USERS MANUAL

COMPUTATION CENTER
STANFORD UNIVERSITY
STANFORD, CALIFORNIA

USERS MANUAL

Second Edition
Second Revision, May 1968

Computation Center
Campus Facility
Stanford University

Revised May 1968

FOREWORD

This Users Manual was written to acquaint the Users of the Stanford Computation Center with the facilities and services offered by the Campus Facility.

In particular, this Second Edition describes the facilities available and the methods for using the IBM 360/67.

Users can expect revisions to this manual as new systems and methods for the 360/67 develop. Changes may also be made to the present material in order to rectify omissions and to clear up possible ambiguities of which we may or may not be aware. In this regard, the Users Services Group, Campus Facility, would welcome constructive contributions from Users for future issues.

Eileen Jensen,
Editor

May 1968

To Our Users. . .

The enclosed material updates the Second Edition of the Campus Facility Users Manual. This is the second revision of the Second Edition. The following changes should be made.

Pages to be Inserted	Pages to be Removed	Pages to be Inserted	Pages to be Removed
title page	title page	3-24	3-24
Foreword	Foreword	3-26	3-26
iii thru vii	iii thru vi	3-29 and 3-30	3-29 and 3-30
2-1	2-1	3-33a and 3-33b	---
2-2 and 2-2a	2-2 and 2-2a	3-34	3-34
2-5	2-5	3-45 thru 3-52	---
2-8	2-8	4-6 and 4-7	4-6 and 4-7
2-10	2-10	4-16	4-16
2-11 thru 2-12b	2-11 and 2-12	5-4	5-4
2-15 and 2-16	2-15 and 2-16	6-8 thru 6-14a	6-8 thru 6-14
2-19 thru 2-21	2-19 thru 2-21	6-21	6-21
3-2 thru 3-2b	3-2	6-31	6-31
3-17a	---	6-35	6-35
3-19	3-19	6-35a	---

Changes to the text are indicated by a vertical bar to the left of the change. New material is indicated by the phrase "Added May 1968" in the upper right hand corner of the page.

Future revisions and updates are sent automatically and without charge to all holders of the Campus Facility Users Manual.

Eileen Jensen,
Editor

February 1968

To Our Users. . .

The attached material updates the Second Edition of the Campus Facility Users Manual. This is the first revision of the Second Edition. The index tabs should be placed at the beginning of each appropriate section.

Pages to be Inserted	Pages to be Removed
title page	title page
Foreword	Foreword
iii thru vi	iii thru vi
1-1 thru 1-6c	1-1 thru 1-7
2-2 thru 2-2c	2-2
2-5	2-5
2-6	2-6
2-15	2-15
2-17 thru 2-20	2-17 thru 2-20
2-22a	2-22a
2-23 thru 2-25	2-23 thru 2-26
3-4 thru 3-5a	3-4 and 3-5
3-13	3-13
3-15	3-15
3-21	3-21
3-23 thru 3-26	3-23 thru 3-26
3-28	3-28
3-34 thru 3-44	3-34 thru 3-38
4-7 and 4-8	4-7 and 4-8
5-2	5-2
6-8 thru 6-41	6-8 thru 6-31

Future revisions and updates are sent automatically and without charge to all holders of the Campus Facility Users Manual.

Eileen Jensen,
Editor

Revised May 1968

TABLE OF CONTENTS

Revised May 1968

LIST OF ILLUSTRATIONS

1. INTRODUCTION

1.1 The Stanford Computation Center

1.1.1 History

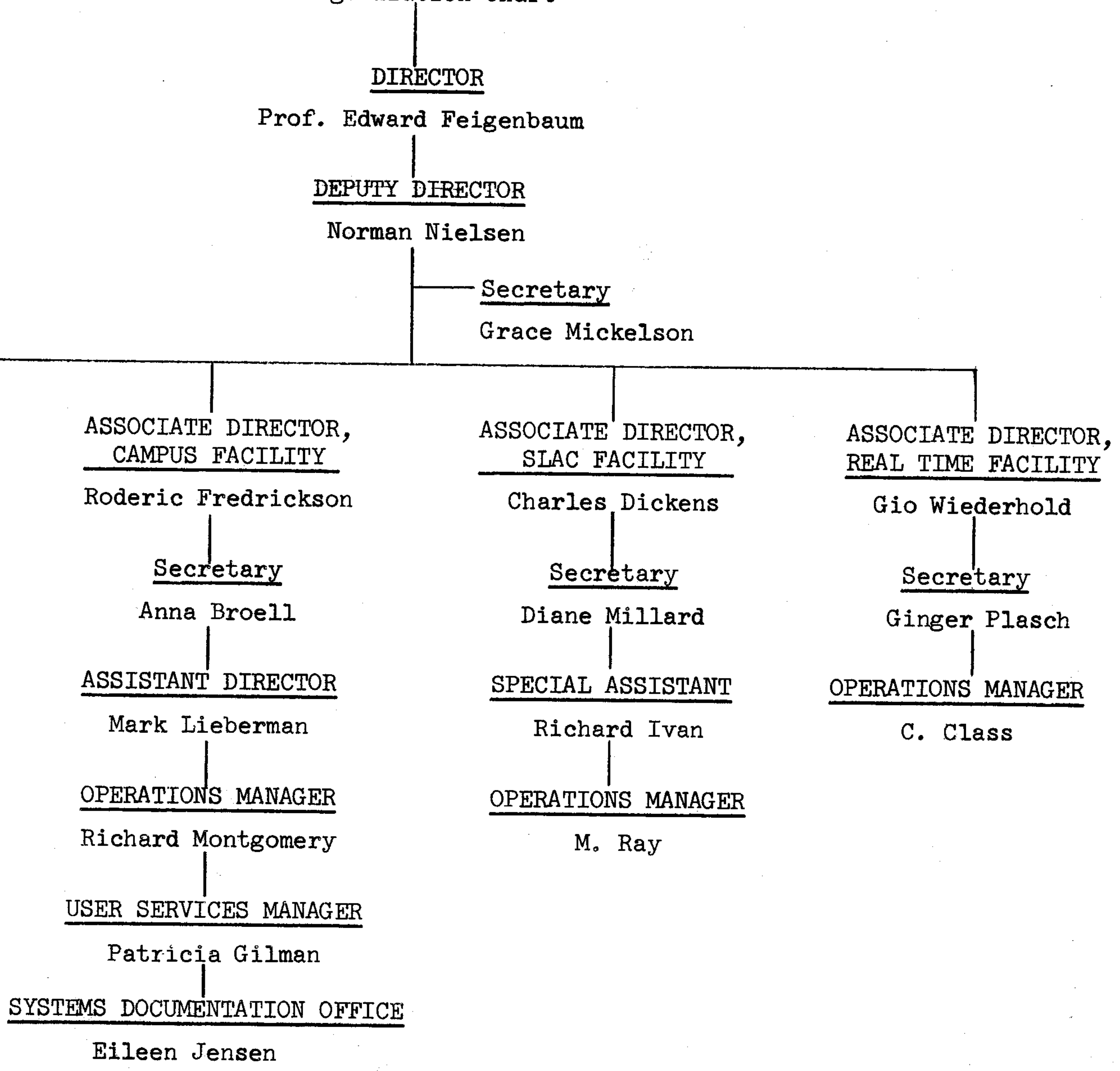
The Stanford Computation Center was formed in March 1953, with the installation of an IBM Card Programmed Calculator in the basement of Encina Hall. As computing grew at Stanford, so did the role of the Computation Center, until it came to encompass the major computing activities at Stanford. In order to service this growing and sophisticated user community, the Computation Center reorganized as a set of several facilities. The computers located in Pine Hall became the Campus Facility. As work on the Stanford Linear Accelerator Center neared completion, the SLAC Facility was formed to provide the special computation requirements associated with it. ACME (A Computer for MEdical Research) is a research project sponsored by the National Insitutes of Health for the purpose of furthering medical research with an on-line computer system tailor-made for data acquisition and the control of experiments. The Real Time Facility was created to handle the needs of this project.

1.1.2 Organization

While each of the facilities is a physically separate center, the responsibility for coordination, control, and operation is vested in the Computation Center as a whole. In this way, Stanford's computer expertise is consolidated to the benefit of everyone. The following chart indicates the organization of the Computation Center and lists the key personnel for each group. A personnel directory is also included and provides a handy list of telephone numbers which are relevant to the users of the various facilities.

STANFORD COMPUTATION CENTER

Organization Chart



Revised February 1968

COMPUTATION CENTER PERSONNEL DIRECTORY

NAME	TITLE/FUNCTION	LOCATION	EXTENSION
Stanford Computation Center			
E. Feigenbaum	Director	Polya 115	4879
N. Nielsen	Deputy Director	Polya 119	4885
G. Mickelson	Secretary, Director's Office	Polya 117	4770
Campus Facility			
R. Fredrickson	Associate Director	Polya 109	2897
M. Lieberman	Assistant Director	Polya 107	4943
A. Broell	Secretary	Polya 105	2897
R. Montgomery	Operations Manager	Pine 186	4873
P. Gilman	User Services Manager	Polya 156	4400
E. Jensen	Systems Documentation	Pine 185	4877
SLAC Facility			
C. Dickens	Associate Director	SLAC	8545
D. Millard	Secretary	SLAC	8545
R. Ivan	Special Assistant	SLAC	8655
M. Ray	Operations Manager	SLAC	8545
Real Time Facility			
G. Wiederhold	Associate Director	S009	5818
G. Plasch	Secretary	S009	5818
C. Class	Operations Manager		5903
K. Holtz	Engineer	S007D	5820
Inter Facility			
R. Jamtgaard	Associate Director	Polya 112	3104

1.2 The Facilities

1.2.1 Real Time Facility

Function and Equipment: The Real Time Facility provides specialized real time capabilities for medical researchers. To this end the ACME/PL programming language and the ACME time-sharing system were specially constructed to provide real-time data collection capability. This system is now running on an IBM 360/50 - 1800 system located at the Stanford Medical School. This system and the planned SDS Sigma 5 in the Facility's Hybrid Computer Laboratory will also be available to non-medical researchers who have real time requirements. A diagram of the hardware configuration (Figure 1.1) is at the end of this section.

1.2.2 SLAC Facility

Function and Equipment: The SLAC Facility was created for the sole purpose of providing computational support for the accelerator. A 360/75 with a half-million byte memory is currently installed, and a 360/91 is scheduled to replace it in 1968. A diagram of the hardware configuration (Figure 1.2) is at the end of this section.

1.2.3 Campus Facility

Function and Equipment: The Campus Facility provides service for the computing needs of the Stanford Community. To this end it provides standard batch processing service on its 360/67 and text editing and remote job entry from terminals is available. A diagram of the hardware configuration (Figure 1.3) is at the end of this section. Descriptions of the system's software capabilities may be found in Sections 3, 4 and 6 of this manual.

1.3 Inter-facilities

The centralized organization of the various facilities allows the Computation Center to offer a number of common services. In this way users of all facilities are able to share the benefits arising from such cooperation. A few of these inter-facility services are:

Program Library: programs incorporated in the program library by one facility can often be included in the libraries of other facilities. Thus a much larger library of programs can be certified and offered to the user.

Systems Documentation Office: this office maintains a record of all printed information available on the systems and program library used at Stanford. It also provides a central source for procuring system manuals, program documentation, SHARE programs, etc.

Operations: the operational requirements of the facilities are coordinated so that operators or machine time can be provided one facility by another in the case of protracted hardware difficulties. In this way more consistent service can be provided to all of our users.

Systems expertise: a close liaison is maintained between the systems personnel at the various facilities. Thus a bug discovered and corrected at one facility can be immediately corrected at the other facilities without each group having to "re-invent the wheel." Again, this enables us to provide better computational service to our users.

Computer Science Library: this library in Polya Hall is also available as a resource for the Computation Center. It contains hundreds of volumes on programming, numerical analysis, and nonnumeric computations. Also available is a set of the major periodicals in the computer field.

1.4 User Eligibility

The services of the Campus Facility are available to all faculty, staff, and students of the University and to University-related personnel off-campus for use in connection with research and instructional activities. Special arrangements for non-university work may be made with the Associate Director, Campus Facility, Room 105, Polya Hall, Ext. 2897.

Use of the other facilities is also restricted by these same conditions. However, since the other facilities are not general purpose facilities, there are additional restrictions on their use. The SLAC Facility is open only to SLAC personnel working on projects connected with the research work being performed at the linear accelerator. Likewise, the Real Time Facility is restricted to Stanford Medical School researchers and projects requiring real-time data collection.

Users who are uncertain about their eligibility to use a particular facility should contact the Director of the Computation Center or the Associate Director of the facility in question.

