is not as important a factor as the time required to load the subroutines with each program.

This reloading of subroutines can be eliminated at the cost of storage space by keeping all of the subroutines in core at the same time. Programs can then be loaded behind the subroutines into a fixed location. This configuration is shown in Figure 5. It has been found that the "variable core" remaining for object programs in this system is adequate for practically all programs in the student environment. This "student version" also allows the batch execution of object programs under control of a monitor subroutine.

There are three execution subroutine configurations possible. These are specified to the compiler by a field on the JOB card. The compiler origins the object programs to load into the proper location. The configurations are

1. Basic Student Version:

   The following functional subroutines are included: SIN, COS, SQRT, LOG, EXP, ABS. Object programs begin at approximately 09600.

2. Expanded Student Version:

   This consists of the basic set plus ATAN and DRH routines. Object programs begin at approximately 10500.

3. Original Relocatable Version:

   Object programs begin at approximately 07400.

Option 3 is rarely required since most programs can be batch executed in the student version. The same subroutine deck can be used for either of the student versions.

Execution Monitor

The monitor provides batch execution capabilities by loading the next program stacked in the card reader when certain end-of-program conditions occur. The start of programs in the input stream is defined by a JOB card. The automatic loading and executing of the next stacked program is initiated by any one of the following:

EXECUTION CONFIGURATION

ORIGINAL VERSION

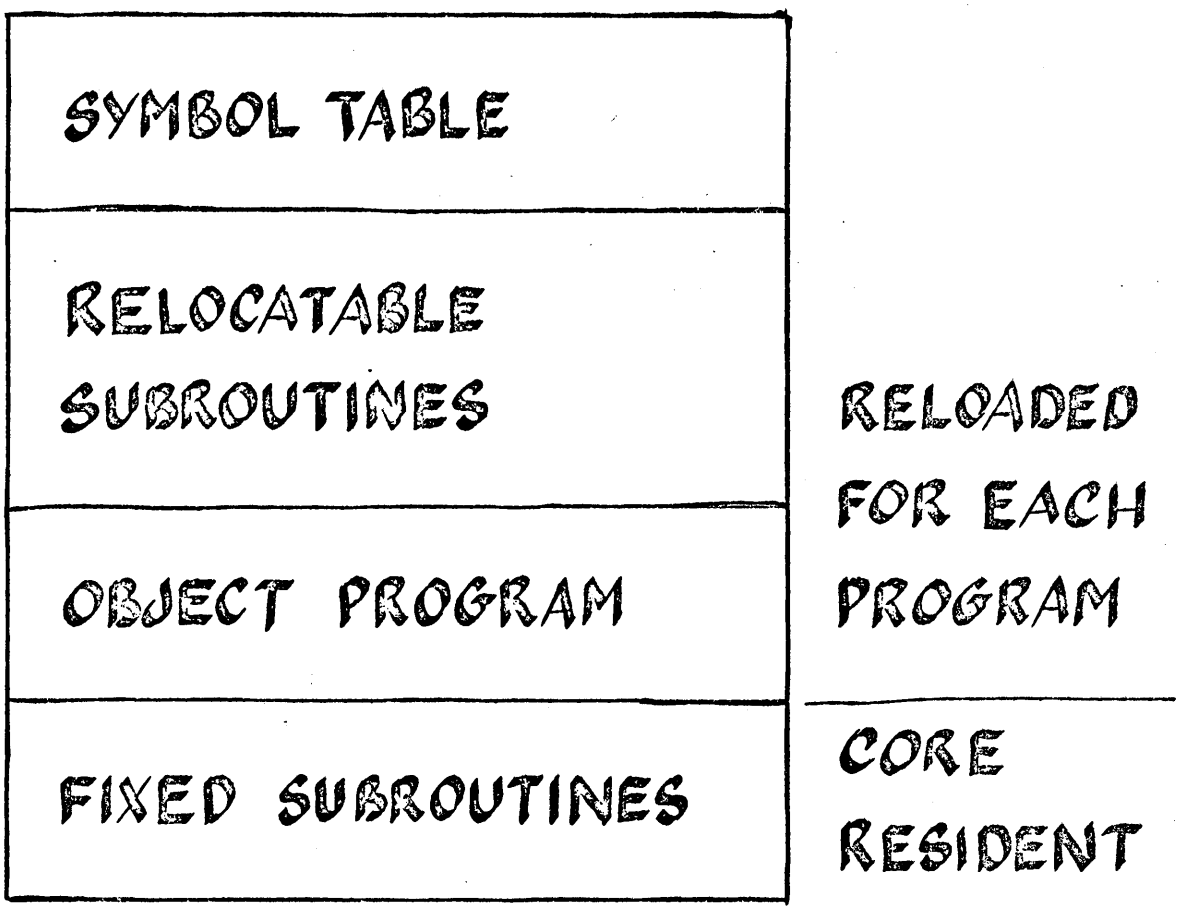


Fig. 4

EXECUTION CONFIGURATION

STUDENT VERSION

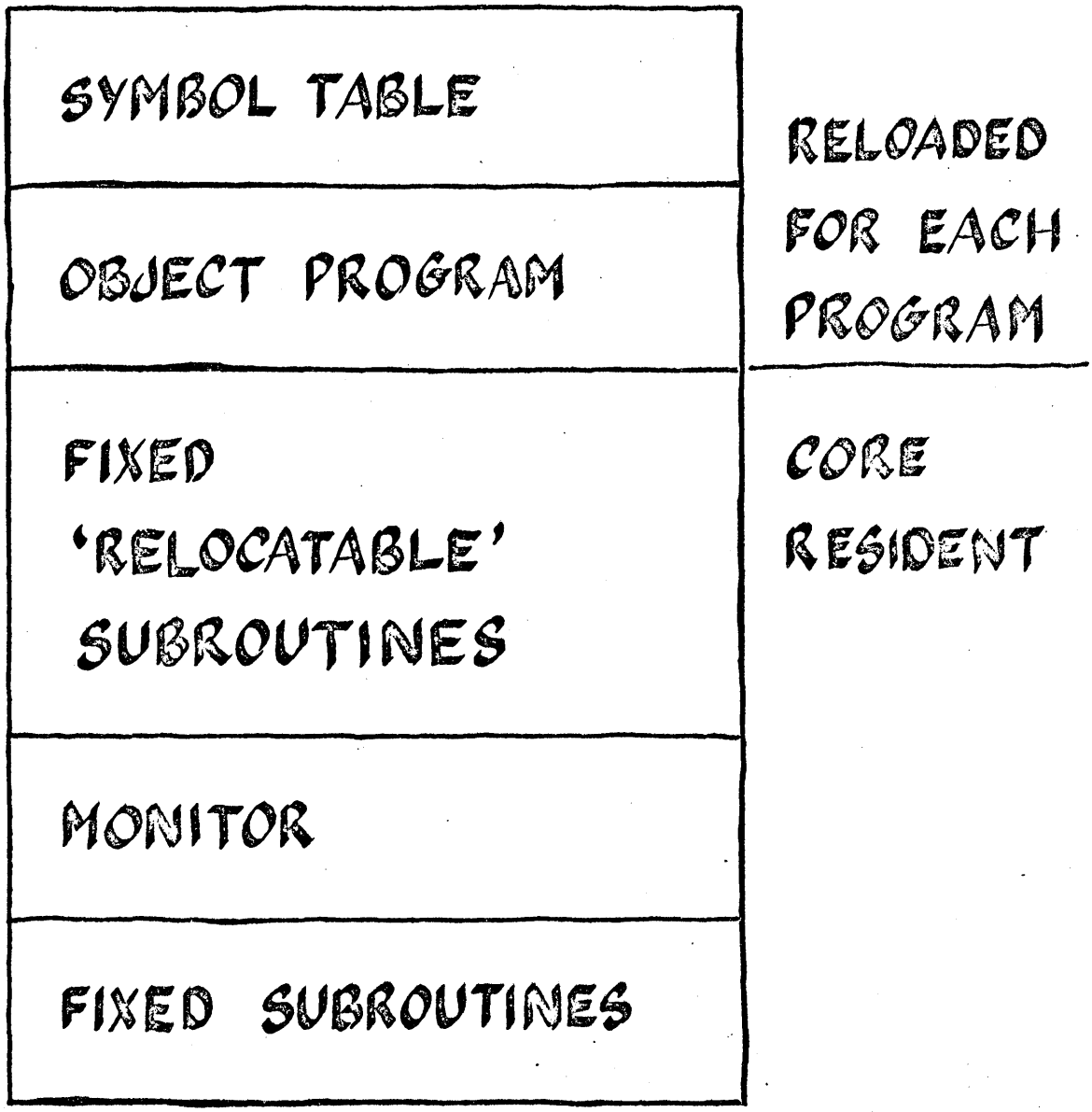


Fig. 5

1. Execution of a CALL LINK or CALL EXIT statement:

   These statements functionally terminate a FORTRAN program. The primary difference between the statements is conceptual. CALL EXIT terminates a job and CALL LINK is used together with COMMON to link overlay phases of a program.

   The overlay of a program can be profitably used to conserve core in large programs. For example, all program variables can be put in COMMON and initialized in a phase which is then overlaid by the main program. Figure 6 illustrates the source deck arrangement for this example.

2. Reading of a JOB card:

   An attempted READ of a JOB card by the FORTRAN program will be treated as an end-of-file condition for the input to the program. This will cause the monitor to branch to the specified statement if an ON ENDFILE statement is used or to terminate the job and cause loading of the next program. The ON ENDFILE statement is described later.

3. Execution of an END statement:

   For those who forget a CALL EXIT.

4. An Input Data Error

   An input error will be considered a job terminating error unless an ON DATA ERROR statement is used. This statement is further described under the FORTRAN compiler.

5. A Console 'Interrupt':

   Pressing INSTANT STOP, RESET, INSERT, RELEASE, and START at any time during execution will cause the current job to be aborted and the next program found and loaded.

6. Line Count or Times Overflow:

   These will be described below.

A separate termination message is typed by the monitor for each of these end-of-job conditions. The JOB card is also typed for each job.