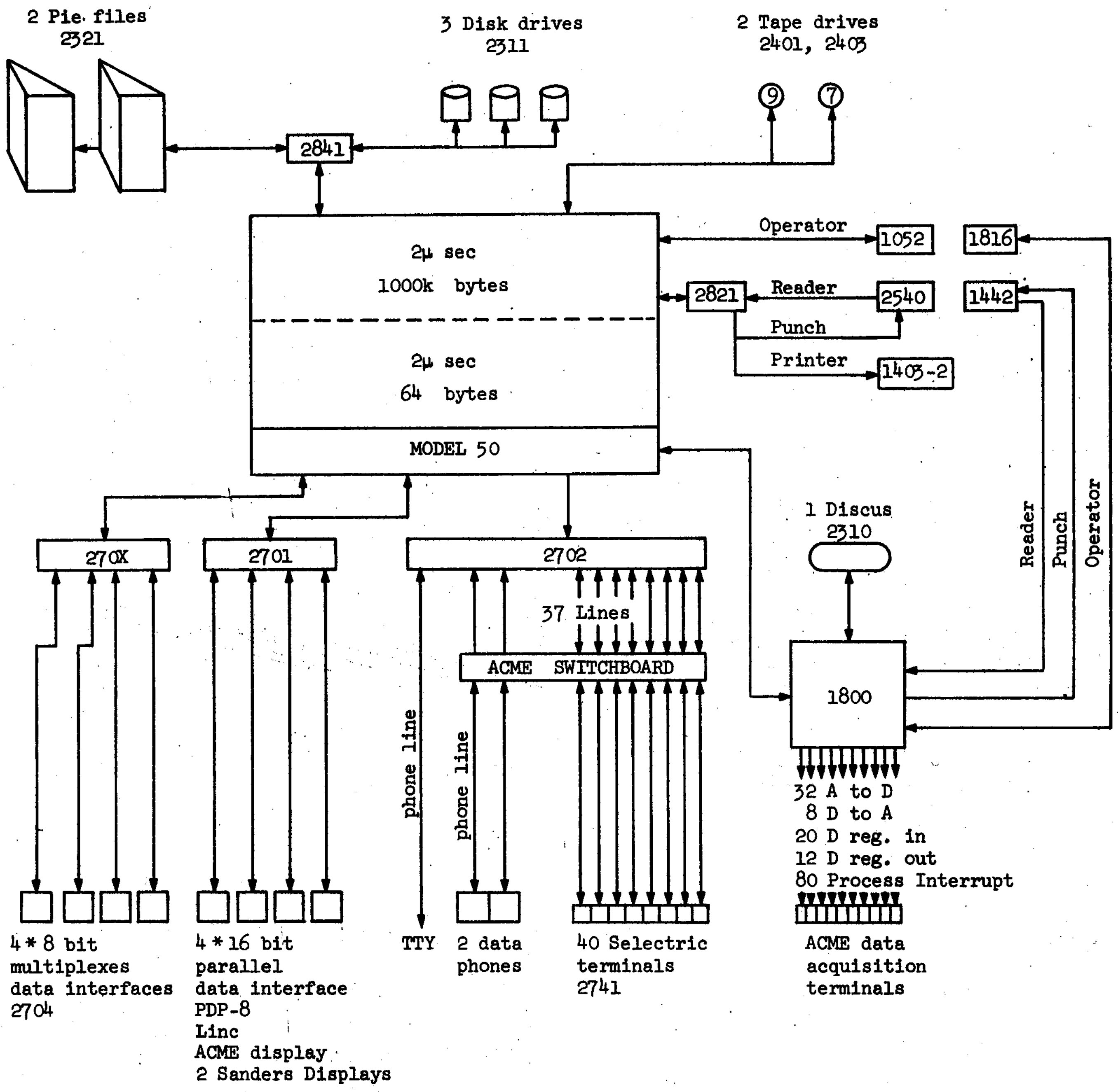


Figure 1.1 Real Time Facility Machine Configuration

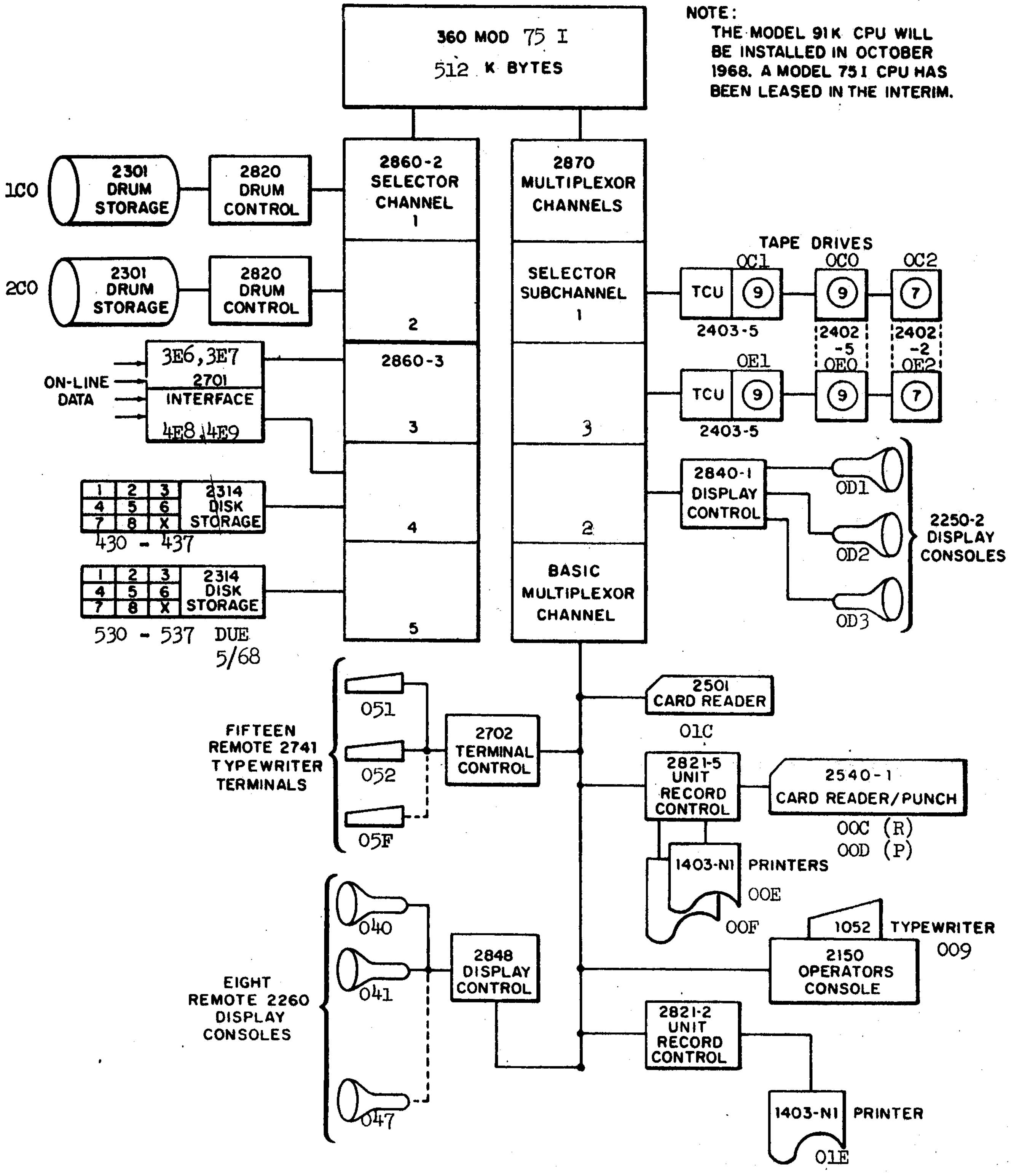


Figure 1.2 SLAC Facility Machine Configuration

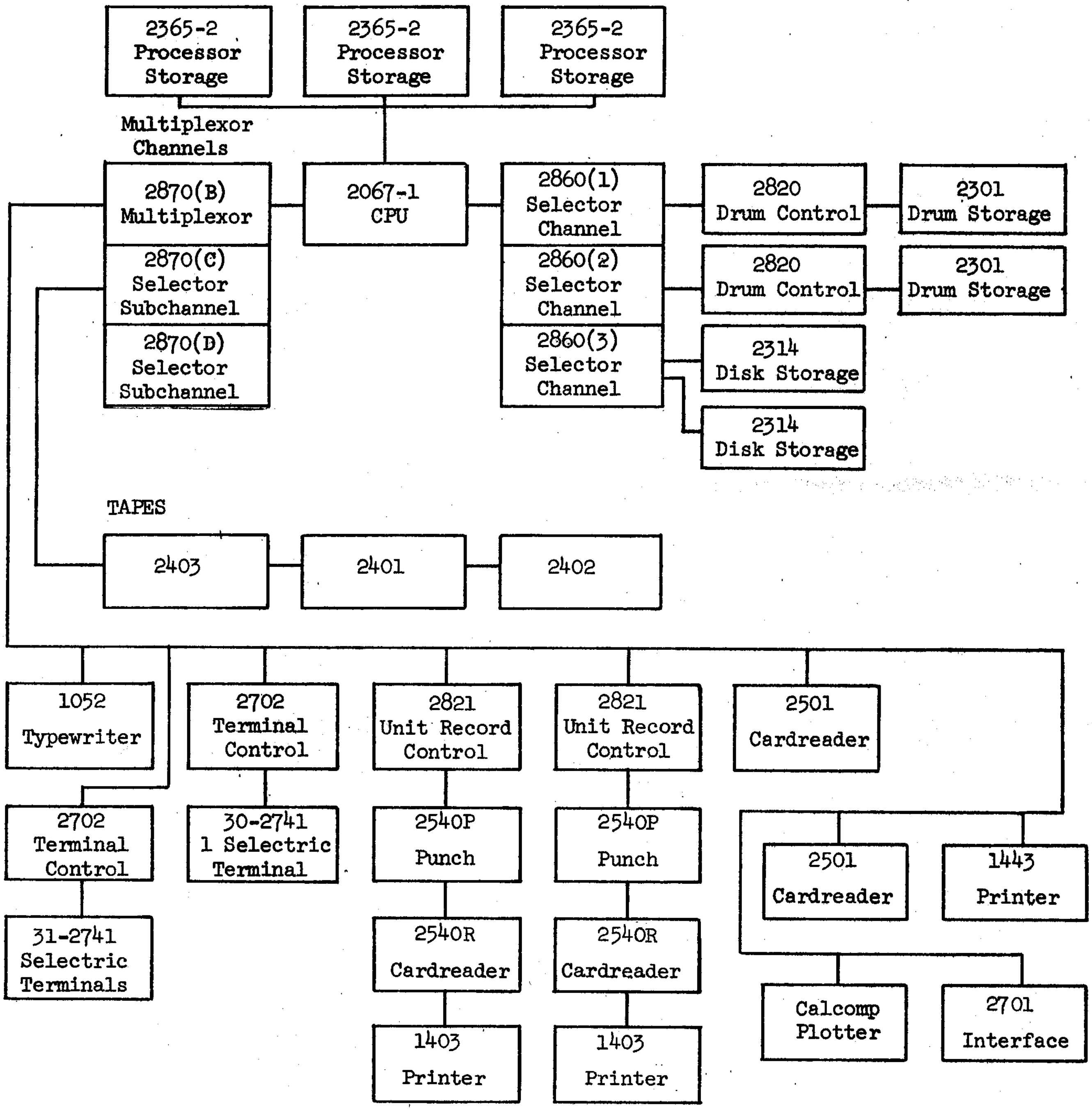


Figure 1.3 Campus Facility Machine Configuration

Revised May 1968

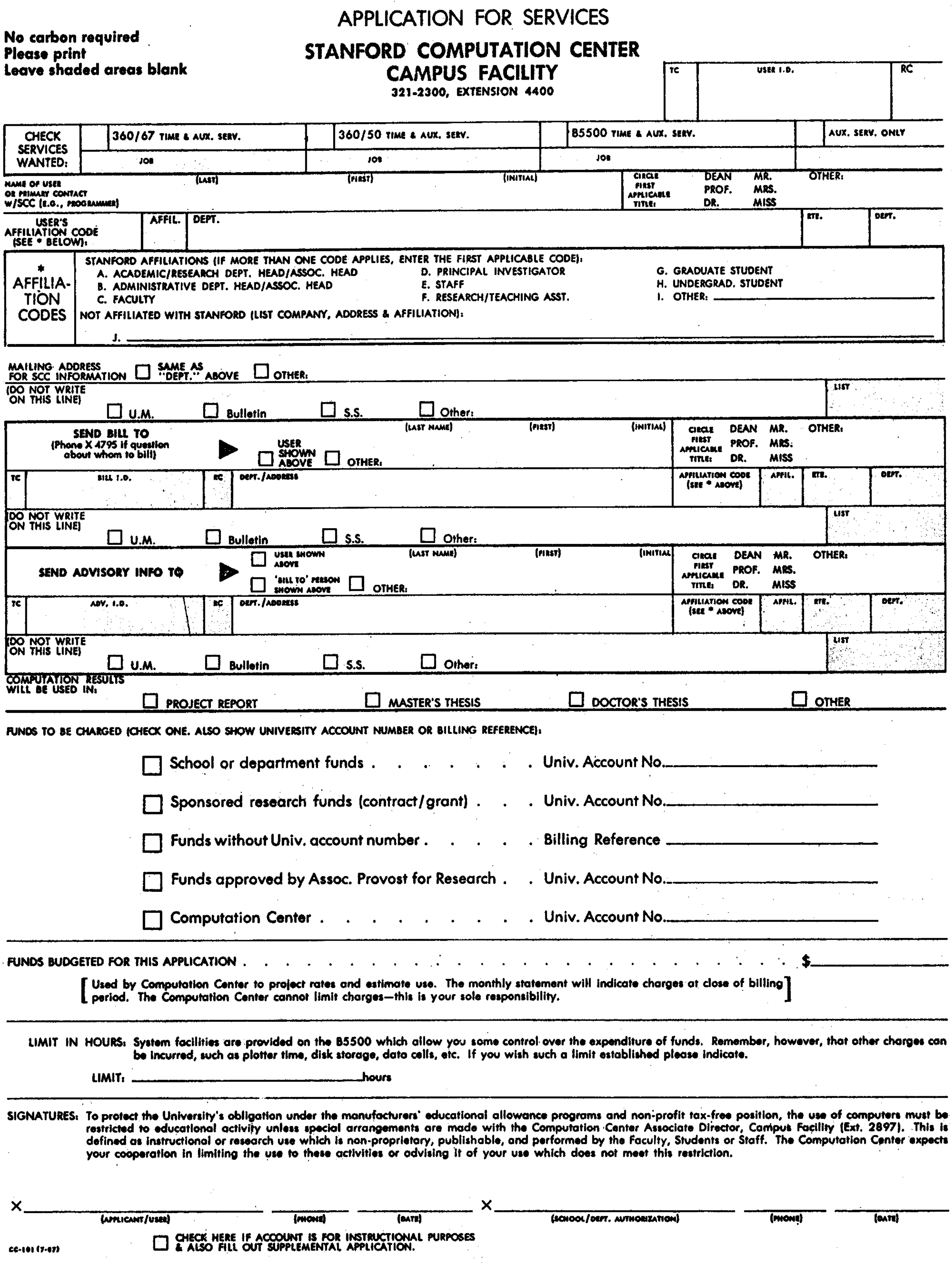
2. SERVICES AND OPERATIONS

2.1 Administrative

2.1.1 Account Numbers and Eligibility

The facilities and services of the Stanford Computation Center, Campus Facility, are available for use in connection with research and computer-related classwork to the faculty, staff, and students of the University; to University-related personnel off-campus; and to non-University users by special arrangement.

- o Computer time is granted on the basis of a formal request made on the standard application form and submitted to the Accounting Office in Polya Hall. (The application form and supplemental application form are illustrated in Figure 2.1 and Figure 2.2.) Approval of this form entitles the applicant to the use of all services; i.e., the computer, keypunches, and tabulating machines, as well as supplies needed, such as paper and cards.
- o All work done on the computer requires the approval of the authorized faculty member or administrator.
- o Costs of the computer use are recharged at current schedules. (See Rates, Section 2.1.2.)
- o Instructional Use: Total time of computer usage in courses will be charged against department budgets at the current rates. Departments should contact the office of the Associate Provost for Computing for assistance in procuring funds to supplement department budgets for this purpose.
- o Billing: At the end of each calendar month the User receives a monthly statement showing how much computer time was charged to his account during the billing period. The billing cut-off date is the 20th of each month.
- o Closing of an Account: At the conclusion of a project, the User should notify the Campus Facility Accounting Office in writing that the project has ended.



No carbon required
Please print
Leave shaded areas blank

APPLICATION FOR SERVICES
STANFORD COMPUTATION CENTER
CAMPUS FACILITY
321-2300, EXTENSION 4400

TC	USER I.D.	RC

CHECK SERVICES WANTED:	360/67 TIME & AUX. SERV.	360/50 TIME & AUX. SERV.	B5500 TIME & AUX. SERV.	AUX. SERV. ONLY
	JOB	JOB	JOB	

NAME OF USER OR PRIMARY CONTACT W/SCC (E.G., PROGRAMMER) (LAST) (FIRST) (INITIAL) CIRCLE FIRST APPLICABLE TITLE: DEAN PROF. DR. MR. MRS. MISS OTHER:

USER'S AFFILIATION CODE (SEE * BELOW): AFFIL. DEPT. RTE. DEPT.

* AFFILIATION CODES

STANFORD AFFILIATIONS (IF MORE THAN ONE CODE APPLIES, ENTER THE FIRST APPLICABLE CODE):

A. ACADEMIC/RESEARCH DEPT. HEAD/ASSOC. HEAD	D. PRINCIPAL INVESTIGATOR	G. GRADUATE STUDENT
B. ADMINISTRATIVE DEPT. HEAD/ASSOC. HEAD	E. STAFF	H. UNDERGRAD. STUDENT
C. FACULTY	F. RESEARCH/TEACHING ASST.	I. OTHER: ______

NOT AFFILIATED WITH STANFORD (LIST COMPANY, ADDRESS & AFFILIATION):

J. ______

MAILING ADDRESS FOR SCC INFORMATION ☐ SAME AS "DEPT." ABOVE ☐ OTHER:

(DO NOT WRITE ON THIS LINE) ☐ U.M. ☐ Bulletin ☐ S.S. ☐ Other: LIST

SEND BILL TO (Phone X 4795 if question about whom to bill) ☐ USER SHOWN ABOVE ☐ OTHER: (LAST NAME) (FIRST) (INITIAL) CIRCLE FIRST APPLICABLE TITLE: DEAN PROF. DR. MR. MRS. MISS OTHER:

TC BILL I.D. RC DEPT./ADDRESS AFFILIATION CODE (SEE * ABOVE) AFFIL. RTE. DEPT.

(DO NOT WRITE ON THIS LINE) ☐ U.M. ☐ Bulletin ☐ S.S. ☐ Other: LIST

SEND ADVISORY INFO TO ☐ USER SHOWN ABOVE ☐ 'BILL TO' PERSON SHOWN ABOVE ☐ OTHER: (LAST NAME) (FIRST) (INITIAL) CIRCLE FIRST APPLICABLE TITLE: DEAN PROF. DR. MR. MRS. MISS OTHER:

TC ADV. I.D. RC DEPT./ADDRESS AFFILIATION CODE (SEE * ABOVE) AFFIL. RTE. DEPT.

(DO NOT WRITE ON THIS LINE) ☐ U.M. ☐ Bulletin ☐ S.S. ☐ Other: LIST

COMPUTATION RESULTS WILL BE USED IN: ☐ PROJECT REPORT ☐ MASTER'S THESIS ☐ DOCTOR'S THESIS ☐ OTHER

FUNDS TO BE CHARGED (CHECK ONE. ALSO SHOW UNIVERSITY ACCOUNT NUMBER OR BILLING REFERENCE):

☐ School or department funds . . . . Univ. Account No.______

☐ Sponsored research funds (contract/grant) . . . Univ. Account No.______

☐ Funds without Univ. account number . . . . . Billing Reference ______

☐ Funds approved by Assoc. Provost for Research . . Univ. Account No.______

☐ Computation Center . . . . Univ. Account No.______

FUNDS BUDGETED FOR THIS APPLICATION . . . . $______

[Used by Computation Center to project rates and estimate use. The monthly statement will indicate charges at close of billing period. The Computation Center cannot limit charges—this is your sole responsibility.]

LIMIT IN HOURS: System facilities are provided on the B5500 which allow you some control over the expenditure of funds. Remember, however, that other charges can be incurred, such as plotter time, disk storage, data cells, etc. If you wish such a limit established please indicate.

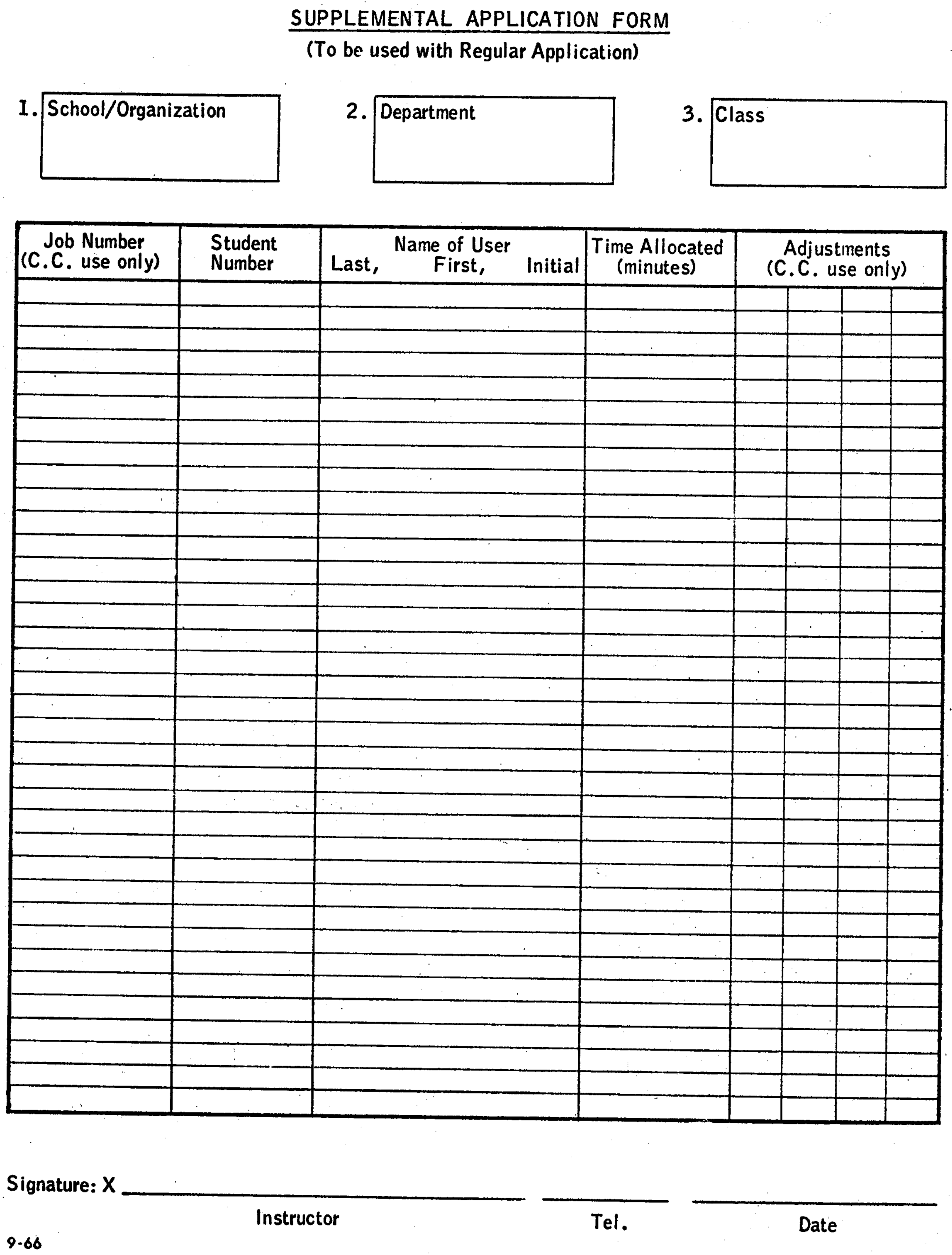
LIMIT: ______hours

SIGNATURES: To protect the University's obligation under the manufacturers' educational allowance programs and non-profit tax-free position, the use of computers must be restricted to educational activity unless special arrangements are made with the Computation Center Associate Director, Campus Facility (Ext. 2897). This is defined as instructional or research use which is non-proprietary, publishable, and performed by the Faculty, Students or Staff. The Computation Center expects your cooperation in limiting the use to these activities or advising it of your use which does not meet this restriction.

X______ (APPLICANT/USER) ______ (PHONE) ______ (DATE) X______ (SCHOOL/DEPT. AUTHORIZATION) ______ (PHONE) ______ (DATE)

CC-101 (7-67) ☐ CHECK HERE IF ACCOUNT IS FOR INSTRUCTIONAL PURPOSES & ALSO FILL OUT SUPPLEMENTAL APPLICATION.

Figure 2.1



SUPPLEMENTAL APPLICATION FORM

(To be used with Regular Application)

1. School/Organization	2. Department	3. Class

Job Number (C.C. use only)	Student Number	Name of User Last, First, Initial	Time Allocated (minutes)	Adjustments (C.C. use only)			

Signature: X ______________________ __________ ______________

Instructor Tel. Date

9-66

Figure 2.2

Revised May 1968

2.1.2 Rates

The following rate schedule reflects the rates that went into effect October 21, 1967 and also reflects the effect of the new scheduling algorithm that went into operation on February 20, 1968. These rates apply to the 360/67 services offered by the Stanford Computation Center, Campus Facility.

Interim 360/67 Rate #3*

Processor Charge:

PRIORITY OPTIONS

Execution Order	Available Execution Priority		Maximum Execution Time (minutes)	Priority Service Charge	Rate		
	WYLBUR	Card Reader			0-5 mins.	5-10 mins.	10+ mins.
1	URGENT	URGENT	None	$25	$8/min	$12/min	$16/min
2	PRIORITY	--	None	$ 5	$8/min	$12/min	$16/min
3	--	PRIORITY	None	$ 5	$8/min	$12/min	$16/min
4	EXPRESS	--	2	None	$8/min (2 min. limit)		
5	STANDARD	STANDARD	None	None	$8/min	$12/min	$16/min
6	IDLE	IDLE	1	None	$4/min (1 min. limit)		
7	OVERNITE	OVERNITE	None	None	$7/min***		

*** This is a declining rate dropping $.50 per minute each successive half hour of continuous running. The minimum rate permissible is based on the average rate per hour for the run which is floored at $300 per hour.

EXPRESS is the default priority assigned jobs entered from the terminals when the estimated job time is 2 minutes or less; otherwise, for jobs entered from the terminals with an estimated job time > 2 minutes, the priority is STANDARD. The default priority assigned jobs entered from the card readers is STANDARD.

The purpose of the IDLE priority and its low rate is to attract a backlog of jobs to be run at times when the machine would otherwise be idle. The relatively short 1 minute limit is set to minimize possible impact on higher priority jobs. A job submitted which has an execution estimate > 1 minute is rescheduled by the system to an OVERNITE priority and OVERNITE rates are applied. The print priority for an IDLE execution job is IDLE.

Revised May 1968

Output Lines:

Service Order	Print Priority		Rate
1	URGENT		$5.00 + $1.00/thousand lines
2	PRIORITY		$1.00/thousand lines
3	STANDARD	< 2k lines	$0.50/thousand lines
4	STANDARD	< 5k lines	$0.50/thousand lines
5	STANDARD	< 15k lines	$0.50/thousand lines
6	STANDARD	> 15k lines	$0.50/thousand lines
7	IDLE		$0.50/thousand lines
8	OVERNITE		$0.50/thousand lines
9	NOPRINT		---

(See the notes at the end of the rate schedule on how to use priority.)

Other Rates:

2741 Terminal Access Duration Charge and CPU Editing (WYLBUR)	$4/terminal/hour
2741 Terminal and Line Rental:**	
Business Telephone Line Connection	$125.47/month (as of July 1, 1968, $138.91/month) + $135 initial installation.
Private Line Concentrator Distributor System	$118.42/month (as of July 1, 1968, $131.86/month) + $130 initial installation.
Disk Storage (2314) (reserved file, not scratch use)	1¢/track (~7,188 bytes)/day.
Tape Storage	$2/tape/month.
Input Cards	25¢/thousand cards.
Output Cards	$1 + $4/first thousand cards, $1.25/additional thousand/run.
Plotting	$25/hour for Off-line Plotting and $1 + 10¢/minute for On-line Plotting (when available).
Forms Change	$5/change.
Printer Adjustment Service	$5/adjustment.

Revised February 1968

Key Punching	$4.25/hour.
Special Services (As described in Section 2.3.4)	$4.25/hour.

*NOTE: The 2741 terminal rates are new. Other changes from Interim Rate #2 include overnight CPU time at $7.00/minute, a reduction to 1¢/track/day for 2314 disk storage, and a reduction for output cards beyond the first 1,000 cards on any one run.

**NOTE: Terminal, line rental, and installation costs cannot be charged to Provost's Funds for computer usage.

NOTE 1: How to use priority from the terminal.

Priority control over both execution and printing may be exercised when issuing the RUN, LIST OFFLINE, and PUNCH commands. If no explicit priority is made, the execution priority is assumed to be EXPRESS and the print priority is assumed to be STANDARD (the service order depends on the number of lines).

EXPRESS runs having an estimated execution time in excess of two (2) minutes are rescheduled by the system to STANDARD.

If only one priority option is indicated, it is assumed for both execution and printing.

```
USAGE:  RUN <execution option>,<print option> <other options>
        LIST OFFLINE <execution option>,<print option> <other
                     options>
        PUNCH <execution option>,<print option> <other options>
EXAMPLE:  COMMAND ? run idle bin 555
          COMMAND ? run p,s b 666
```

NOTE 2: How to use priority from the card readers.

Priority control over both execution and printing may be exercised when entering jobs from the card readers. If no explicit priority is made (by the /*RUN card before the job card) the job will be scheduled for execution and printing under STANDARD priority.

There is no EXPRESS priority available to a job entered from the card readers.

If only one priority option is indicated, it is assumed for both execution and printing.

```
USAGE:  /*RUN              <execution option>,<print option>
```

Options must be specified starting in column 16, separated by a comma, and with <u>no interspersed blanks</u>. A short form consisting of just the first character may be used in place of the full word.

```
EXAMPLE:  /*RUN            PRIORITY,STANDARD
          /*RUN            P,S
```

Remember: <u>for a completely standard run no /*RUN card is required</u>.

2.1.3 360/67 Refund Policy and Procedure

The Computation Center, Campus Facility, may refund the time charged to a User's account for a 360/67 run which failed due to circumstances beyond the User's control. It is our opinion that the responsibility for obtaining the information necessary for using the computing facilities lies with the User. In considering refund requests, we take the position that the programmer is responsible for writing his program and preparing his control cards in a manner that provides complete protection for his job. Users who believe they have incurred an unfair charge should ask the Consultants, Room 182, Pine Hall, for assistance. The Consultant, together with the User, will examine the program deck, listing, and output. If the Consultant feels the problem could not have been prevented by the User, he will suggest a Refund Request Form be completed, subject to the following considerations:

1. No refunds will be given for jobs whose total run time is less than 2 minutes.
2. Jobs of greater than 2 minutes duration which run longer than their time estimate may receive a refund. Refunds will only be given on the actual run time--minus 2 minutes--exceeding the maximum time specified on the JØB card.
3. Jobs whose output exceeds the line and/or punched card estimates will be terminated as soon as possible. The User will be charged for the excess output.
4. Users will be charged for setup time for those jobs whose setup cannot be completed while a prior job is running.
5. The message option on the JØB card must have been "MSGLEVEL=1".

Extenuating circumstances regarding any of the above items should be pointed out to the Consultant.

The listing and output for the suspect run must accompany the request. If a system or machine error is suspected, a complete source listing (including a listing for all subprograms) and an interpreted copy of the run deck must also be submitted.

Written notification of the disposition of the request will be sent to the User as soon as possible.

Known problems and their solutions will be announced in the Bulletin, and subsequent refund requests stemming from these problems will not be considered favorably.

2.1.4 Campus Facility Personnel Directory

The following directory provides a handy list of telephone numbers which are relevant to users of the Campus Facility.

NAME	TITLE/FUNCTION	LOCATION	EXTENSION
Briggs, Norman R.	Head of Procurement and Plant Office	Polya 108	4372
Bruguera, Jorge	Head Librarian, Computer Science Library	Polya 170	2894
Fredrickson, Roderic	Associate Director, Campus Facility	Polya 109	2897
Gilman, Patricia	User Services Manager	Polya 156	4400
Hagan, Helen	Asst. Accountant (job numbers, acct. information)	Polya 164	4795
Jensen, Eileen	Systems Documentation Office	Pine 185	4877
Konrad, Dianna	Publications Processing Coordinator	Polya 153	4374
Lieberman, Mark	Assistant Director, Campus Facility	Polya 105	4943
McGowan, Cecile	Secretary, Campus Facility Associate Director's Office	Polya 107	2897
Montgomery, Richard	Operations Manager	Pine 186	4873
Smith, Karen	Polya Hall Receptionist/ Information	Polya	2895
Vesanovic, Peter	Operations Supervisor	Pine 188A	4394
Wood, Connie	Keypunch Supervisor	Pine 188H	4395
Consulting		Pine 185	2046
Recorded message regarding machine and job status			4391
Operations -- general inquiries			4392

2.2 Equipment

2.2.1 Hardware

The Campus Facility IBM 360/67 incorporates the following components:

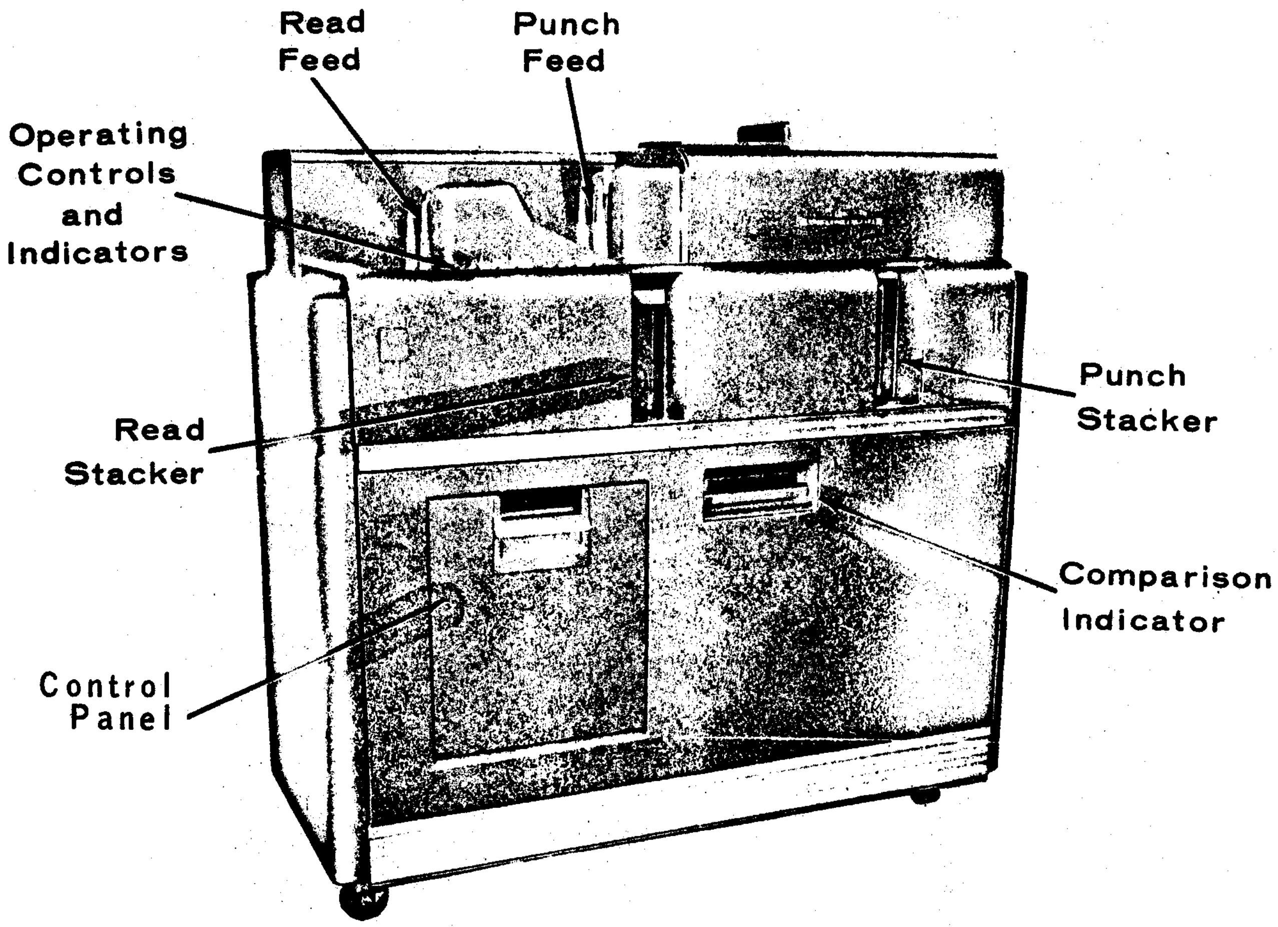
- One Central Processing Unit
- 786,432-bit bytes of core storage
- Two 7-track tape drives
- Two 9-track tape drives
- Two 2314 Disk Storage Units
- Two 1403 Printers
- Two 2540 Card Read/Punches
- Two 2501 Card Readers
- Two 2702 Transmission Control Units
- Appropriate control and console equipment

See Figure 1.3 for a complete configuration of the 360/67.

2.2.2 Unit Record Equipment

Necessary unit record equipment is available in Pine Hall, and may be operated by Users to prepare and correct punched cards and list, interpret and duplicate punched card decks. Brief operating instructions appear below. The personnel in Dispatch will be happy to assist the User in learning how to use and operate the machines. A word of caution -- in the event of a card jam or machine failure, contact a Dispatch clerk immediately and do not attempt to clear the failure or jam.

1. 519 Reproducing Punch



Revised May 1968

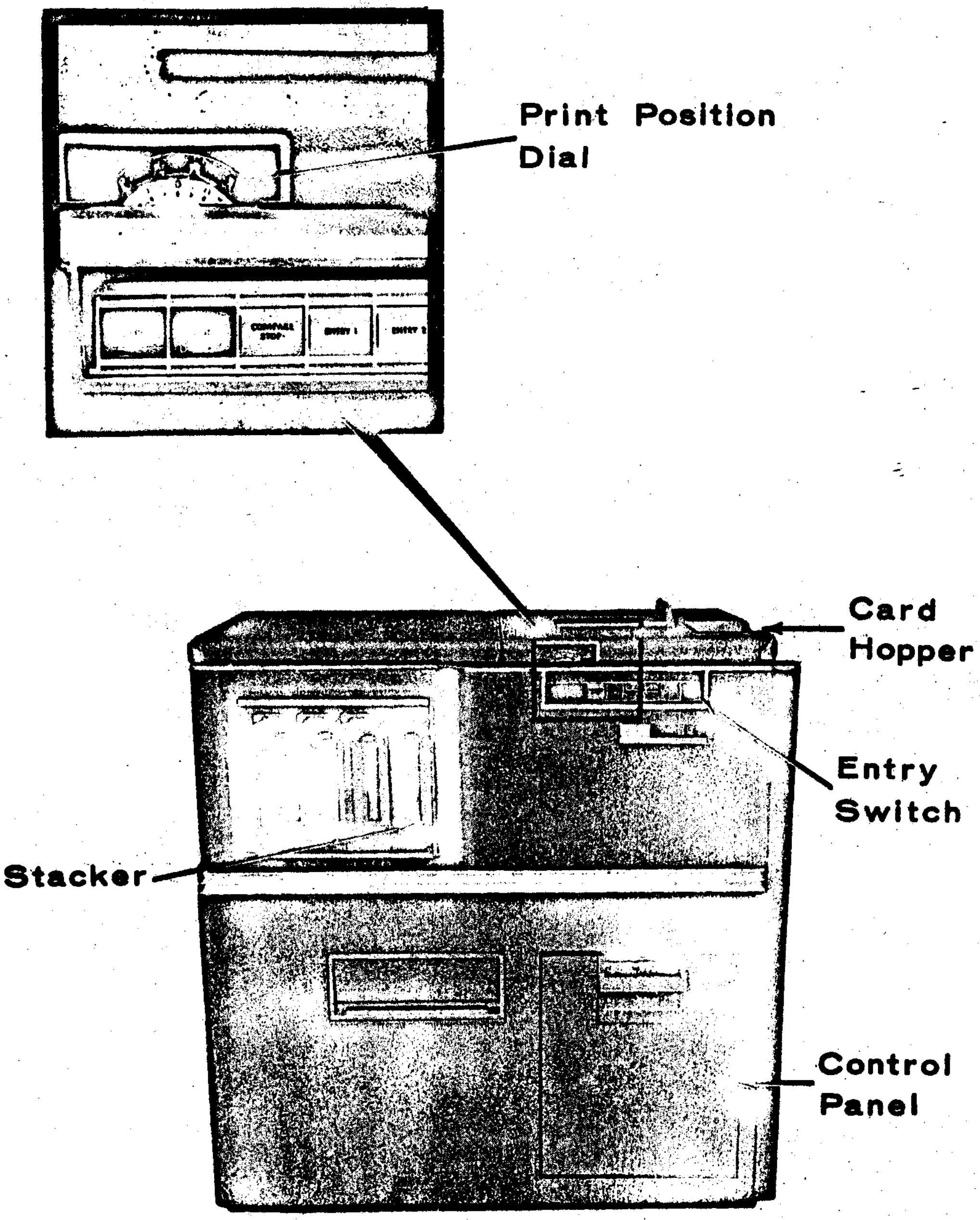
Register with a clerk at the Dispatch counter in Pine Hall for use of the 519 Reproducing Punch. The power switch is controlled from the Dispatch area. Notify the Dispatch clerk when your work is completed so that the unit can be turned off.

To duplicate a deck, place the source cards into the READ FEED with the top of the cards face down and toward your right. In the same way, place a supply of blank cards in the PUNCH FEED. Open the CONTROL PANEL cover, insert the 80 X 80 DUPLICATE Control Panel and then reclose the cover. Control panels should be handled with care. Hold down the START key for a couple of seconds. The cards will begin feeding and will fall into their respective STACKERS. Always stop the machine to replenish the blank card or source card supply. When the last source card has been read, remove the remaining cards from the PUNCH FEED and hold down the START key a few seconds until all cards are in the STACKERS.

Cards can be duplicated in columns 1-76 and punched with new sequence numbers in columns 77-80. On the Col. 1-76 DUPE and 77-80 NEW SEQ Control Panel select the switch setting desired: count by units or count by 10's. On a blank card, keypunch the starting number you want in your deck into columns 77-80. Put this card in front of your blank card supply and then load and operate the machine as explained for 80 X 80 duplicating. WARNING: The 519 Reproducing Punch cannot be used to reproduce binary cards.

For comparing, the "Compare" Control Panel is used. The master deck is put in the left-hand feed, and the reproduced deck in the right-hand feed. The machine will stop and the red "ERROR" light will glow, if a discrepancy is encountered.

2. 557 Interpreter

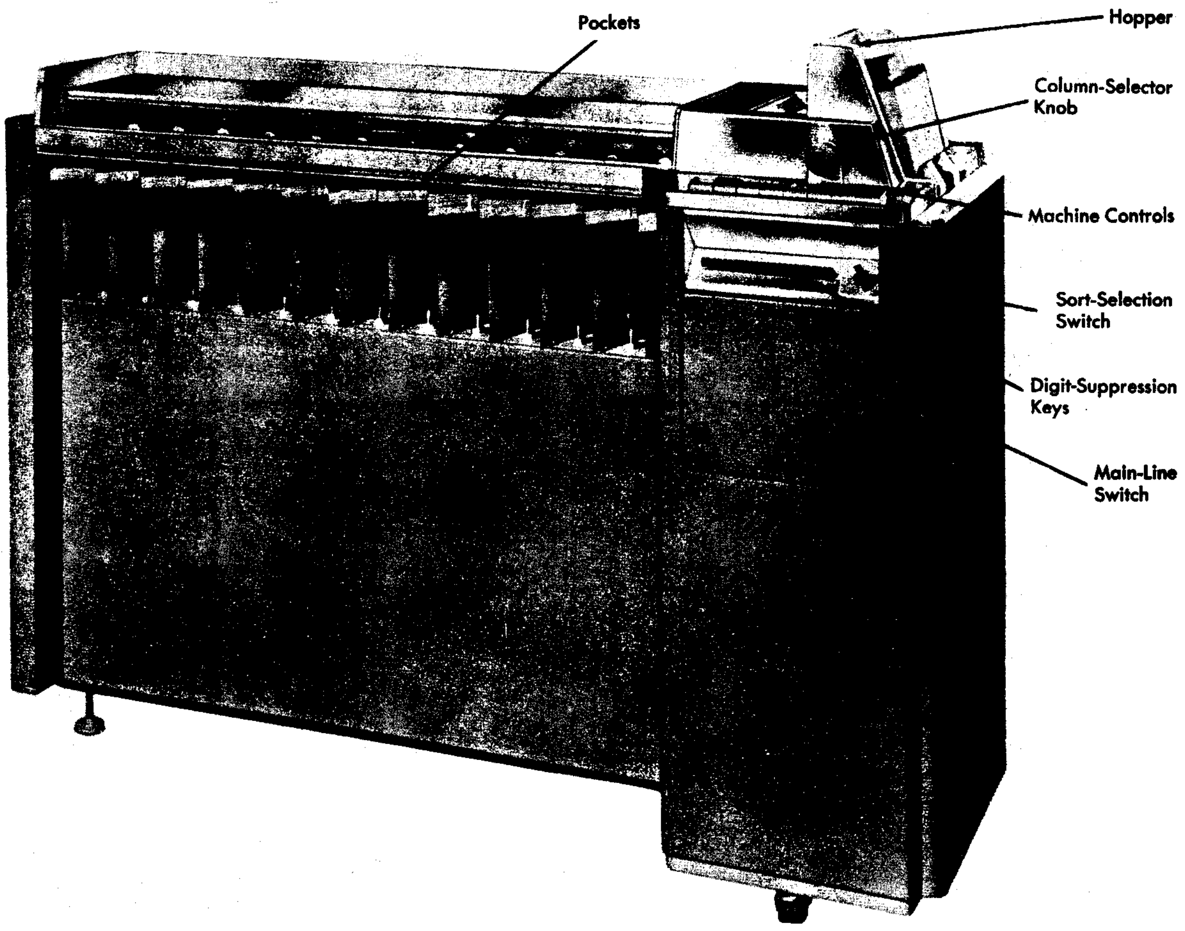


The interpreter reads information punched into a card and prints it on the card at the rate of 100 cards per minute. Up to 60 characters can be printed in a single pass through the machine. The remaining 20 characters on the card can be printed on a second pass. Printing can be positioned on the card on any one of 25 lines. This machine is not yet equipped to interpret all 360/67 code.

Operating Instructions

- Register with a clerk at the Dispatch counter in Pine Hall for use of the 557 Interpreter. The power switch is controlled from the Dispatch area. Notify the Dispatch clerk when your work is completed so that the unit can be turned off.
- Verify that the proper control board is in the machine.
- Joggle the cards into perfect alignment and place them face down in the hopper with the 12-edge inward (to the left).
- Set the printing position control (the clear plastic knob with numbers on the edge) to the desired print line. Line no. 1 is above the 12-line on the top edge of the card, line no. 2 is the 12-punch line, line no. 3 is between the 11 and 12 punch lines, etc. The odd-numbered lines (3 through 23) are between the punch lines.
- Set the "ENTRY" toggle switch at the right-hand end of the controls to the "UP" position for entry 1 (the first 60 characters), or the "DOWN" position for entry 2 (the remaining 20 characters), and push the black "START" button.
- The machine will interpret the punches in the cards which will emerge in their original order in the stacker.
- The machine will stop automatically when the final card has been interpreted, when the stacker is full, if the feed mechanism fails or if the "STOP" button is pushed.
- A special control board is provided for interpreting binary cards.

3. 83 Sorter



The sorter arranges punched cards in either alphabetical or numerical sequence, sorting a single column at a time.