EXAMPLE OF CALL LINK

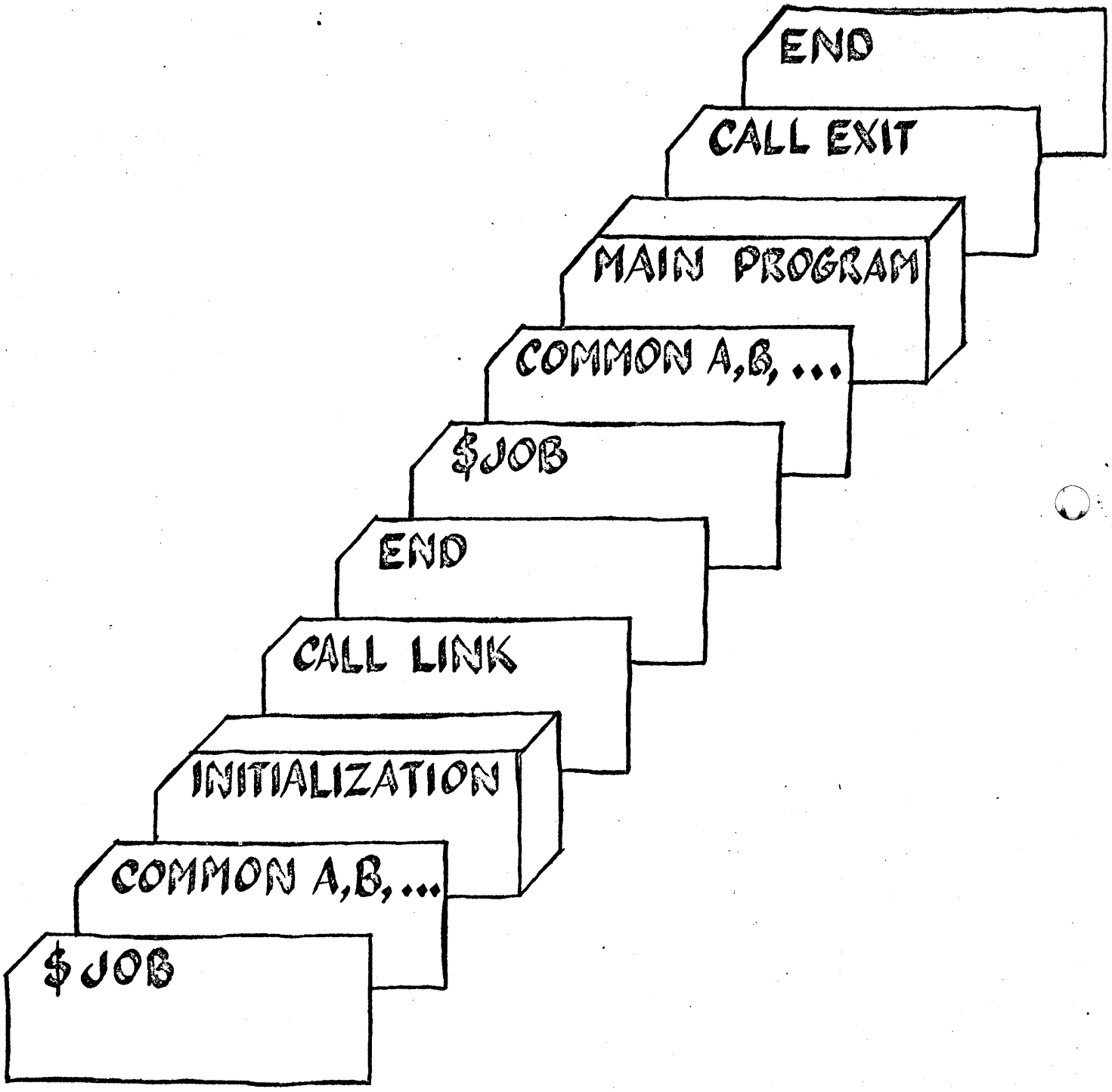


Fig. 6

The PDQ system has two major drawbacks, however. Source language error detection is minimal, and is unsatisfactory for use in a teaching system. In addition, the subroutines must be separately loaded with each object deck. This constant loading becomes quite expensive when a large number of short programs are to be run.

The C.S.C.H. FORTRAN maintains the speed and language attributes of PDQ while reducing these disadvantages. The three principal components of the system are

1. FORTRAN Precompiler

   Since the level of error detection required could not be provided in the compiler, a separate precompiler was written which provides a high level of error detection. The total time -- both student and machine -- required to produce a working program is greatly reduced in this manner, since the previous compile-execute-debug phase necessary to find one error is replaced with a precompiler phase which detects multiple errors.

2. FORTRAN Compiler

   Some additional features have been added to the PDQ language and some modification and deletion of existing statements made. Statements have been included that provide communication with the batch processing execution monitor.

3. Subroutines and Batch Processing Monitor

   A monitor has been added to the subroutine library which allows batch execution with only one loading of the subroutines for any number of stacked execution jobs. Job card control, FORTRAN calls to the monitor, output line counts, program timing, program linking facilities, and continuous no-halt execution of programs are features provided by the monitor.

Both the compiler and precompiler operate in a batch processing mode under control of JOB cards. This means that any number of programs can be processed to execution in three phases with only one loading of the precompiler, compiler, and subroutines. The procedure is indicated in Figures 1, 2, and 3.

The entire system is card oriented, and no typewriter input or listing of source statements is allowed. It is also possible to suppress typewriter output during execution (by using a monitor option) in order to achieve faster throughput.

FORTRAN Compiler

The compiler is essentially the PDQ FORTRAN compiler with the following modifications designed to increase language and batch compiling capabilities:

1. The compiler has a batch compiling option which suppresses the halt between jobs. A card with eighty asterisks is punched following the trailer cards for an object deck. This permits easy separation of object decks when batch compiling with no stops.

2. A JOB control card must precede each source deck. The JOB control card will be recognized by the compiler and typed to provide a record of jobs. The JOB card also contains compile options that specify execution subroutine configuration.