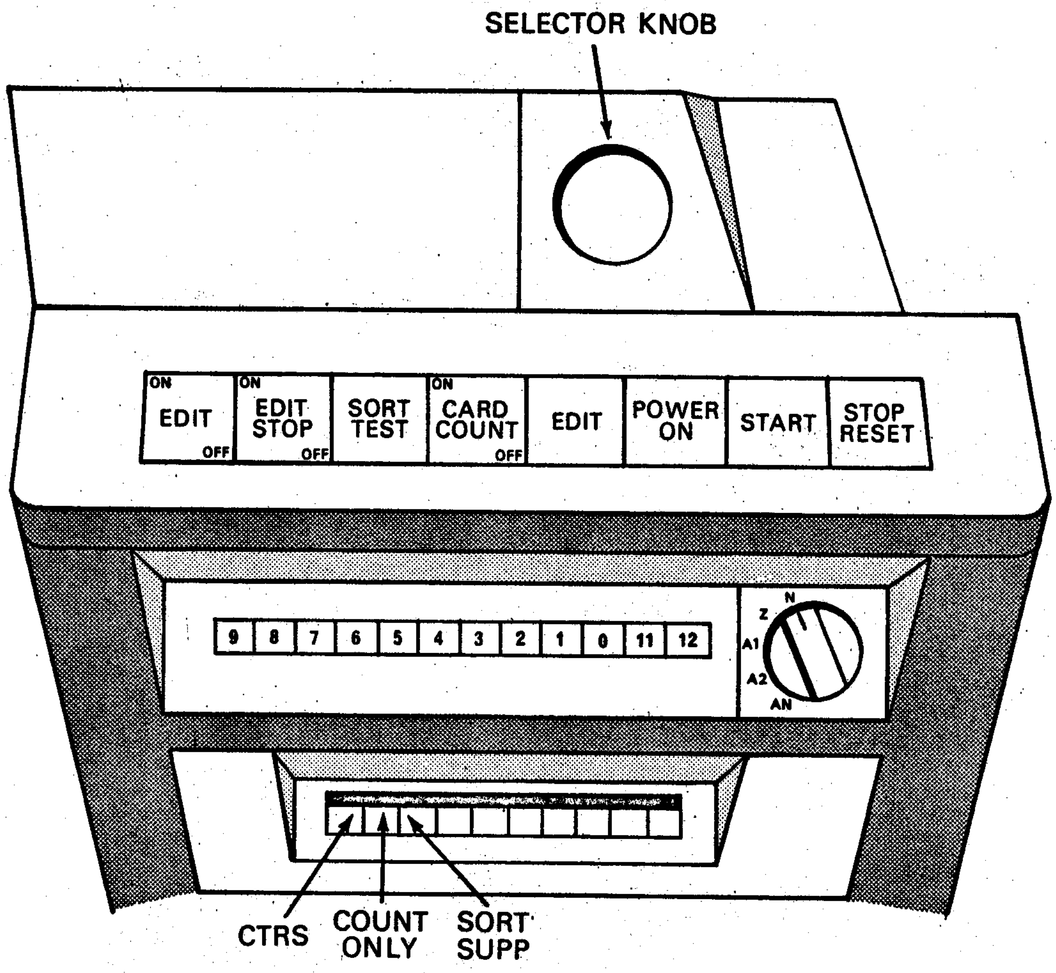
Operating Instructions

- o Register with a clerk at the Dispatch counter in Pine Hall for use of the sorter. The power switch is controlled from the Dispatch area.
- o After a 60-second warm-up period, press the "START" key to clear the machine of any cards left by the previous user.
- o Joggle the cards into perfect alignment and place them in the hopper at the right-hand end of the machine, then put the card weight on top of the stack. The cards must be face down with the 9-edge toward the throat (left).
- o Press the "START" key until the machine starts feeding cards from

the bottom of the stack. Each card passes under the brush head, which determines which of the 13 stacker pockets will accept it. There is a pocket for each punch position in the card, and a reject pocket for cards without a valid punch in the column being sorted.

- o If a jam occurs, ask the Dispatch clerk for assistance -- do not attempt to clear the jam yourself.
- o The machine will stop when a pocket is full, when the hopper is empty, when the cover over the brush is raised, or when the "STOP" key is pressed.
- o Notify the Dispatch clerk when your work is completed so that the unit can be turned off.

Operating Features



- Digit-Suppression Keys - The twelve digit-suppression keys correspond to the twelve punching positions on an IBM card. To select from a file all cards that have any specified punch or punches in one column, suppress sorting of these punches by pressing the corresponding digit-suppression keys. When pressed, the keys automatically latch down. To unlatch or reset the keys, run a fingertip along the bottom edge of the keys.

- Sort-Selection Switch - A 5-position rotating switch determines the sorting pattern: Numerical, Zone, Alphabetic-Sort 1, Alphabetic-Sort 2, or Alpha-Numerical. Figure 2.2a on page 2-12b shows the sorting pattern for each setting of the sort-selection switch. Blanks fall into the reject pocket on every setting.

Numerical (N): Cards are sorted on the first punch that is read. Double punches fall into the reject pocket as errors if the edit switch or edit-stop switch is on.

Zone (Z): Cards sort on zone (0, 11, 12) punches only. Cards without a zone punch fall into the reject pocket. Any card with more than one zone punch in the column being sorted falls into the reject pocket as an error if the edit switch or edit-stop switch is on.

Alphabetic-Sort 1 (A1): Cards punched with a digit and a 12-zone (letters A through I) sort on the digit punches (1 through 9). Cards with an 11-zone punch fall into the 11-pocket. Cards with a zero punch fall into the zero pocket. Cards with only a digit punch and cards with only a 12-zone punch fall into the reject pocket. If the edit switch or edit-stop switch is on, cards with multiple-digit punches or multiple-zone punches fall into the reject pocket as errors.

Alphabetic-Sort 2 (A2): Cards punched with an 11-zone and a digit (J through R) or a zero-zone and a digit (S through Z) sort on the digit punches. Cards with a zero or 11-punch only, cards with a digit punch only, and cards punched with letters A through I fall into the reject pocket. The error condition is the same as A1.

Alpha-Numerical (AN): Cards with a digit punch (0-9) but no zone punch fall into their respective digit pockets. Cards with an 11-zone punch fall into the 11-pocket; cards with a 12-zone punch fall into the 12-pocket; cards with a zero-zone punch fall into the reject pocket. The error condition is the same as A1.

- Edit Switch - With this switch on, errors fall into the reject pocket without stopping card-feeding.
- Edit-Stop Switch - With this switch on, errors fall into the reject

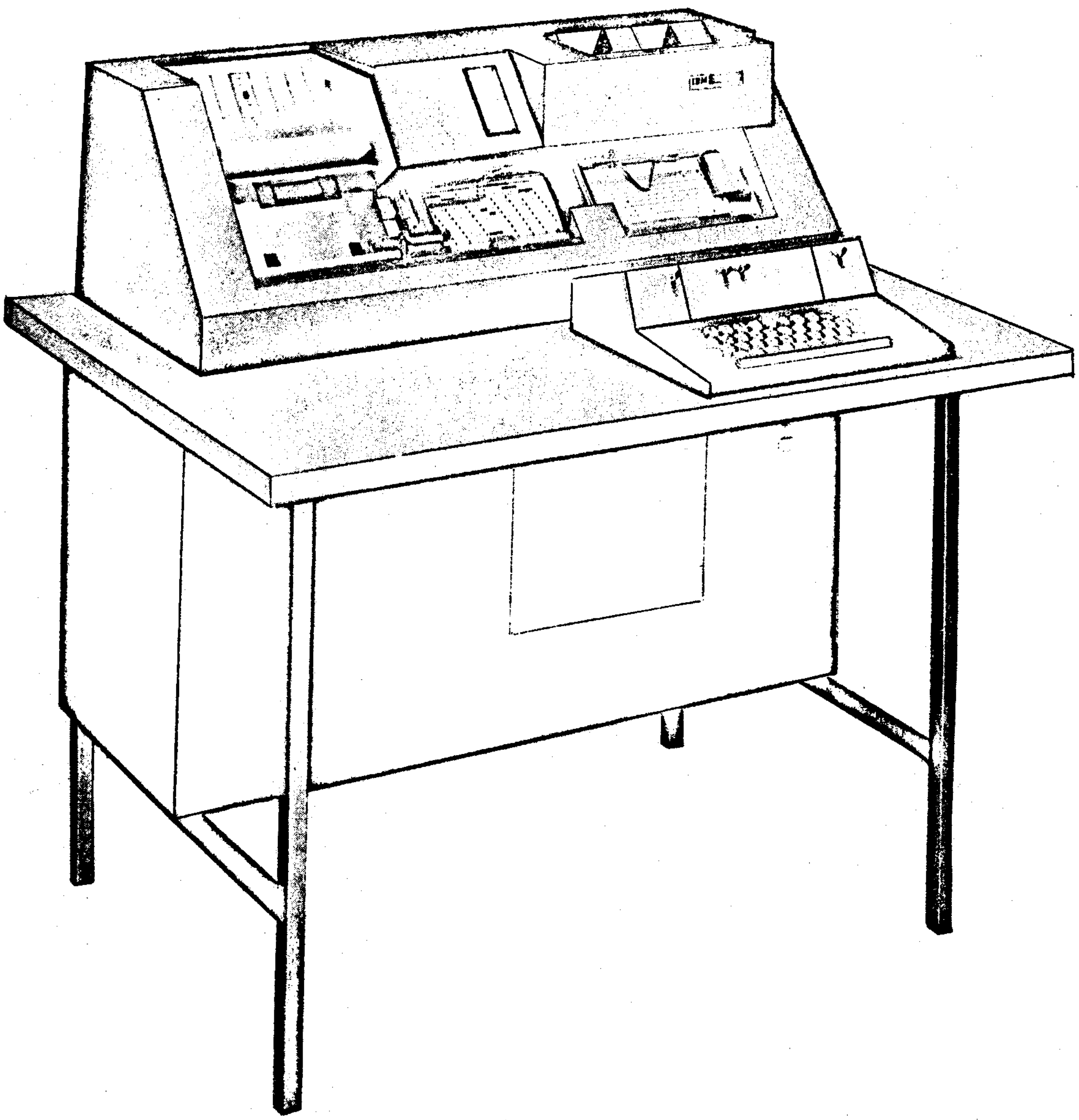
pocket, the error light comes on, and card-feeding stops. To reset the error circuits when the machine has stopped and the error light is on, press the stop-reset key.

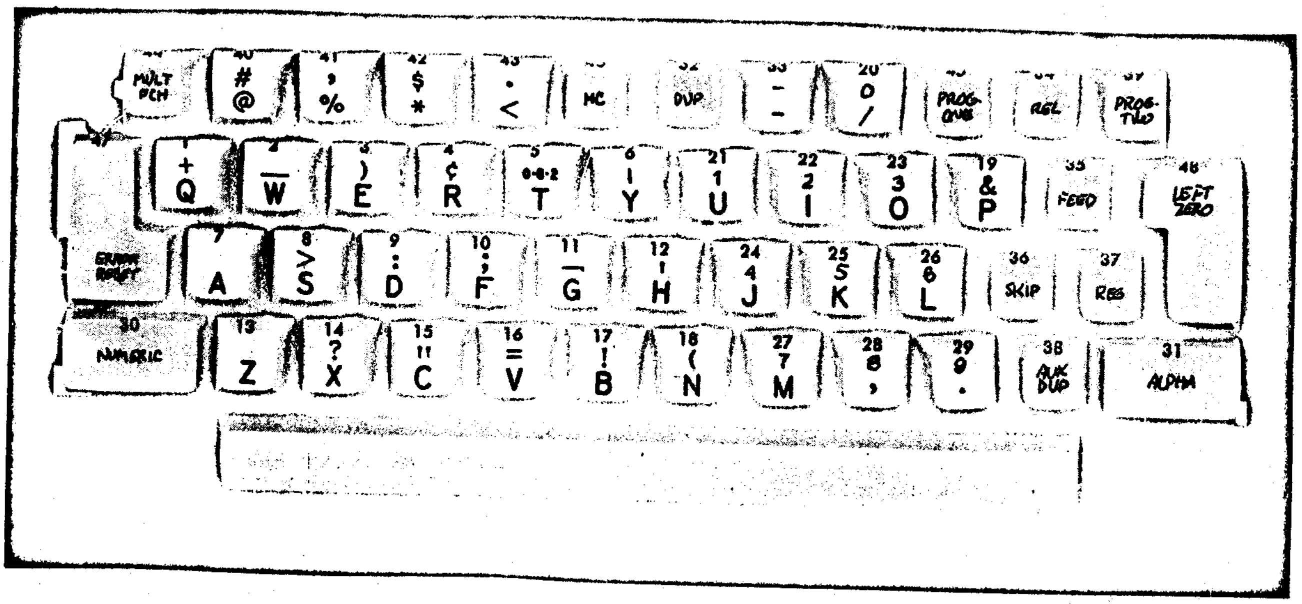
- o Sort-Test Switch - To check machine timing, the customer engineer sets this switch to "TEST". Set the switch to "SORT" for sorting operations.
- o Edit Light - When the edit-stop switch is on and the machine senses an error, this light comes on. It also comes on when the test-sort switch is set at "TEST" and the brush is reading a punch in the card.
- o Power-on Light - This light glows when the main-line switch is on and the machine is ready to operate.
- o Stop-Reset Key - Press this key to stop card-feeding or to reset the error circuits when the sorter has stopped with the error light on.

SORT SELECTION SWITCH SETTING	POCKETS												REJECTS REGARDLESS OF EDIT	ERRORS (When Edit or Edit-Stop is ON)
	9	8	7	6	5	4	3	2	1	0	11	12		
Numerical (N)	9	8	7	6	5	4	3	2	1	0	11	12	Blanks	Multiple-punched cards (incl. letters)
Zone (Z)										0	11	12	Any card without a zone punch	Any card with more than one zone punch
Alpha-1 (A-1)	I	H	G	F	E	D	C	B	A	0 S-Z	11 J-R		Blanks and cards with a 12-zone punch but no digit punch. Digits 1 to 9.	Any card with more than one zone punch or with more than one digit punch
Alpha-2 (A-2)	R,Z	Q,Y	P,X	O,W	N,V	M,U	L,T	K,S	J 0-1				Cards with 0 or 11-zone only. Blanks. Letters A to I, and 12-zone spec. char. Digits 1 to 9.	Same as A-1
Alpha-Numerical (A-N)	9	8	7	6	5	4	3	2	1	0 (digit)	11 J-R	12 A-I	Blanks, 0-zone (S-Z)	Same as A-1

Figure 2.2a

4. <u>029 Keypunch</u>





Key Number	ALPHABETIC		NUMERIC	
	Card Code	Graphic	Card Code	Graphic
1	11-8	Q	12-8-6	+
2	0-6	W	0-8-5	_
3	12-5	E	11-8-5	)
4	11-9	R	12-8-2	¢
5	0-3	T	0-8-2	0-8-2
6	0-8	Y	12-8-7	|
7	12-1	A	none	none
8	0-2	S	0-8-6	>
9	12-4	D	8-2	:
10	12-6	F	11-8-6	;
11	12-7	G	11-8-7	¬
12	12-8	H	8-5	'
13	0-9	Z	none	none
14	0-7	X	0-8-7	?
15	12-3	C	8-7	"
16	0-5	V	8-6	=
17	12-2	B	11-8-2	!
18	11-5	N	12-8-5	(
19	11-7	P	12	&
20	0-1	/	0	0
21	0-4	U	1	1
22	12-9	I	2	2
23	11-6	O	3	3
24	11-1	J	4	4
25	11-2	K	5	5
26	11-3	L	6	6
27	11-4	M	7	7
28	0-8-3	,	8	8
29	12-8-3	.	9	9
33	11	-	11	-
40	8-4	@	8-3	#
41	0-8-4	%	0-8-3	,
42	11-8-4	*	11-8-3	$
43	12-8-4	<	12-8-3	.

Key Graphics and Punched-Hole Codes

o Cards can be punched under "manual" control or " program" control. "Program" controlled punching is advisable when preparing a large number of cards all with a similar format. "Manual" punching is simple and is recommended when a few or randomly formatted cards are to be prepared.

o Manual Control Punching:

a) Put the supply of cards to be punched into the hopper on the upper right-hand side of the machine.

b) Turn the three switches "AUTO FEED","AUTO SKIP","AUTO DUP", and "PRINT" to the "ON" position.

c) Press the "FEED" key at the right of the keyboard twice. This will bring down two cards. The first card is ready for punching.

d) If you punch through col. 80 the machine will automatically eject the card punched, position the next card for punching and feed another card.

e) If you want to eject a card before reaching col. 80, manually press the "REL" key.

f) When punching a very few cards, you can insert cards into the punch station at the right of the machine. Press "REG" and begin punching.

g) To duplicate a card, put a blank card in the punch station and the source card in the read station (to the left) and then press "REG". Next, hold down the "DUP" key for continuous duplicating, or use the "DUP" key to duplicate column by column. This procedure is commonly used to correct punching errors.

o Program Controlled Punching

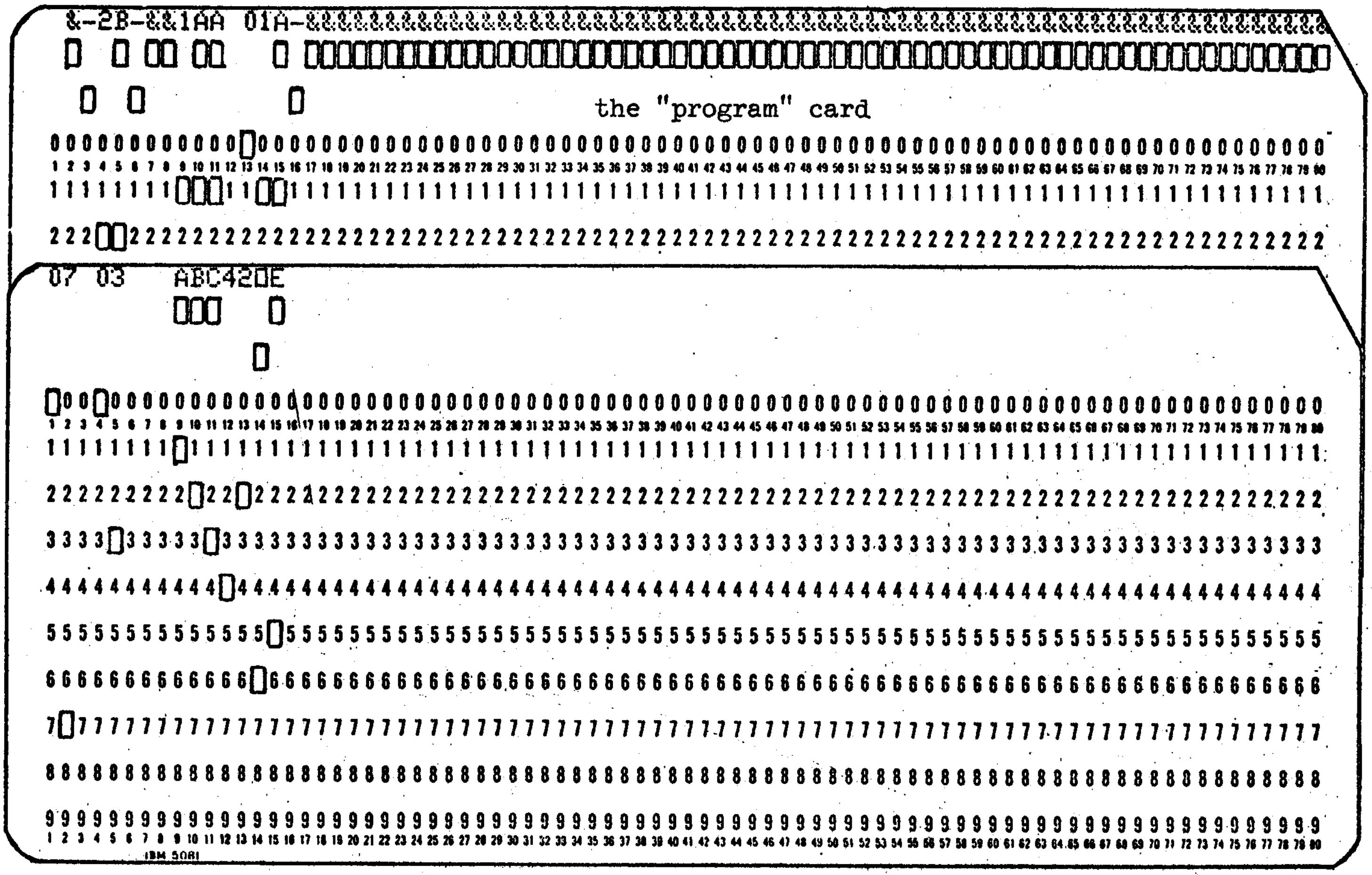
a) Preparing a "program" card. In the program control mode, the program card controls the format of the cards and the characters (alphabetic or numeric) to be punched.

1) Program control symbols

1	This punch allows punching of alphabetic characters
b	A blank column allows punching of numeric characters
0	This symbol causes duplicating from the column at the read station to the column at the punch station
-	This code causes the card to skip
&	A & symbol in each column in the field, except the first, defines a field.

Revised May 1968

2) This example of a program card shows all common combinations of codes and their resultant products.


&-2B-&&1AA 01A-&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
the "program" card
07 03 ABC420E
IBM 5081

- o The keypunch is designed for punching and duplicating only those characters contained on the keyboard. Heavily coded cards (e.g., cards with more than 3 punches per column) cannot be duplicated on the machines.

- o Please remember to consider the other users and clean up the machine before leaving it. Dispose of cards in a nearby "CARD disposal can."

- o See the Keypunch Supervisor for assistance with keypunch machines and to report failures.

2.3 Operational Procedures

The computers, related equipment, and Operations Group are located in Pine Hall. These facilities are open and staffed 24 hours a day, seven days a week, including holidays. The Operations Group is responsible for the operation and maintenance of the equipment and also offers many related services and facilities. The Group is headed by the Operations Manager who is located in Room 186, Pine Hall, extension 4873. The Operation Supervisor is responsible for smooth operations and workflow. He is located in 188-A Pine Hall, extension 4394. This office is continuously staffed, and all special Operations requests or problems should be brought here.

At the Dispatch counter, Operations personnel handle input and output to the computers, assign lockers and output bins, and generally assist with the user's operations needs. Users with programming questions should go to the Consultants Office, Room 185, Pine Hall.

The following sections describe details of the facilities and services provided for the user. These sections are intended to give the user sufficient information to use these services and facilities. Additional information and assistance can always be obtained by contacting the person in charge or the personnel in Dispatch.

The Computation Center invites your comments and suggestions regarding the facilities and services offered. Your comments will be studied and a reply will be sent through the User Services Office. Forms for submitting suggestions (see Figure 2.3) are available at the Dispatch counter and the Systems Documentation Office.

COMPUTATION CENTER

The Computation Center invites your comments and suggestions, general and specific, regarding the facilities and services offered. Your comments will be studied and a reply will be sent through the User Services office.

USER COMMENT FORM

NAME ____________________

CHARGE NO. __________ BIN NO. __________

MAIL ADDRESS:

DATE __________

SUBJECT ....

Figure 2.3

Revised February 1968

2.3.1 Service Schedule

WEEKDAYS	Monday thru Thursday	WYLBUR Service Available	Self-Service Card Readers in use
00:01 to 04:00	OVERNIGHT SERVICE BLOCK	No	No
04:01 to 08:00	Hardware maintenance period.	No	No
08:01 to 09:00	Machine start-up period.	No	No
09:01 to 17:00	PRIME SERVICE BLOCK.	Yes	Yes
17:01 to 20:00	Software maintenance period.	No	No
20:01 to 00:00	EVENING SERVICE BLOCK.	Yes	Yes
	Friday		
00:01 to 04:00	OVERNIGHT SERVICE BLOCK.	No	No
04:01 to 08:00	Hardware maintenance period.	No	No
08:01 to 09:00	Machine start-up period.	No	No
09:01 to 18:00	PRIME SERVICE BLOCK.	Yes	Yes
18:01 to 00:00	Engineering change period.	No	No
WEEKENDS	Saturday		
00:01 to 06:00	Engineering change period.	No	No (Note 1)
06:01 to 08:00	Machine start-up period.	No	No
08:01 to 12:00	PRIME SERVICE BLOCK.	Yes	Yes
12:01 to 18:00	Software maintenance period.	No	No (Note 2)
18:01 to 00:00	EVENING SERVICE BLOCK.	Yes	Yes
	Sunday		
00:01 to 06:00	OVERNIGHT SERVICE BLOCK.	No	No
06:01 to 08:00	Machine start-up period.	No	No
08:01 to 12:00	PRIME SERVICE BLOCK.	Yes	Yes
12:01 to 18:00	Software maintenance period.	No	No (Note 2)
18:01 to 00:00	EVENING SERVICE BLOCK.	Yes	Yes

Note 1: There is no Friday overnight scheduled for Saturday morning. If the machine is not required for the stated purpose a block will be scheduled. Notice will be made in Pine Hall and on the recorded message.

Note 2: There is no service block scheduled for Saturday or Sunday afternoon. If the machine is not required for the stated purpose, a service block will be scheduled. Notice will be made in Pine Hall and on the recorded message.

Special service scheduling for such things as weekend institutes should be discussed with the Associate Director, Campus Facility.

2.3.2 Job Flow

This procedure will be used during the Prime and Evening Service Blocks. Users should check any posted notices in the Dispatch area to see if there is a change in this procedure.

1. Enter your own decks:
   The user enters his job directly into the system by the two card readers provided. These jobs are queued on disk and then executed according to the current scheduling algorithm.

2. A service card is NOT required unless you have tapes or need to give special information to the operator:
   The service card should not be entered into the system.

   Note: If your job requires tapes or any other special service, a service card should be presented to the operator at the window in the card reader room before you enter your program deck.

3. Output:
   Output will be returned to your bin. Slight delays may be caused by requests for punched cards. Input card decks left for overnight runs are returned to the trays on the dispatch counter.

4. Expected turnaround time:
   This is highly dependent upon inter-arrival time at the card readers and, of course, on the number and time estimates of the jobs submitted.

5. TV Monitor of job progress:
   A TV monitor is provided to show you the progress of the system as it works on the job stream. You should observe the number assigned your job by the system. (It is started at 1 at the beginning of the block.) By comparing your number with the number of the job currently in execution, you can estimate when your job will be out.

Revised May 1968

6. WYLBUR terminal used to display system status:

The WYLBUR terminal in the 360/67 card reader room is for the convenience of persons entering their jobs into the system using the card readers. With this terminal, users may issue the following WYLBUR commands to determine system load and to locate jobs which are in the system:

COMMAND ? show status	shows the current workload on the 360 in terms of the jobs being executed and the jobs awaiting processing.
COMMAND ? show run	returns the number of jobs awaiting execution by priority.
COMMAND ? show print	returns the number of jobs awaiting print by priority.
COMMAND ? locate n	where n is the system assigned job number for your job.

The user may also use the LOCATE command to search for a job by its job name rather than its job number.

```
COMMAND ? locate fire
JOB NAME - FIRE        - NOT FOUND
```

In the above example the job name specified was not found by the system. Either the user entered the wrong name (spelling error?) or the system had completed all action for the job.

```
COMMAND ? locate newfix1
JOB NAME -NEWFIX1 - IS JOB NUMBER -    57-
JOB 0057 IS AWAITING EXEC
```

In this example the system reported back both the job number and the status of the job having the name NEWFIX1.

When the system finds more than one job with the same job name it will report the job number of all and give the status of the last (highest numbered) one.

7. Special Runs:

Requests for "Scheduled Programmer Present" runs should be directed to the Dispatch Desk or call Ext. 4392.

8. <u>Assistance with large decks (in boxes)</u>:

Users with large decks should request assistance from one of the dispatch clerks. An operator will read your deck in using one of the operator card readers. Your cards will be returned to you on the shelves in the lobby.

2.3.3 Keypunch Service

The Campus Facility maintains a Keypunch Section staffed by qualified personnel to handle all keypunching needs. The current rate for this service is listed in the Rate Schedule in Section 2.1.2 of this manual.

The Keypunch Section is located in 188 Pine Hall, extension 4395. Hours of operation are 8 a.m. to 4 p.m., Monday through Friday. Completed work is left in a central bin so that it can be picked up by the user at anytime.

The Keypunch Supervisor can provide assistance in the preparation and coding of documents to be punched. It is recommended that users who plan to use the keypunch service consult with the supervisor in the document design stage. Planning for the keypunching step in this early stage can mean less expense later.

2.3.4 Special Services

Decks can be left at the Keypunch Section for reproducing, interpreting, and other special services. The current rate for these services is listed in the Rate Schedule in Section 2.1.2 of this manual.

2.3.5 Magnetic Tape Library

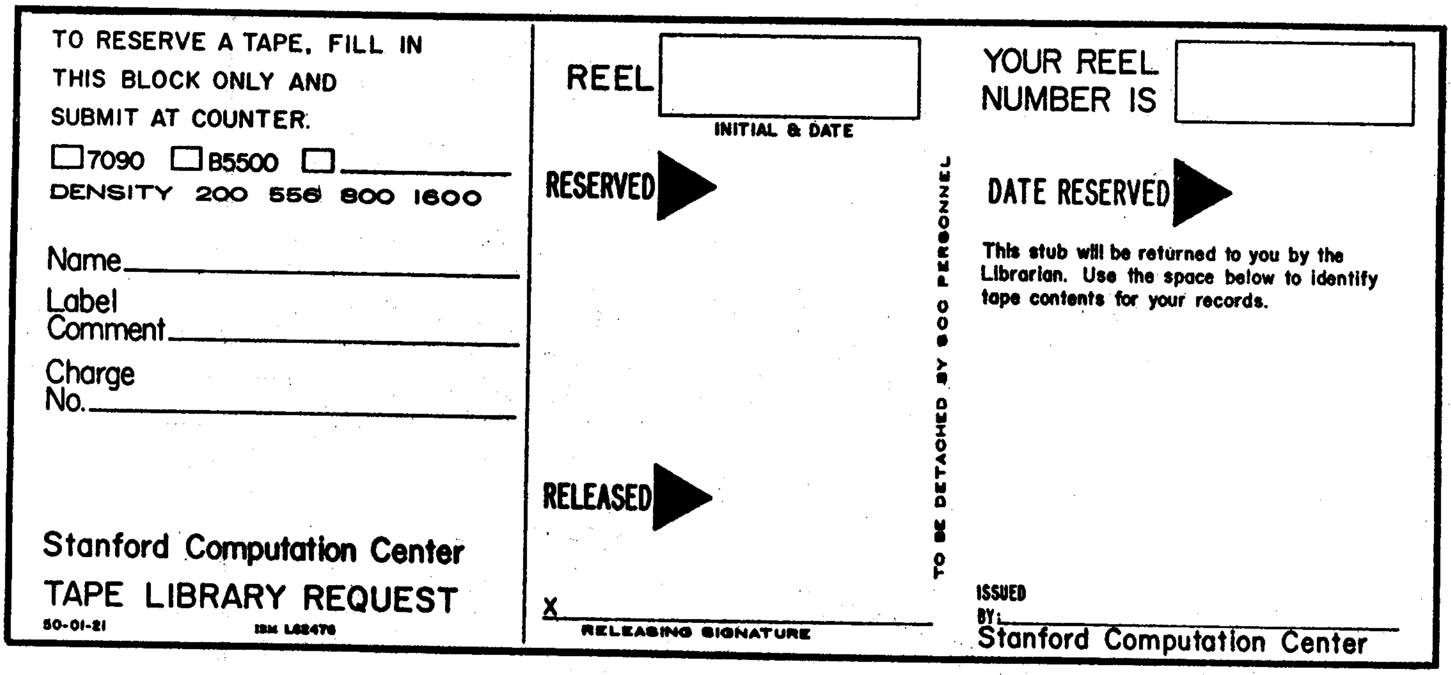
The Tape Library, located in Dispatch, Room 188, maintains a supply of tapes which may be reserved by the users. SCC tapes may be reserved by any Center user with a valid charge number.

The user can reserve an SCC tape by filling out and submitting a Tape Request Form (Figure 2.3), which is available at the counter in Dispatch. A tape will be reserved in the user's name and the operator receiving the tape request will record the tape number on the request form and return the stub to the user. When the user wishes to release an SCC tape reserved for him, he may either sign the request form on file in Dispatch, or sign his name on the request form stub and send it to the Tape Librarian.

Tapes owned by the user can be entered in the Magnetic Tape Library by submitting the tape to the Tape Librarian and filling out a Tape Request form. A number will be assigned to the user's tape and entered on the portion of the request form which he retains. User-owned tapes will only be released to the user in person.

A small charge is made for the storage and handling of both user-owned and SCC reserved tapes in the Tape Library.

SCC tapes are cleaned periodically according to a standard schedule; in addition, it will be cleaned and certified at the user's request. User-owned tape will be cleaned and certified at the user's request. The Tape Librarian can be reached at extension 4392.



TO RESERVE A TAPE, FILL IN THIS BLOCK ONLY AND SUBMIT AT COUNTER.

☐7090 ☐B5500 ☐ ________

DENSITY 200 556 800 1600

Name ________

Label Comment ________

Charge No. ________

Stanford Computation Center

TAPE LIBRARY REQUEST

50-01-21 IBM L82476

REEL ☐

INITIAL & DATE

RESERVED ▶

RELEASED ▶

X ________

RELEASING SIGNATURE

TO BE DETACHED BY SCC PERSONNEL

YOUR REEL NUMBER IS ☐

DATE RESERVED ▶

This stub will be returned to you by the Librarian. Use the space below to identify tape contents for your records.

ISSUED BY: ________

Stanford Computation Center

Figure 2.4

2.4 User Services

The User Services Group is comprised of full-time professional programmers who are responsible for providing assistance to the user community. The specific areas covered by this group are consulting, education, documentation, and programming assistance.

2.4.1 Consulting Services

The consulting service is intended to help the user get his program running in the shortest possible time. The consultant is able to answer any question regarding job setup, job control language, FORTRAN, PL/1, the interpretation of diagnostics and dumps, tape handling, the creation and deletion of temporary and permanent data sets, the use of library facilities, and requests for time adjustments due to operator, system, or hardware failure. In short, consultants should be able to answer, or find the answer, to any question regarding hardware and programming systems supported by the Campus Facility.

While the consultant can often indicate where a problem may lie, he is not expected to write, debug, or rewrite programs. The consultants are specifically prohibited from advising the user as to which technique or method should be used to solve a problem. After the user has made a decision, however, the consultant may then indicate which available routines perform the necessary calculations. The user is advised to seek directions early in the developmental stages of a research project. Assistance in designing questionnaires and coding sheets is available.

The Campus Facility Consultants are on duty Monday through Friday, 9:00 a.m. - 12 noon and 1;00 p.m. - 5:00 p.m., in Room 185, Pine Hall.

2.4.2 Educational Activities

The current educational effort centers around FORTRAN IV, described in the publication IBM System/360 FORTRAN IV Language, Form No. C28-6515. Several short courses are offered each quarter. These courses assume no previous programming experience, and provide the student an opportunity to learn enough FORTRAN to write simple programs and run them on the 360/67. The student is also introduced to Job Control Language to enable him to use the standard catalogued procedures described in the IBM System/360 Operating System FORTRAN IV (H) Programmer's Guide, Form No. C28-6602.

Courses in Job Control Language (described in the publication IBM System/360 Operating System Job Control Language, Form No. C28-6539) are offered on a regular basis. These courses cover basic Job Control Language and give the student sufficient information about the 360/67 system to enable him to write his own job control statements for special programming application.

Assembly Language Programming (described in the publication IBM System/360 Operating System Assembler Language, Form No. C28-6514) is taught at least once each quarter. Although no previous programming experience is required, a working knowledge of Job Control Language is necessary. Familiarity with some programming language is helpful.

Siminars are scheduled on Wednesday of each week, 3:30 - 5:00 p.m., in Room 111, Polya Hall. Special topics such as the use of plotting routimes, library facilities, data-set handling, conversion of seven-track tapes, etc., are covered.

The course and Seminar schedule is published in the Campus Facility Bulletin. This schedule is also available in Room 185, Pine Hall, and the User Services Office, Room 156, Polya Hall.

2.4.3 Documentation

Documentation for all of the services supported by the Campus Facility is available from the Systems Documentation Office, Room 185, Pine Hall. Abstracts for the subroutines in the Stanford Extrinsic Program Library, the IBM documents for the System/360, and the Campus Facility Users Manual are among the available materials. Reference copies of these materials are located in the Computer Science Library and in the Study Room located off Pine Hall's Dispatch area.

New facilities, changes or problems in existing facilities, short courses and seminar schedules, and policy announcements are communicated to the user community through the Bulletin. Users are urged to subscribe to this publication by submitting their name and address to the Systems Documentation Office, Room 185, Pine Hall, Ext. 4877.

The Users Manual is supplemented by the Bulletin and is revised as needed. Revisions are mailed to holders of the Users Manual.

2.4.4 Programming Services

Users are encouraged to write their own programs, and to use the available library facilities for solving their problems. However, the User Services Group may, on occasion, develop programs to solve an individual user's particular problem. The user will be charged for the machine and programmer time required for this type of development. Users desiring this service are asked to discuss their project with the supervisor of the User Services Group, Room 156, Polya Hall.

3. COMMUNICATION WITH THE COMPUTER

3.1 Operating System (OS)

The IBM System/360 Operating System introduces programs to the computing system, initiates their execution, and provides them with all the resources and services necessary for them to do their work. To be effective, the operating system must be general enough to accommodate a variety of applications on a wide range of hardware configurations. It is, therefore, made up of a general library of programs that can be tailored to meet many requirements. The user can select those portions that he needs, add his own procedures to them, and update his procedures as his needs change.

For illustration purposes, the programs and routines that compose the operating system are classified as a control program and processing programs. The three main functions of the control program are to accept and schedule jobs in a continuous flow (job management); supervise on either a sequential or parallel basis each unit of work to be done (task management); and simplify retrieval of all data, regardless of the way it is organized and stored (data management). The processing programs consist of language translators (such as the FØRTRAN compiler), service programs (such as the Linkage Editor), the problem programs (such as User programs). The processing programs are used to define the work that the computing system is to do and to simplify program preparation. (Figure 3.1 shows the operating system elements.)

The most important feature of the operating system is its unity. This unity establishes a direct line of communication between the user and the operating system and, within the system, between the control program and the processing programs. The net result of this effective communication is a reduction in the time from submitting a problem to receiving a solution.

Revised May 1968

Operating System Elements

Control Program Elements		
Job Management	Data Management	Task Management
Languages	Service Programs	Application Programs
ALGØL Assembler CØBØL FØRTRAN PL/1 PLA LISP SLIP ALGØL W WATFØR	Independent Utilities Data Set Utilities System Utilities Linkage Editor Sort/Merge TESTRAN	User Written

Figure 3.1

3.1.1 HASP

The Houston Automatic Spooling System (HASP) is a specialized program which performs the peripheral functions normally associated with offline support computers. It has the ability of operating an essentially unlimited number of peripheral devices (i.e., card readers, punches, printers) concurrently and in conjunction with OS/360 processing. The user does not program with HASP in mind -- it is actually transparent to the user. However, the information that appears on the header and trailer sheets of your output; the accounting information gathered for your job; the assigning of a job number for your job; the checks for line, card, and time estimates; the priority system and several other related features on the 360/67 are all produced and controlled by HASP.

HASP performs these functions and operates the peripheral devices by "stealing" small amounts of CPU time. In fact, HASP only uses six minutes of processor time in an average twelve hour service block. A better understanding of HASP and its major functions can be obtained from the following description of a job (a FORTRAN job with data cards in the deck, for example) and its flow through the system.

HASP starts by reading a job from one of the card readers. At this point HASP inspects the job card for validity and records any special services requested, such as tape setup or multiple-part forms. If the job card is in proper format HASP then "spools" the deck onto a disk for storage until this job is to be executed. At execution time the job is read in from the spooling disk and submitted to the OS processor (and thus to the FORTRAN compiler, linkage editor, etc.). During this execution phase any requests by the job for the card reader (probably a request for the data), the printer, or the punch are given to HASP rather than the actual physical device. HASP spools the output (FORTRAN compiler listing, storage map, etc.) to disk as it is processed by OS and reads in the data cards entered earlier with the job. At the end of OS execution, this job is scheduled for printing and the next job is submitted to the OS processor. After the job is scheduled for printing, the job enters the punch phase. After the punch phase the job is "purged," i.e., HASP erases the entry for this job from the job table and makes room for another job to be read from the card readers and spooled to a disk.

The above example follows one job through the four major processing stages in HASP. What is not apparent from the example is that while one job is in the execution phase, other jobs are concurrently being read into the system, punched, or printed. The number of jobs in these last three stages depends on the number of peripheral devices available. Currently at the Campus Facility, different jobs can be read in from 4 card readers, punched on 2 punches, and printed on 3 printers all simultaneously <u>and</u> while still another job is being executed by OS. HASP, therefore, increases throughput by giving this job-to-job continuity and by making efficient and continuous use of the available peripheral devices.

3.2 Job Control

Job processing requires that the programmer communicate with the operating system. The language used for this communication under Operating System/360 (ØS) is the Job Control Language (JCL). JCL provides the operating system user with a means of influencing the execution of work and the allocation of system resources, in addition to transmitting control information. Because the information can be submitted on devices other than a card reader, the elements of the language are called "control statements" instead of control cards. (Users of other operating systems are familiar with the necessity of supplying special control cards to set up their jobs properly; JCL control statements serve the same purpose under ØS.)

3.2.1 Job Control Statements

There are four JCL statements that concern the user. They are:

1. The JØB statement marks the beginning of a job and precedes all other statements in the job. It identifies the user and his job.

2. The execute statement (or EXEC statement) is the first control statement in each job step -- it identifies the program to be executed or the catalogued procedure to be used.

3. The data definition statement (or DD statement) describes a data set (a file on tape, disk, or a deck of cards) and requests the allocation of input/output devices.