3. All source input to the compiler is from cards. Also, no typed listing of the source deck or symbol table is allowed, although the symbol table can be punched alphamerically.

4. Subroutines cannot be compiled into the object deck.

5. Almost all error detection has been eliminated. A program with no precompiler errors will compile correctly, however.

6. The following statements have been added to the PDQ language:

   a. CALL nnnnn

      where nnnnn is a five digit memory address. This generates a BTM instruction to the address specified. The operand is the return address to the calling program. This statement is used for communication with assembly language subroutines.

   b. IF (SENSE LIGHT n) S_1, S_2
      SENSE LIGHT n ON
      SENSE LIGHT n OFF

      where n is a single digit number and S_1 and S_2 are statement numbers. The IF statement is similar to the IF SENSE SWITCH statement except that it tests an internal logical indicator. The test does not turn off the "senselight". The other two statements are used to set the ten indicators.

      Since the senselights are internal indicators and require no core storage, their use will require less object code and symbol table space than the use of arithmetic variables as indicators.

c. CHECKPOINT nnnn

This statement is the same as the STOP statement except that the program does not halt after typing the specified number. This has been found to be a useful logic flow debugging aid without the slow execution typeout disadvantages of TRACE.

d. ON ENDFILE GO TO S_1

where S_1 is a statement number. This statement specifies the number of the statement to be executed when the last data card has been read. This condition is defined by the last card indicator or the reading of a JOB card. The statement can be placed anywhere in the program but must be executed before the end of file condition occurs. If this statement is missing, the reading of a JOB card will terminate the program and cause execution of the following stacked program.

e. ON DATA ERROR GO TO S_1

This is similar to the ON ENDFILE statement and specifies a program entry point if an input conversion error is detected. An input error will terminate the job if this statement does not precede the occurance of the error.

f. CALL EXIT and CALL LINK

These are described under the batch processing monitor.

The PAUSE and CONTROL statements have been deleted from the language and the syntax of Procedure Statements has been changed.

FORTRAN Precompiler

The FORTRAN precompiler is a one pass error checking program which executes at approximately the same speed as the compiler. JOB card recognition and batch processing capabilities are included. Some of the major features of the precompiler are

1. As far as is known, all non-logical checkstop errors are detected. Execution errors such as a calculated subscript going out of bounds are, of course, undetected.

2. A large number of separate diagnostics (approximately eighty) are provided. Errors are identified by a two digit number which is typed out along with the statement containing the error.

In addition to controlling the initiation and termination of programs in a batch execution mode, the monitor also provides optional monitor functions during the execution of a program. The features provided are

1. Output Line Count Monitoring:

   The number of output lines allowed is specified on the JOB card. If this limit is exceeded, the job is aborted.

2. Suppression of Typed Output:

   A TYPE or PRINT statement will execute as a PUNCH statement. This results in considerable throughput increase in a batch processing environment. JOB cards are punched when encountered in order to easily separate output when listing the cards from a series of jobs. Because of the construction of the JOB card, a 407 board may be easily wired to cause ejection to a new page when a JOB card is encountered during listing.

3. Suppression of STOP Statement:

   In order to maintain the no-stop, batch processing concept, the halt of a STOP statement may be suppressed.

4. Execution Time Monitoring:

   Since the FORTRAN compiler produces very little in-line code and practically all operations are performed by a few basic execution subroutines, it is possible to provide software execution timing capabilities. This is done by placing code in fifteen basic subroutines which will increment a software timer by an amount proportional to the length of time required to execute that subroutine. This will increase the execution time of such a subroutine by 240 μs if the time monitor is disabled and by 1040 μs if the monitor is operative. The total increase in execution time has been found to be less than 3% if the time feature is not used and approximately 15% if the timer is used. The software time count has been found to be accurate to within 10% of the actual time with one-tenth of a minute accuracy.

   The maximum time limit is specified on the JOB card and overflow is considered a terminal condition. The timer feature is primarily used in closed-shop, batch processing operations where quite lengthy jobs are often run.

All of the above options are enabled or suppressed by sense switch settings when the subroutines are loaded.

3. Multiple errors in one statement can be detected. This reduces the number of precompiler passes necessary to produce an error free program.