4. The delimiter statement (/*) marks the end of a data set in the input stream. It is used to separate the data set from subsequent control statements. It is analogous to the "end-of-file" statement in other systems.

3.2.2 JØB Statement

Format:

```
//aaaaaaaa△JØB△(bbbb,ccc,ddd.d,eeee,ffff,g,hh,s,x),'Your Name',MSGLEVEL=1
```

Where:

aaaaaaaa	= job name (1 to 8 characters, the first of which must be alphabetic). Don't use special characters in the job name.
bbbb	= your account number (1 alphabetic character followed by 3 numeric characters). Users submitting jobs under invalid account numbers will be flushed and a message 'ILLEGAL ACCOUNT NUMBER' will be printed on the 1443 printer.
ccc	= your bin number (3 numeric characters).
ddd.d	= time estimate. Up to three digits to the left of the decimal point and up to one digit to the right of the decimal point are permitted. The field is read as minutes and tenths of minutes. The following configurations are legal: 124 124. 124.0 24.3 0.3 .3
eeee	= your estimate of printed output - lines - in thousands (1 to 4 numeric characters) 5 = 5000.
ffff	= your estimate of number of cards to be punched (1 to 4 numeric characters). If this field is omitted, 100 cards is assumed.
g	= number of parts in form you want your printed output on (1 numeric character). If this field is omitted, single part paper is assumed. If a digit higher than 1 is inserted in the g field, the program will automatically be placed in hold state at <u>print</u> time. The desired part paper will be put on the printer and the output will be printed at the discretion of the operator.
hh	= number of copies of printed output desired (1 to 2 numeric characters). If this field is omitted, 1 copy is assumed.
s	= insertion of the letter s in the eighth field indicates that a tape is to be mounted. When the job is card to disked it is automatically placed in hold status. The operator mounts the tape and releases the job for execution giving consideration to efficient throughput and fast turnaround time. This feature eliminates wasteful entry into wait state while tapes are being mounted. You must continue to fill out tape mounting information on the number 1 card. Jobs which do not have s in the eighth field of the job card will be canceled by the operator.

x = An insertion of the letter X in the ninth field of the job card will cause page ejects to be supressed during the go step, thus permitting users to print across page margins.

Notes:

1. Symbol definitions:

   ' indicates an apostrophe (single quote).

   △ indicates 1 or more blanks - otherwise no blanks.

2. All control statement information must start in column 1.

3. The job name is used by both the Operating System and the 360 Operator to distinguish between jobs. It is important that no two jobs with the same name be present in the system at any one time. (Imagination in naming your jobs will help eliminate potential problems. Avoid such common job names as TEST, JØB1, RUN1, etc.)

4. The total line count will include a count of all messages from the Operating System and any language processors used in the job. This must be taken into consideration when estimating the maximum count.

5. The message ***JOB TERMINATED BECAUSE OF JOB CARD ESTIMATE EXCESSION*** appears if your job was cancelled for exceeding estimated run time, estimated line count, or estimated punch count. A message identifying the error appears on the 1443 printer when the error occurs during execution time.

6. Requests for multiple part forms will receive an additional service charge. For small print jobs, the multiple copies option provided in the JØB statement will be an acceptable alternative to multiple part forms.

7. Multiple copies are printed on one-part paper in succession until the number requested is satisfied.

8. The parameter MSGLEVEL=1 causes all job control statements to be printed as part of the user's output, including the entire contents of the catalogued procedure(s) used. This parameter

is required if the consultants are to assist you with any problems you may have in using the system.

9. The following information appears on the tail sheet of your printed output.

NNN I/O CALLS	This is the total number of I/O calls made from your partition to the drums, disks, or tapes during execution of your program.
TIME = NN:NN:NN	This is the time of day in hours, minutes, and seconds at the time of printing your job.
DATE = NN/NN/NN	This is the date on which your job was run.

Examples:

1. Example of a minimum JØB statement:

```
//FIRSTGØ △ JØB △ (Z000,999,1,1),'W. WØØDPECKER',MSGLEVEL=1
```

2. Example of a JØB statement with a 4 minute time estimate and 2 copies of output:

```
//SECØNDGØ △ JØB △ (Z000,999,4,1,,,2),'W. WØØDPECKER',MSGLEVEL=1
```

3. Example of a JØB statement with a 5 minute time estimate, 5000 lines of output, and a tape required.

```
//THIRDGØ △ JØB △ (Z000,999,5,5,,,S),'WØØDPECKER',MSGLEVEL=1
```

3.2.3 Execute Statement

Format:

$$//\text{stepname}\triangle\text{EXEC}\triangle \left\{\begin{array}{l}\text{PGM=program name}\\ \text{procedure name}\end{array}\right\} , \left\{\begin{array}{l}\text{PARM='value'}\\ \text{PARM.procstepname='value'}\end{array}\right\}$$

Where: "stepname"	=	job step identification field (optional). It has a maximum field length of 8 characters, alphabetic or numeric. The first character must be alphabetic.
"program name"	=	the name of the program to be executed.
"procedure name"	=	the name of the catalogued procedure to be used.
"procstepname"	=	stepname in a catelogued procedure.
"value"	=	the field for passing control information to the processing program (e.g., compiler option). The maximum field length is 40 characters. The field should be enclosed in single quotes since blanks and special characters are allowed.

Notes:

1. Symbol definitions:

'	indicates an apostrophe (single quote).
△	indicates 1 or more blanks - otherwise no blanks.
{...}	indicates that either of the two alternate forms inside may be used.

2. All control statement information must start in column 1.
3. There is only one EXEC statement per job step.
4. If the stepname is omitted, at least one blank must separate the // characters and the keyword EXEC.
5. The program or procedure name parameter must appear first in the operand field.

6. No blanks are allowed in the operand field.

7. The PARM parameter is optional.

8. To specify that information be passed to a step in a catalogued procedure, the keyword PARM must be qualified by the particular procedure stepname involved, i.e., PARM.procstepname. This specification overrides all the PARM values which have been catalogued in the named procedure step, if any are present. As many parameters of this form may be coded as there are steps in the catalogued procedure. (See the second example following, where parameter values are passed to the procedure steps named FØRT and LKED.)

Examples:

```
//△EXEC△FØRTHCLG

//△EXEC△FØRTHCLG,PARM.FØRT='DECK',PARM.LKED='LET,XREF'

//STEP1△EXEC△PGM=ECAP

//△EXEC△FØRTHCLG,PARM.FØRT='LIST,ØPT=2'
```

3.2.4 Data Definition Statement

General Format:

Format 1:

```
        //ddname△DD△*
```

Format 2:

```
        //ddname△DD△SYSØUT=A
```

Format 3:

$$//\text{ddname}\triangle\text{DD}\triangle\text{DSNAME=dsname,UNIT=}\begin{Bmatrix}\text{device type}\\ \text{symbolic name}\end{Bmatrix},$$

$$\text{VØLUME=}\begin{Bmatrix}\text{SER=serial number}\\ \text{REF=dsname}\end{Bmatrix},$$

```
              DCB=(data set attributes),
              LABEL=(sequence number,type),
              DISP=(status,disposition),
              SPACE=(units,(quantity,increment,directory))
```

Where:

"ddname"	DD statement identification field. Maximum field length of 8 characters, alphabetic or numeric. The first character must be alphabetic.
"*"	An asterisk (*) in the operand field specifies that an input data set immediately follows this card (e.g., input source deck to the FØRTRAN compiler).
"SYSØUT=A"	Specifies that the output data set is to be written on the system output device (i.e., printed as part of the system output stream).
"dsname"	Data set identification field. The simplest form of the data set name has a maximum of 8 characters, alphabetic or numeric. The first character must be alphabetic.

"UNIT"	This keyword parameter allows the user to specify the type on input or output unit to be used by a data set. A unit may be requested by specifying

"device type"	Unit model numbers of input/output devices in the system (e.g., UNIT=2311);
or	
"symbolic name"	Names assigned to individual units or collections of units (e.g., UNIT=DISK).

(See item 3 under "Notes, Format 3" for a list of model numbers and symbolic names.)

"VØLUME"	This keyword parameter allows the user to specify information about the volume(s) on which an input or output data set will reside. The only subparameters which concern us here are those for making specific volume requests:

"SER"	Identifies the volume(s) by its serial numbers.
"serial number"	1 - 6 characters, alphabetic or numeric, volume identification field (e.g., VØLUME=SER=CAMP31).
"REF"	Uses the same volume(s) assigned to an earlier data set.
"dsname"	Defined earlier. (e.g., VØLUME=REF=DATASET1).

"data set attributes"	The DCB keyword parameter allows the user to describe the attributes to a data set at execution time (e.g., record format, logical record length, block size, buffer requirements). This field contains the attributes coded as keyword parameters separated by commas. (e.g., DCB=(RECFM=F,BLKSIZE=80)).
"LABEL"	This provides a means of describing a label for a data set. Specifically, this parameter must be supplied for all data sets residing on magnetic tape.

	"sequence number"	A 1-4 digit number corresponding to a specific file on a reel of magnetic tape where multiple data sets ("files") reside on one reel.
	"type"	Type of label on a magnetic tape. Replace "type" with: SL standard system label, or NL no label. Note: If the sequence number field is not specified, use the format LABEL=(,type) (e.g., LABEL=(,NL)).
"DISP"	This keyword parameter is concerned with	
	"status"	The status of a data set. Replace "status" with: ØLD Existing data set. NEW Newly created data set. MØD Existing sequentially organized data set to be used for additional output. MØD provides a means by which a user can extend a data set beyond the last record previously written in the data set with automatic positioning handled by the system.
	"disposition"	This indicates what is to be done with the data set after it is processed. Replace "disposition" with: DELETE Do not keep the data set and release its space.

KEEP	Keep the data set for later use.
PASS	Pass the data set to a succeeding step in this job.

(e.g., DISP=(NEW,KEEP))

"SPACE" Allocate space for a newly created data set on a direct-access volume.

"units" Amount of space is requested in units of tracks, cylinders, or blocks. Replace "units" with:

TRK	for space in tracks.
CYL	for space in cylinders, or average block length in bytes for space in blocks.

"quantity" Amount of space desired in the units selected (e.g., SPACE=(TRK,200) for 200 tracks or SPACE=(80,1000) for 1000 blocks with an average length of 80 bytes).

Note: If only the units and quantity fields are specified, the inner level of parentheses shown in Format 3 (pg.   ) in the "SPACE" parameter can be omitted.

"increment" Amount of additional space to be allocated on the same volume in case the data set at some time exceeds the total space requested. Incrementing by this amount may take place for a maximum of fifteen (15) times. (e.g., SPACE=(TRK,(200,100)).

"directory"	Number of 256-byte blocks to be allocated to the directory if allocating space for a partitioned data set (e.g., SPACE=(TRK,(200,100,2)).

Notes:
(Those items marked with an asterisk (*) are common to all three formats.)

Format 1

1. No other parameters should be coded when an asterisk (*) is coded in the operand field.

2. The input data set whose beginning is marked by a DD card of format 1 must be terminated by a delimiter card (/*).

* 3. Omission of the ddname field causes the data set to be concatenated with the data set defined in the immediately preceding DD card. (See the Job Control Language Reference Manual for more detail.)

* 4. If the ddname is omitted, at least one blank must separate the // characters and keyword DD.

* 5. If the job step uses a catalogued procedure, the ddname must be qualified by the appropriate procedure stepname, i.e., procstepname.ddname.

Examples:

//SYSIN△DD△*	
//FØRT.SYSIN△DD△*	(Specifies that the input data set to the FØRTRAN compiler follows this card. Assumes a catalogued procedure was used and the stepname in the procedure which invoked the compiler was FØRT)

Format 2

* 1. No blanks are allowed in the operand field.

Example:

```
//SYSPRINT △ DD △ SYSØUT=A
```

<u>Format 3</u>

1. The keyword parameters may appear in any order and are separated by commas.
2. If the DSNAME=dsname parameter is omitted, the data set is temporary and exists only for the duration of the job. This is a means whereby a user may define a scratch data set.
3. Users are urged to use symbolic names when requesting a unit. Figure 3.2 is a list of the symbolic names which have been generated into the system and their corresponding unit addresses and device types. Volume serial numbers are supplied where applicable and should be used when a specific volume is desired.
4. The UNIT parameter need not be coded if the parameter VØLUME=REF is specified.
5. Refer to the Programmer's Guides for the language processors or the publication <u>IBM SYSTEM/360 Operating System, Supervisor and Data Management Macro-Instructions</u> (Form No. C28-6647) for DCB macro-instructions and operands.
6. The MØD specification in the DISP parameter is changed to NEW if <u>no volume information</u> is available for the data set at that time.
7. Subsequent DD cards that refer to a data set which has been passed via the PASS subparameter of the DISP parameter must
   (1) identify that passed data set with the DSNAME parameter,
   (2) provide unit information that is consistent with that of the original specification, if unit information is given at all, and
   (3) issue another disposition with the DISP parameter.

8. When allocating space for a data set on a direct-access volume, the user should check the capacity and organization of space on that particular volume.

9. To aid in determining the amount of space to allocate for a directory in the SPACE parameter, each directory entry occupies approximately 17 bytes.

Examples:

```
col. 1         col. 16                                                col. 72
    //DD1△DD△DSNAME=ØUTPUT1,UNIT=TAPE9,                                      X
    //                VØLUME=SER=123456,LABEL=(,SL),                         X
    //                DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),                   X
    //                DISP=(NEW,KEEP)
```

(For a data set residing on a 9-track tape to be kept.)

```
    //DD2△DD△DSNAME=ØUTPUT2,VØLUME=REF=MYDATA,                               X
    //                DCB=(RECFM=U,BLKSIZE=3200),                            X
    //                DISP=(NEW,PASS),                                       X
    //                SPACE=(TRK,(200,10))
```

(For a data set residing on the same volume as a previously defined data set, the volume being a direct-access device. The data set is to be passed to a subsequent jobstep.)

Devices	Symbolic Name	Device Type	Unit Number	Volume Serial Number
Card punch	SYSCP	2540-2	00D, 01D	
7-track tapes (2 drives)	TAPE TAPE7	2400-1	0C0, 0C1	
9-track tapes (2 drives)	TAPE TAPE9	2400	0C2, 0C3	
Card reader		2540-1	00C, 01C	
Disk Storage	SYSDA DISK	2314	330-337, 390-397	SYS04-SYS07, SYS12-SYS17
Printer		1403	00E, 01E	

Figure 3.2

3.2.5 Delimiter Statement

Format:

```
            /*    (blank)
```

Notes:

1. The delimiter statement must be coded in columns 1 and 2.
2. A delimiter statement must be used to mark the end of data which has been included in the input stream so as to separate the data from subsequent job control cards. (Data refers to that which is passed to a language processor -- i.e., a source deck; that which is passed to the linkage editor -- i.e., an object deck; and that which is read as input at execution time.)
3. A delimiter statement must be the last statement of the user's deck.

3.2.6 Continuation Cards

The information of a control statement must be contained in columns 1 through 71 of a card. If this information exceeds 71 columns, continuation cards may be coded according to the following rules:

1. Complete a parameter, including the comma that follows it, at or before column 71.
2. Code any nonblank character in column 72.
3. Code the // characters in columns 1 and 2 of the following card.
4. Continue the control information beginning in column 16. Columns 3-15 must be blank.

Note: For further information on control statements beyond that which has been covered in the preceding pages, refer to the publication IBM System/360 Operating System Job Control Language, Form No. C28-6539.

3.2.7 Special Control Cards

3.2.7.1 /*LIST

Use of the /*LIST card allows the user to take advantage of the fast listing service. To use this service, place a /*LIST card in front of your job card. This will cause the deck to be listed -- including all JCL cards. The listing will be bracketed by standard HASP header and trailer sheets.

The "setup" and "suppress page" parameters on the job card are ignored. However, the "number part forms" and "number copies" parameters are honored by the listing service. Users are reminded to avoid using job cards with "forms" and "copies" parameters unless multiple copies of the listing are desired.

NOTE: Users may also obtain listings of their decks when using utility programs for character conversion and reproducing card decks. These utilities are described in Section 3.4, pg. 3-33a.

3.2.7.2 /*RUN

The /*RUN card is used to show priority for both execution and printing for jobs submitted from the card readers. A description of the use of this card is given on pg. 2-2b in Section 2.1.2.

3.3 Catalogued Procedures

Introduction

Jobs submitted for processing may be considerably simplified through the use of catalogued procedures. Frequently used job control statement sequences are prepared and placed in a library called a procedure library, SYS1.PRØCLIB. Each catalogued procedure placed in this system library becomes a member of that library and so has a member name and possibly one or more alias names. A catalogued procedure can be retrieved by using its member name or one of its alias names in the operand field of an EXEC statement, rather than naming a program. The entire catalogued procedure is substituted in place of the EXEC statement that named it.

Following is a summary of the catalogued procedures in the Campus Facility System by name and function.

Catalogued Procedures for the FØRTRAN, PL/1 and ASSEMBLER Processors

Function	FØRTRAN H	FØRTRAN G	PL/1	ASSEMBLER
Compile only	FØRTHC	FØRTGC	PL1LFC	ASMFC
Compile only, with object deck punched	-----	-----	PL1DFC	-----
Compile and link edit	FØRTHCL	FØRTGCL	PL1LFCL	ASMFCL
Compile, link edit and execute	FØRTHCLG	FØRTGCLG	PL1LFCLG	ASMFCLG
Link edit only	LKED	LKED	LKED	LKED
Link edit and execute	FØRTHLG	FØRTGLG	PL1LFLG	LKEDG

A brief description of each of the above catalogued procedures follows. Member names and alias names, processor and linkage editor options, and procedure stepnames are given. The catalogued options listed with each procedure are those which differ from the standard options defined for each processor. A brief description of the standard (or default) options is given at the beginning of the sections for each processor. (A description of the linkage editor options indicated in the LKED steps can be found in section 3.3.6.) If no specific reference is made to a particular option in a catalogued procedure, the standard (or default) option applies for that procedure. The user must not override the LØAD option under any circumstances when using a catalogued procedure which contains a linkage editor step.

If you override the catalogued procedure with a PARM list of your own, the new PARM list completely replaces the one in the catalogued procedure. Unspecified parameters are given the standard (or default) option. For example, if you override the catalogued procedure for FØRTHCLG with a PARM list specifying DECK (i.e., //△EXEC△FØRTHCLG,PARM.FØRT='DECK'), then the catalogued option for that procedure, which is MAP, will be defaulted to the standard option, which is NØMAP. Therefore, if you want to keep the catalogued options and you use a PARM list you must repeat the catalogued options in the PARM list.
(i.e., // △ EXEC △ FØRTHCLG,PARM.FØRT='DECK,MAP')

Copies of the current Catalogued Procedure Library listing are located in the Consulting Office and in the Study Room adjacent to the Dispatch Lobby in Pine Hall. Users are encouraged to refer to those listings whenever questions arise relating to the contents of a catalogued procedure. (For those who may wish to have their own copy of the PROCLIB listing, a deck set up is given in Section 3.4, 'Utilities'.)

3.3.1 FØRTRAN H Catalogued Procedures

The standard (or default) options which have been generated for the FØRTRAN H-level compiler are:

ØPT=00	No optimization of the object code produced by the compiler.
NAME=MAIN	Name assigned to the main program module by the compiler.
LINECNT=57	Number of lines per page for a source listing.
NØLIST	Object code produced by the compiler is not printed.
EBCDIC	Indicates that the source deck is punched in EBCD (029) code.
SØURCE	Source code is printed.
NØDECK	No object deck is punched for the compiled program.
NØMAP	No tables are printed of names and labels, neither those appearing in the source deck nor the compiler-generated ones.
LØAD	An object module is generated by the compiler which may be used as input to the linkage editor.
NØLD	The object code generated by the compiler does <u>not</u> contain an identifier for the compiler-assigned internal statement number associated with each function reference or CALL statement.
NØEDIT	No structured listing of the source code is printed.

The stepnames are the same throughout all the FØRTRAN H catalogued procedures.

stepname	step function
FØRT	compilation step
LKED	linkage editor step
GØ	execution step

The FØRTRAN Subroutine Library, SYS1.FØRTLIB, and the SCC Extrinsic Library, SYS2.XTRINSIC, have been defined in the FØRTHCL, FØRTHCLG, and FØRTHLG procedures as automatic call libraries. The libraries will be searched in an attempt to resolve any unresolved symbols.

1. Compile only

FØRTHC	Member name
FTNH	Alias name

Purpose: A one-step procedure (FØRT) to compile FØRTRAN source code for diagnostic messages for source code debugging.

Catalogued options:

FØRT step	MAP

2. Compile and link edit

FØRTHCL	Member name
FTNHLINK	Alias name

Purpose: A two-step procedure (FØRT,LKED) to compile source code and link edit the object modules generated by the compiler.

Catalogued options:

FØRT step	MAP
LKED step	LET,LIST,MAP

3. Compile, link edit, and execute

FØRTHCLG	Member name
FØRTRAN	Alias name

Purpose: A three-step procedure (FØRT,LKED,GØ) to compile FØRTRAN source code, link edit the object modules generated by the compiler, and execute the resulting load module.

Catalogued options:

FØRT step	MAP
LKED step	LET,LIST,MAP

4. Link edit and execute

FØRTHLG	Member name

Purpose: A two-step procedure (LKED,GØ) to link edit previously generated modules and execute the resulting load module.

Catalogued options:

LKED step	LET,LIST,MAP

3.3.2 FØRTRAN G Catalogued Procedures

The standard options which have been generated for the FØRTRAN G-level compiler are identical to those for the FØRTRAN H-level compiler with the exception that the ØPT, ID, and EDIT options are not available in FØRTRAN G.

The stepnames are the same throughout all the FØRTRAN G catalogued procedures.

stepnames	step function
FØRT	compilation step
LKED	linkage editor step
GØ	execution step

The FØRTRAN Subroutine Library, SYS1.FØRTLIB, and the SCC Extrinsic Library, SYS2.XTRINSIC, have been defined in the FØRTGCL, FØRTGCLG, and FØRTGLG procedures as automatic call libraries. The libraries will be searched in an attempt to resolve any unresolved symbols.

1. Compile only

FØRTGC	Member name
FTNG	Alias name

Purpose: A one-step procedure (FØRT) to compile FØRTRAN source code for diagnostic messages for source code debugging.

There are no catalogued options for this procedure.

2. Compile and link edit

FØRTGCL	Member name
FTNGLINK	Alias name

Purpose: A two-step procedure (FØRT,LKED) to compile FØRTRAN source code and link edit the object modules generated by the compiler.

Catalogued options:

LKED step	LET,LIST,MAP

3. Compile, link edit, and execute

FØRTGCLG	Member name
FTNGGØ	Alias name

Purpose: A three-step procedure (FØRT,LKED,GØ) to compile FØRTRAN source code, link edit the object modules generated by the compiler, and execute the resulting load module.

Catalogued options:

LKED step	LIST,LET,MAP

4. Link edit and execute

FØRTGLG	Member name

Purpose: A two-step procedure (LKED,GØ) to link edit previously generated object modules and to execute the resulting load module.

Catalogued options:

LKED step	LIST,LET,MAP

3.3.3 Assembler Catalogued Procedures

The standard (or default) options which have been generated for the Assembler are:

NØLØAD	No object module is generated by the compiler
DECK	An object deck is punched for the compiled program
LIST	Assembly code is printed
NØTEST	Source symbol table is not added to the object module
XREF	A cross-reference table of symbols is printed
LINECNT=56	Number of lines per page for the assembly listing
NØRENT	No checking is done by the assembler for possible coding violations of program re-enterability

The stepnames are the same throughout all Assembler catalogued procedures.

stepname	step function
ASM	assembly step
LKED	linkage editor step
GØ	execution step

The FØRTRAN Subroutine Library, SYS1.FØRTLIB, and the SCC Extrinsic Library, SYS2.XTRINSIC, have been defined in the ASMFCL and ASMFCLG procedures as automatic call libraries. The libraries will be searched in an attempt to resolve any unresolved symbols.

1. Assemble only

ASMFC	Member name
ASM	Alias name

Purpose: A one-step procedure (ASM) to assemble Assembler Language code for diagnostic messages for source code debugging.

Catalogued options:

ASM step LØAD,NØDECK

2. <u>Assemble and link edit</u>

ASMFCL	Member name
ASMLINK	Alias name

Purpose: A two-step procedure (ASM,LKED) to assemble Assembler Language source code and link edit the object module generated by the assembler.

Catalogued options:

ASM step	LØAD,NØDECK
LKED step	LIST,MAP

3. <u>Assemble, link edit, and execute</u>

ASMFCLG	Member name
ASMGØ	Alias name

Purpose: A three-step procedure (ASM,LKED,GØ) to assemble Assembler Language source code, link edit the object modules generated by the assembler, and execute the resulting load module.

Catalogued options:

ASM step	LØAD,NØDECK
LKED step	LET,LIST,MAP

3.3.4 PL/1 Catalogued Procedures

The standard (or default) options which have been generated for the PL/1 compiler are:

EBCDIC	Indicates that the source deck is punched in EBCD (029) code
CHAR60	Character set used by source code -- 60 character set
NØMACRØ	The compile-time processor is bypassed
SØURCE2	Input to the compile-time processor is printed
CØMP	Compile-time processing is followed by compilation
SØURCE	Source code is printed
NØATR	No listing of the attributes of identifiers is provided
NØXREF	No listing of cross-references to identifiers is provided
SØRMGIN=(002,072)	Margins set for scanning source statements
NØEXTREF	No listing of the external symbol dictionary is provided
NØLIST	Object code produced by the compiler is not printed
LØAD	An object module is generated by the compiler which may be used as input to the linkage editor
NØDECK	No object deck is punched for the compiled program
FLAGW	Level of diagnostic messages printed; FLAGW indicates all levels
NØSTMT	Diagnostic messages during execution include only hexadecimal offsets relative to the entry points -- statement numbers from the source program are not included
SIZE=413696	The amount of main storage allocated to the compiler
LINECNT=057	Number of lines per page for a source listing
ØPT=00	No optimization of the object code produced by the compiler

The stepnames are the same throughout most of the PL/1 catalogued procedures:

stepname	step function
PL1L	compilation step
LKED	linkage editor step
GØ	execution step

Note: The one exception to the above is the stepname for the catalogued procedure PL1DFC:

stepname	step function
PL1D	compilation step

The PL/1 Subroutine Library, SYS1.PL1LIB has been defined in the PL1LFCL, PL1LFCLG and PL1LFLG procedures as the only automatic call library. The library will be searched in an attempt to resolve any unresolved symbols.

1. Compile only

PL1LFC	Member name

Purpose: A one-step procedure (PL1L) to compile PL/1 source code for diagnostic messages for source code debugging.

Catalogued options:

PL1L step	LØAD,NØDECK

2. Compile only, object deck punched

PL1DFC	Member name

Purpose: A one-step procedure (PL1D) to compile PL/1 source code and get an object deck punched.

Catalogued options:

PL1D step	DECK,NØLØAD

3. Compile and link edit

PL1LFCL	Member name

Purpose: A two-step procedure (PL1L,LKED) to compile PL/1 source code and link edit the object module generated by the compiler.

Catalogued options:

PL1L step	LØAD,NØDECK
LKED step	LIST,LET,MAP

4. Compile, link edit, and execute

PL1LFCLG	Member name

Purpose: A three-step procedure (PL1L,LKED,GØ) to compile PL/1 source code, link edit the object modules generated by the compiler, and execute the resulting load module.

Catalogued options:

PL1L step	LØAD,NØDECK
LKED step	LIST,LET,MAP

5. Link edit and execute

PL1LFLG	Member name

Purpose: A two-step procedure (LKED,GØ) to link edit previously generated object modules and execute the resulting load module.

Catalogued options:

LKED step	LIST,LET,MAP

3.3.5 Miscellaneous Catalogued Procedures

1. Link edit only

LKED	Member name
LINK	Alias name

Purpose: A one-step procedure (LKED) to link edit object modules generated previously by the assembler or any compiler.

Catalogued options:

LKED step	LIST,LET,MAP

The FØRTRAN Subroutine Library, SYS1.FØRTLIB, the PL/1 Subroutine Library, SYS1.PL1LIB, and the SCC Extrinsic Library, SYS2.XTRINSIC, have been defined in this procedure as automatic call libraries. The libraries will be searched in an attempt to resolve any unresolved symbols.

2. Link edit and execute

LKEDG	Member name

Purpose: A two-step procedure (LKED,GØ) to link edit object modules (card input only) generated previously by the assembler or any compiler and execute the resulting load module.

Catalogued options:

LKED step	LIST,LET,MAP

The FØRTRAN Subroutine Library, SYS1.FØRTLIB, the PL/1 Subroutine Library, SYS1.PL1LIB, and the SCC Extrinsic Library, SYS2.XTRINSIC, have been defined in this procedure as automatic call libraries. The libraries will be searched in an attempt to resolve any unresolved symbols.

3.3.6 Notes

1. The job control statements which are catalogued have been assigned sequence numbers. The sequence numbers have the following format:

   00000000 (identifies the first control statement in the catalogued procedure) with an increment of 100 added to each subsequent control statement in the same procedure. If the user supplies the MSGLEVEL=1 parameter in his JØB statement, the catalogued JCL will be listed with the sequence numbers so that the catalogued JCL is easily identified.

2. The linkage editor options for each catalogued procedure have been listed where applicable. The following is a list of the options used and an explanation of each one.

LET	Informs the linkage editor to mark the load module executable even though error conditions, which could cause execution to fail, have been detected.
LIST	Indicates that linkage editor control statements are listed in card-image format in the diagnostic output data set specified by the SYSPRINT DD statement.
MAP	Informs the linkage editor to produce a map of the load module. This map indicates the relative location and length of main programs and subprograms.

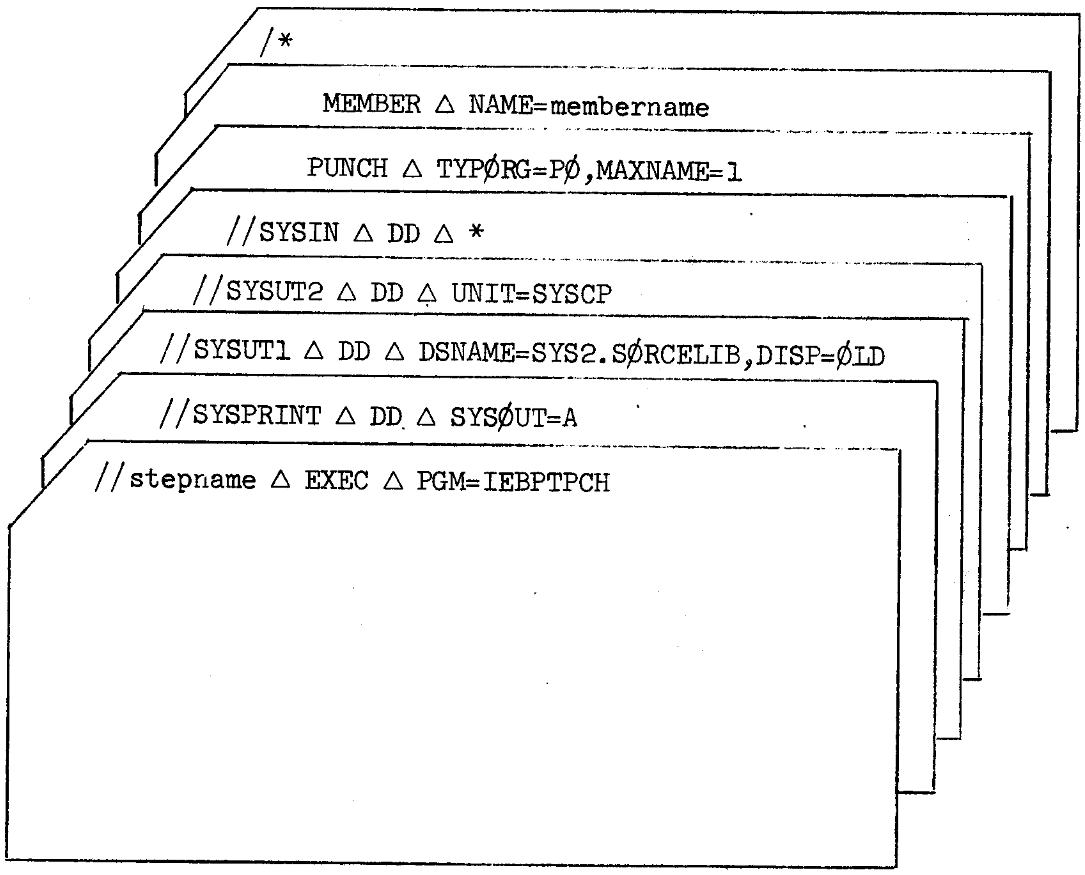
3.4 Utilities

Information on the utilities for the 360/67 can be found in the reference manual IBM Operating System/360 Utilities User's Guide, Form No. C20-1661. Copies are available for reference in the study room and the Systems Documentation Office.

The following sections contain useful programs that have been included for your convenience.

3.4.1 Obtaining Source Decks of Programs or Subprograms in the Source Library

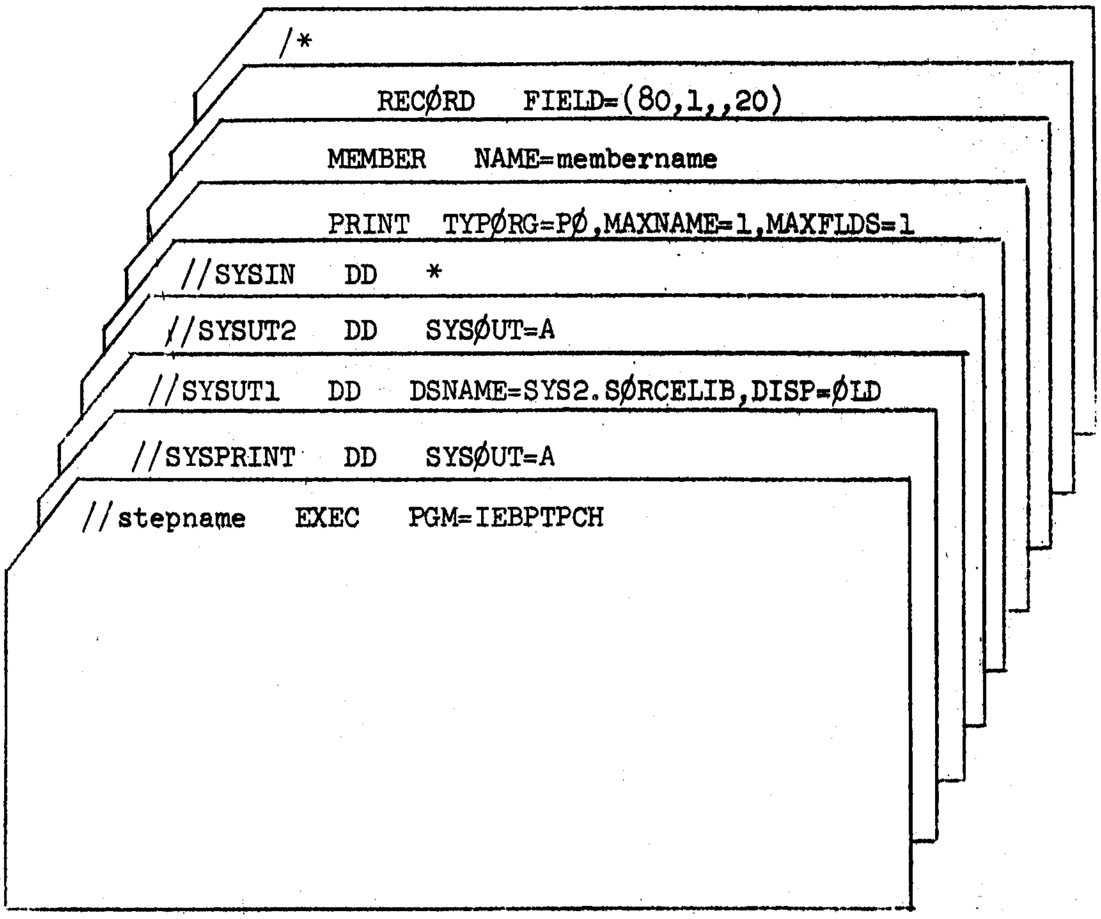
This Job Control Language coding may be used to punch source decks of any of the programs or subprograms in the Source Library.



```
/*
          MEMBER △ NAME=membername
          PUNCH △ TYPØRG=PØ,MAXNAME=1
//SYSIN △ DD △ *
//SYSUT2 △ DD △ UNIT=SYSCP
//SYSUT1 △ DD △ DSNAME=SYS2.SØRCELIB,DISP=ØLD
//SYSPRINT △ DD △ SYSØUT=A
//stepname △ EXEC △ PGM=IEBPTPCH
```

3.4.2 Obtaining Source Listings of Programs or Subprograms in the Source Library

This Job Control Language coding may be used to print a listing of any of the programs or subprograms in the Source Library.



```
/*
          RECØRD    FIELD=(80,1,,20)
          MEMBER    NAME=membername
          PRINT  TYPØRG=PØ,MAXNAME=1,MAXFLDS=1
//SYSIN     DD    *
//SYSUT2    DD    SYSØUT=A
//SYSUT1    DD    DSNAME=SYS2.SØRCELIB,DISP=ØLD
//SYSPRINT  DD    SYSØUT=A
//stepname    EXEC    PGM=IEBPTPCH
```

3.4.3 Character Conversion

Users who wish to convert their programs from the 026 character set to the corresponding 029 characters may do so with the program CHARCØNV.

```
//   JOB card
//JØBLIB △ DD △ DSNAME=SYS2.PRØGLIB,DISP=(ØLD,PASS)
//  △  EXEC △ PGM=CHARCØNV
//FT07F001 △ DD △ UNIT=SYSCP
//FT06F001 △ DD △ SYSØUT=A
//FT05F001 △ DD △ *

    Data cards to be reproduced with the EBCDIC character set

/*
```

Note: This conversion will not correctly convert the special characters of the ALGOL Extended Character Set.

3.4.4 How to Reproduce a Card Deck

Users who want to reproduce a deck without performing character conversion should use the program called REPRØ. REPRØ has all the features of CHARCØNV except character conversion.

```
//   JOB card
//JØBLIB △ DD △ DSNAME=SYS2.PRØGLIB,DISP=(ØLD,PASS)
//STEP1 △ EXEC △ PGM=REPRØ
//FT07F001 △ DD △ UNIT=SYSCP
//FT06F001 △ DD △ SYSØUT=A
//FT05F001 △ DD △ *

    Data cards to be reproduced
/*
```

Note: When using REPRØ OR CHARCØNV, it is possible to inhibit the punching of cards. This is done by supplying a dummy DD card for FT07F001, e.g.,

```
//FT07F001 △ DD △ DUMMY
```

Added May 1968

3.4.5 PRØCLIB Listing

The following deck setup can be used to obtain a listing of the Catalogued Procedure Library (PRØCLIB).

```
//   Job Card
// △ EXEC △ LISTPRØC
/*
```

3.5 Using Direct-Access Storage Devises

3.5.1. Reserved File Volumes Available

The following 2314 volumes are available for reserved file storage:

SYS04
SYS05
SYS06
SYS07
SYS08
SYS09
SYS12
SYS13
SYS14
SYS15
SYS16
SYS17

Users must obtain information regarding the availability of space on a particular volume before trying to allocate a new data set on the volume. Each morning the VTOC (Volume Table of Contents) listing for the available volumes is posted on the bulletin board outside Room 184, Pine Hall. Users are advised to check the listing and allocate new data sets on the volume with the most space available.

WYLBUR users who do not have access to the VTOC listings may examine the file &T000.SPACE LRECL=121. In it they will find information concerning the space available on each of the volumes.

Users are cautioned that &T000.SPACE and the VTOC listings may be out of date by the time they go to use the particular volume selected. Prudent use of the information, however, should be an aid in finding space for the storage of programs and data run in the batch stream. Failure to select a volume with enough space available will result in a wasted computer run.

3.5.2 Space Allocation for Data Sets on the 2314 Disk

There are several factors which must be considered when setting up data set definitions for direct access files.

1. Space is used more efficiently if large blocks of data are written, rather than small blocks. For example, this means that if card images are stored on disk, many more cards may be stored per track if blocks of 44 cards (3520 bytes) are written, rather than blocks of one card. The following DCB attributes would be specified for this type of blocking.

   ```
   DCB=(RECFM=FB,BLKSIZE=3520,LRECL=80)
   ```

2. A table giving the correlation between physical record size and number of records per track (see Introduction to IBM System/360 Direct Access Storage Devices and Organization Methods, Form No. C20-1649) indicates that on the 2314, using sequential access methods, 2 physical records of 3520 bytes may be written on one track. Thus with the above DCB attributes 88 cards may be written per track. The only block size which is more efficient for storing card images (80 bytes each) on the 2314 is a block size of 7120, which allows 89 cards per track. However, WYLBUR cannot accept blocks larger than 3624 bytes, so that we advise the use of the 3520-byte blocks, in most cases. The block size should be a multiple of the logical record length.

3. If 2000 cards are to be written, 2000/88 = 22.7 tracks will be required. The following space allocation could be used to create a data set for storing the cards on the 2314 disk.

   ```
   SPACE=(TRK,(23,2),RLSE)
   ```

   Space is normally allocated by tracks, since it is easier to find free tracks than free cylinders. (A cylinder contains 20 tracks.)