4. Error checking is to a higher level than is required by the compiler in order to produce valid code. For instance, key words are generally identified by only one or two letters in the compiler but are checked for six letters in the precompiler.

Some of the major errors detected are

1. Possibly undefined symbol

   Any variable which has not been defined in a previous statement (in terms of card order, not logical flow) will be typed out as a possible undefined symbol when it is used. This procedure catches most undefined symbols. Statement numbers which are undefined at the end of the program will be typed out.

2. Branch to FORMAT Statement

3. Duplicate Statement Numbers

4. All DO Loop Errors

   These include overlapped DO loops, modification of DO parameters within the loop, end of theloop is previous statement, etc.

5. All Subscripting Errors

   These include contradictions involving the number of subscripts on a variable and complete checking of subscript structure.

6. All FORMAT Errors

7. All Procedure Errors

8. I/O Statement References non-FORMAT Statement

9. Exponentiation of Fixed Quantity

10. All Syntax Errors (misplaced commas, etc.) in Other Statements

In addition, some logical errors and some potential errors such as program possibly too large are detected.

The precompiler symbol table is the same size as that of the compiler so that compiler symbol table overflow conditions can be detected.

The following special implementation restrictions exist but, aside from the first, they have no practical implications in our environment:

1. A FORMAT statement must precede the first I/O statement to reference it.

2. Maximums of three continuation cards, five nested DO's and five levels of exponentiation are allowed.

3. A maximum of twenty symbols can be undefined at any time.

The important known flaws in the precompiler are

1. There is no way to detect all undefined symbols in a general program. The likelihood of undefined symbols at execution causing checkstops is reduced, however, because the monitor fills in the unused core with fixed point constants. This produces invalid results in the case of an undefined symbol but prevents the subroutines from being destroyed by the Transmit Field instruction.

2. The appearance of a variable in a DIMENSION statement causes that variable to be considered defined.

3. Arguments of functions are incompletely checked but, this will not produce checkstop errors. Mixed mode expressions in the argument will be detected.

W. 2. 6
(a)

ABSTRACT

The following describes and addition to the programming system for the IBM 1800 computer. The expanded system will support up to 16 remote teletype terminals being used in a time-sharing environment primarily to solve real-time information processing problems.

INTRODUCTION

1800 Hardware

The 1800 CPU makes extensive use of hardware interrupt levels and data channels to operate its standard data processing I/O equipment. In addition the 1800 can have analog and digital I/O capable of communicating directly with almost any kind of equipment. The terminal system uses 2 words of digital input and 1 word of digital output (16 bits/word) to control all 16 terminals simultaneously. No data channels or additional hardware are employed.

TSX Programming System

The 1800 programming system provides many conveniences. Programs are of two types, process and non-process.

Process programs can be initiated by externally generated interrupts or can be queued by any program for execution. They can be a part of the system skeleton which is in core at all times or they can be kept on disk in core-image form. Process programs have highest priority and generally use some analog or digital I/O.

Non-process programs ordinarily do not use the process (analog and digital) I/O. They are usually stacked jobs of a conventional type to be run under the non-process monitor.

When time-sharing, the system does the following:

(1). Runs non-process programs for background (job shop).

(2). Periodically checks the queue for process programs.

(3). Permits externally or internally generated interrupts to load and execute process programs at any time.

Both (2) and (3) require that the non-process job be saved and restored when the process work is finished.

The Terminal System

The terminals extend the time-sharing capabilities of the system by allowing up to 16 users to communicate with the system simultaneously.

Terminal activities include:

(1). Queing process programs for execution.

(2). Communicating with programs during execution.

(3). Programming in NUtran (conversational fortran).

Many other activities are planned.

Most of the terminal system capabilities are achieved by simply making the standard TSX functions more accessible. The only programming efforts unique to the terminal system are the communications controller (simulated by an in-skeleton program), and the time-slicing of terminal service programs.

DR JOHN MANIOTES
COMPUTER TECHNOLOGY DEPT.
PURDUE UNIVERSITY
CALUMET CAMPUS
HAMMOND, IN 46323