3.5.3 Creating and Using Private Libraries

Programs can be stored in load module form on the 2314 Disk Storage Device. Users wishing to use this feature should be familiar with the section in the Programmer's Guide (For FORTRAN IV (Level H), Form No. C28-6602) that discusses private libraries and Partitioned Data Sets (PDS).

Reserved file storage on the 2314 Disk Storage Drive is limited to the volumes listed in Section 3.5.1. Every data set stored on the 2314 must have a valid account number as the first four characters of its name.

The following examples illustrate an easy method for creating and using your own private program library:

```
//  Job Card
//STEP1 △ EXEC △ FØRTHCL
//FØRT.SYSIN △ DD △ *
      Fortran Source Decks
/*
//LKED.SYSLMØD △ DD △ DSNAME=PO31FIRE(MATCH),DISP=(NEW,KEEP),        *
//              UNIT=2314,VØLUME=SER=SYS07,                          *
//              SPACE=(CYL,(10,10,1),RLSE)
```

This job will put a program on SYS07 so that it can be used later. The important job control statement in this job is the one that begins //LKED.SYSLMØD.

SYSLMØD (System Load Module) is the <u>DDNAME</u> of the output from the linkage editor, and that DD card is in the LKED step of the catalogued procedure FØRTHCL. We want to override all of the parameters on that card so we supply our own:

DSNAME=PO31FIRE(MATCH)	- The name of our data set is PO31FIRE. The output of the linkage editor is always a Partitioned Data Set (PDS), so we have to show a member name also -- here, it is MATCH.
DISP=(NEW,KEEP)	- Keep this new data set at the end of the job.

UNIT=2314	- This data set goes onto a 2314 Disk Storage Device.
VØLUME=SER=SYS07	- The serial number of the disk is SYS07.
SPACE=(CYL,(10,10,1),RLSE)	- Allocate space for this data set by cylinders. Primary allocation is 10 cylinders, secondary allocation is 10 cylinders, one-256 byte block is needed for the PDS directory, and release all unused storage after the data set is created to avoid being charged for it.

```
//     Job Card
//JØBLIB △ DD △ DSNAME=P031FIRE,DISP=(ØLD,PASS),                          *
//                 VØLUME=SER=SYS07,UNIT=2314
//STEP1 △ EXEC △ PGM=MATCH
//FT06F001 △ DD △ SYSØUT=A
//FT07F001 △ DD △ UNIT=SYSCP
//FT05F001 △ DD △ *

         Data for Fortran Program
/*
```

This job will execute the program that was created by the previous job shown in this section. In order to execute a program that resides in a private library, a DD card is required whose DDNAME is JØBLIB. This card must be the second card in the job. There can be only one JØBLIB card in a job. The parameters on the JØBLIB card are:

DSNAME=P031FIRE	- The name of the private library where the program resides.
DISP=(ØLD,PASS)	- This disposition is required.
VØLUME=SER=SYS07	- The serial number of the volume that contains the private library.
UNIT=2314	- The library resides on a 2314 Disk Storage Device.

Notice how the DSNAME, VØLUME, and UNIT parameters match, exactly, those that were used when the library (or PDS) was created.

The 3rd card is an EXEC card with stepname STEP1. The operand of this card is PGM=MATCH, which points to a member of the PDS PO31FIRE. This job step will execute the program named MATCH. MATCH is a FORTRAN program, so some FORTRAN DD cards are required.

```
        //FT06F001 DD SYSØUT=A    - if there is any printer output.

        //FT07F001 DD UNIT=SYSCP  - if there is any punched card output.

        //FT05F001 DD *           - if there is any card input.
```

3.5.4 Leaving Load Modules in the System

How to use disk storage to avoid recompilation of checked out programs, or to avoid the use of object decks.

This section is intended to provide a clear and easy-to-follow set of procedures for storing checked out portions of programs on disk in load module form. It also illustrates how to create a private library once a program and its subroutines are completely checked out and the program is ready for production. This saves both re-compile and re-linkedit time.

1. To put the first load module in a data set (and create the data set at the same time), the following job control statements should be followed.

```
//    Job Card
//STEP1 △ EXEC △ FØRTHCL,PARM.LKED='NCAL'
//FØRT.SYSIN △ DD △ *
         Source Deck
/*
//LKED.SYSLMØD △ DD △ DSNAME=annnWHØL,DISP=(NEW,KEEP),              *
//               VØLUME=SER=SYS04,UNIT=2314,                         *
//               SPACE=(TRK,(2,2,4),RLSE)
//LKED.SYSIN △ DD △ *
     NAME  PRØG1
/*
```

This places the object module, produced from the preceding FORTRAN compilation, in the partitioned data set named annnWHØL. (NOTE: Users should remember to use their account numbers and data set names. annnWHØL is used in this section for illustrative purposes and should not be copied.) The member name of the load module is PRØG1. A member name can contain up to 8 characters.

2. To add subsequent load modules to the data set, the following job control statements should be followed.

```
//    Job Card
//STEP1 △ EXEC △ FØRTHCL,PARM.LKED='NCAL'
//FØRT.SYSIN △ DD △ *
         Source Deck
/*
//LKED.SYSLMØD △ DD △ DSNAME=annnWHØL,DISP=(ØLD,KEEP),UNIT=2314,       *
//                 VØLUME=SER=SYS04
//LKED.SYSIN △ DD △ *
      NAME     PRØG2
/*
```

3. To compile FORTRAN programs and link them to programs previously stored in load module form in data set annnWHØL use the following JCL statements.

```
//    Job Card
//STEP1 △ EXEC △ FØRTHCLG
//FØRT.SYSIN △ DD △ *
         Source Deck(s)
/*
//LKED.DDNAM △ DD △ DSNAME=annnWHØL,DISP=(ØLD,KEEP),                  *
//                 VØLUME=SER=SYS04,UNIT=2314
//LKED.SYSIN △ DD △ *
      INCLUDE DDNAM(PRØG1,PRØG2)
      ENTRY MAIN
/*
```

The list enclosed in parentheses in the INCLUDE statement should contain the names of all members that the user wishes to be used in this run.

4. When the program is completely checked out, a private library should be created. This is a partitioned data set containing a load module of the complete program. The JCL needed to create this private

library is illustrated below.

```
//   Job Card
//STEP1 △ EXEC △ LKED
//LKED.SYSLMØD △ DD △ DSNAME=annnPRIV,DISP=(NEW,KEEP),UNIT=2314,          *
//               VØLUME=SER=SYS04,SPACE=(CYL,(1,1,1),RLSE)
//LKED.DDNAME △ DD △ DSNAME=annnWHØL,DISP=(ØLD,KEEP),                     *
               UNIT=2314,VØLUME=SER=SYS04
//LKED.SYSIN △ DD △ *
     INCLUDE  DDNAME(PRØG1,PRØG2)
     ENTRY  MAIN
     NAME  PRVLIB
/*
```

The list enclosed in parentheses in the INCLUDE statement should contain the names of all members that the user wishes included in the final load module.

5. To use a private library, the following JCL statements should be followed.

```
//   Job Card
//JØBLIB △ DD △ DSNAME=annnPRIV,DISP=(ØLD,PASS),                          *
//               UNIT=2314,VØLUME=SER=SYS04
//STEP1 △ EXEC △ PGM=PRVLIB
//FT06F001 △ DD △ SYSØUT=A
//FT05F001 △ DD △ *

     Data Cards
/*
```

Note that the PGM name on the EXEC card should be the member name of the private library. Use of a private library means it is not necessary to do a link edit each time a job is run.

At this point the user has two partitioned data sets (1) annnWHØL which contains several members (a maximum of 4 per directory block), each of which is a separate load module; and (2) a data set containing a single member which can be executed without re-linkediting.

6. If a programmer wishes to change a load module (i.e., make a change in a program already entered in annnWHØL), this can be done with the following JCL statements.

```
//   Job Card
//STEP1 △ EXEC △ FØRTHCL,PARM.LKED='NCAL'
//FØRT.SYSIN △ DD △ *
    Source Deck
/*
//LKED.SYSLMØD △ DD △ DSNAME=annnWHØL,DISP=(ØLD,KEEP),                    *
//             UNIT=2314,VØLUME=SER=SYS07
//LKED.SYSIN △ DD △ *
   NAME    PRØG1(R)
/*
```

A new private library, containing the revised PRØG1, can then be created as shown below.

```
//   Job Card
//STEP1 △ EXEC △ LKED
//LKED.SYSLMØD △ DD △ DSNAME=annnPRIV,DISP=(ØLD,KEEP),                    *
//             UNIT=2314,VØLUME=SER=SYS04
//LKED.DDNAME △ DD △ DSNAME=annnWHØL,DISP=(ØLD,KEEP),                     *
//             UNIT=2314,VØLUME=SER=SYS04
//LKED.SYSIN △ DD △ *
    INCLUDE  DDNAME(PRØG1,PRØG2)
    ENTRY    MAIN
    NAME     PRVLIB(R)
/*
```

This private library can then be used with the same JCL set up as shown in step 5.

In all of the above examples, the FORTRAN H compiler is referenced. The same procedure may be used with FORTRAN G by calling the FORTRAN G catalogued procedures.

7. <u>Compressing dead space out of a data set.</u> Replacing a member of a partitioned data set with a new member creates dead space in the data set. This dead space can be removed by doing a move of the data set. There are two alternative ways of doing this. One involves moving the data set from one volumne to another with the data set name remaining the same. The second involves moving the data set to another place on the same volume, in which case the data set name must be changed.

(1) The JCL statements used for moving the data set to a new volume are shown below.

```
//   Job Card
//STEP1 △ EXEC △ PGM=IEHMØVE
//SYSPRINT △ DD △ SYSØUT=A
//ØLDSET △ DD △ UNIT=2314,VØLUME=SER=SYS04,DISP=ØLD
//NEWSET △ DD △ UNIT=2314,VØLUME=SER=SYS07,DISP=ØLD
//SYSUT1 △ DD △ UNIT=2314,VØLUME=SER=SYS03,DISP=ØLD
//SYSIN △ DD △ *
      CØPY  PDS=annnWHØL,TØ=2314=SYS07,FRØM=2314=SYS04
/*
```

After such a move, the VØLUME parameter of the DD cards described in steps 1-6 above must be changed to reflect the new location of the data set.

(2) The JCL statements needed for moving the data set to a new place on the same volume are shown below.

```
//   Job Card
//STEP1 △ EXEC △ PGM=IEHMØVE
//SYSPRINT △ DD △ SYSØUT=A
//PACK △ DD △ UNIT=2314,VØLUME=SER=SYS04,DISP=ØLD
//SYSUT1 △ DD △ UNIT=2314,VØLUME=SER=SYS03,DISP=ØLD
//SYSIN △ DD △ *
   CØPY  PDS=annnWHØL,TØ=2314=SYS04,FRØM=2314=SYS04,RENAME=annnWHLE
/*
```

After such a move, the JCL statements described in steps 1-6 must now contain the new DSNAME.

3.5.5 Deleting Data Sets

SYS2.PRØGLIB contains a program which will aid the user in deleting data sets prior to their expiration date. The restrictions which apply to this program are:

1. Only data sets with valid DSNAMES can be deleted.
2. Only data sets residing on volumes where reserved files are allowed can be deleted.
3. The format for the data cards used by this program is:

   volume serial number, data set name

   or

   volume serial number, data set name, MEMBER=member name

   Blanks are ignored, so the layout is not critical; however, the sequence of the items must be correct. Example:

```
//    Job Card
//JØBLIB △ DD △ DSNAME=SYS2.PRØGLIB,DISP=(ØLD,PASS)
//STEP1 △ EXEC △ ERASE
//ERASE.SYSIN △ DD △ *
SYS07,   annnDATA
CAMP31,annnDATB
   CAMP31,   annnDATA
   CAMP31,   annnDATA,MEMBER=SUB1
   CAMP31,   annn.A.B.C
/*
```

This job will execute the catalogued procedure "ERASE". "ERASE", a collection of job control statements, executes a program from SYS2.PRØGLIB. This generates the need for the JØBLIB card.

This job will erase:

annnDATA	in	SYS07
annnDATB	in	CAMP31
annnDATA	in	CAMP31
annn.A.B.C	in	CAMP31

Member SUB1 from PDS annnDATA in CAMP31

Reminder! The JØBLIB card must be the second card in the job, and must look exactly like the one shown above.

3.6 WYLBUR

WYLBUR is a text editor designed to operate in either an OS or HASP environment in order to provide users at 2741 terminals a comprehensive text editing facility with prompt response, while otherwise allowing the computer to proceed with processing of the normal batch job stream.

WYLBUR allows the user to change the contents of a line without retyping all of it, to insert, delete or replace lines, to copy and move lines, to renumber lines, and to list lines. In addition, the user may retrieve external data sets in order to work upon them and save data sets which have been constructed. The user may also insert the data set into the monitor's input stream for processing as a batch job.

3.6.1 Obtaining a Terminal

Requests for a remote terminal device must be made on an Interdepartmental Request form number SU-13. Non-Stanford organizations will use a purchase order. This form should contain the following information:

a. University account number to be charged.

b. Period for which service is to be provided.

c. Location point for installation of terminal device. (Must indicate building and room number.)

d. Person(s) in charge of terminal and their telephone number(s).

e. Number of terminals required.

f. Estimated number of users.

g. Existing account numbers or application for new account numbers (see Section 3.6.2) to be authorized use of the terminal, and a keyword for each account number.

Interdepartmental Request forms require the signature of a person authorized to commit funds. The completed form should be sent to R. Montgomery, Pine Hall.

Terminals are rented by the Stanford Computation Center, Campus Facility, from IBM and re-billed to the user who requests installation. The cost to the owner of a campus installation, exclusive of use, is as follows:

Added May 1968

Line Concentrator System

	Monthly	Installation Cost
IBM 2741 Terminal	$ 84.42*	$ 60.00
103A2 Data Set	30.00	60.00
Leased Line (average)	4.00	10.00
Total	$118.42	$130.00

For off-campus installations where business lines are used, the cost is:

Telephone Business Line System

	Monthly	Installation Cost
IBM 2741 Terminal	$ 84.42*	$ 60.00
103A2 Data Set	30.00	60.00
Business Line	11.05	15.00
Total	$125.47	$135.00

*As of July 1, 1968, monthly rental for 2741 terminals will be increased by $13.44/month. After this time the monthly charge for terminal rental will be $97.46/month.

(NOTE: Terminal rent and installation costs are incremental to the SCC Campus Facility approved budget. Therefore, Provost's Funds which are already part of the Campus Facility Budget cannot be used to cover these costs.)

Assuming the availability of IBM 2741 terminals, normal lead time for installation will be three weeks after receipt of an Interdepartmental Request form. Thirty days prior written notice to IBM from the Campus Facility is required to terminate the rent of a 2741; therefore, rent will be charged to the user 30-full days after the Campus Facility receives written notice to terminate. Proration of monthly charges will be based on a 30-day month.

3.6.2 Account Numbers for Terminals

Only account numbers authorized for use on WYLBUR by terminal renters will be accepted by the WYLBUR system.
The Computation Center Accounting Office (Room 164, Polya Hall) provides

account numbers based on an "Application for Services" form prepared by the user. (General information on an account number application and an example of the form used can be found in Section 2.1.1, pg. 2-1.) Whoever pays rent on a terminal must designate which account numbers are authorized to use terminal service. To establish new account numbers for use on terminals, the standard Application for Services form should be used by the terminal renter. If an existing number is to be used on a terminal, the terminal renter must notify the Computation Center Accounting Office in writing (in the Interdepartmental Request for the terminal or by memo). Normally, account numbers received or assigned by the Accounting Office between Monday morning and Friday at 3:00 p.m. will be entered into the System by 8:00 p.m. the same day. Account numbers received or assigned between Friday at 3:00 p.m. and Monday morning will be available for use by 8:00 p.m. Monday. Cancellation of authorized account numbers for terminals will be handled on the same schedule.

When an account number is authorized for terminal use, it is associated with a system key. While authorized account numbers can be used from any terminal, this system will not accept simultaneous use of more than one account number associated with any one system key. For example, account numbers Z000, Z001, and Z002 are associated with system key H12. Only one of these three account numbers can be used from any WYLBUR terminal at one time. However, it is possible for one account number to be associated with more than one system key.

3.6.3 Current Rate Structure for WYLBUR Services

A composite rate has been established to cover access duration and CPU editing time on the WYLBUR System. This charge is applied as a rate per terminal hour. Thus, charges are based upon the total time between "sign-on" and "log-off". Access duration and editing charges will be levied against the account number used to sign onto the system. The current rate is $4/terminal hour.

Batch jobs entered via terminals will result in processor and related charges for batch service in a manner identical to batch charges for jobs not entered from terminals.

The complete rate schedule for Campus Facility services is on page 2-2.

3.6.4 Security Measures for Account Protection

Users who wish to protect their accounts against use by unauthorized persons may accomplish this through use of a KEYWORD. At the time an account is established, the user may specify the 3-character keyword he desires. Each time he signs on from a terminal, the account number and the keyword will be checked for validity. If non-matching keyword and charge number are specified, the user will not be granted access to the system. While use of keyword is not mandatory, it is strongly encouraged.

If a user wishes to change his keyword, he may submit a request to the Accounting Office specifying the account number, the current keyword, and the new keyword. This request should be signed by the holder of the account. Keyword changes are processed by the Accounting Office on the same schedule specified in Section 3.6.2 for processing of account numbers.

3.6.5 Documentation

A complete description of the WYLBUR text editor -- Appendix E of the Users Manual -- may be obtained from the Systems Documentation Office, Room 185, Pine Hall.

Also available to WYLBUR users is the file &T000.NEWS (edit format). This file contains information on new WYLBUR features and changes that may not have been incorporated in the current WYLBUR manual. The first part of this file contains an index to its contents and the entry date for each item put into the file.

3.6.6 Notes

3.6.6.1 Plot Instructions for WYLBUR Users

The SHOW OPERATOR command is used at the remote terminal to communicate tape mounting for plots and necessary plot instructions to the computer operator. Remember that any job requiring a tape must have an S in the appropriate field of the job card (see pg. 3-4 for job card information). The following information is required.

```
 (1)   (2)      (3)       (4)   (5)  (6)       (7)
PLOT, name, account no., paper, pen, ink, special instructions
```

Item (1) enter the word PLOT; item (2) enter your name; item (3) enter your account number. Items (4)-(7) are optional. If omitted, the default specifications indicated for items (4) and (5) in the following table will be used.

ITEM	ENTER		SPECIFICATION Width Description	COLORS
(4) paper	00	default	10", plain	white
	01(*)		10", graph, 10 divisions/inch	*red, blue, olive-brown
	02(*)		10", graph, 20 divisions/inch	*red, blue
	300		30", plain	white
	301		30", graph, 10 divisions/inch	olive-brown
(5) pen	.3		.3mm, fine	
	.5	default	.5mm, medium	
	.6		.6mm, broad	
(6) ink	red			
	blue			
	black (default)			
	green			

Example:

```
COMMAND? show operator 'PLOT, J. Smith, A000, 01(blue), red'
```

The operator will mount a plot tape. Output will be plotted on 10" blue graph paper (10 divisions/inch) with a .5 (medium) pen in red ink.

3.6.6.3 Tape Handling Instructions for WYLBUR Users

To indicate a tape mount (other than for plotting) to the computer operator use the SHOW OPERATOR command and enter the following information. Remember that any job requiring tape must have an S in the appropriate field of the job card (see pg. 3-4 for job card information).

```
 (1)    (2)        (3)         (4)    (5)         (6)      (7)        (8)
                                     write                          special
TAPE, name, account no., reel, enable, channels, drive, instructions
```

For item (1) enter the word TAPE; item (2) your name; item (3) your account number; items (4)-(7) are explained in the table below.

ITEM	ENTER	
(4) reel	enter reel I.D. if a reserved tape or private tape	
	SCRATCH	operator will mount a scratch tape
	RESERVE	operator will mount a scratch tape and reserve it to the name and account as shown
(5) write enable	WR	operator will write enable the reel; if no entry, the tape will be file protected
(6) channels	7	tape will be mounted on a 7 channel tape drive
	9	tape will be mounted on a 9 channel tape drive
(7) drive	enter physical address of drive if required	

Example:

```
        COMMAND?  show operator 'TAPE, J. Smith, A000, SCC000, 9'
```

The operator will mount tape SCC000 on a 9-track drive. The tape will be file protected.

3.6.6.3 T000.JCL

T000.JCL is intended to help WYLBUR users with Job Control Language. In order to get such information as the FORTRAN compile listing, the LKED map, and the execution output into data sets that can be accessed by WYLBUR (instead of the usual printed output from the printer at the Campus Facility), it is necessary to override the SYSPRINT and SYSØUT definitions for the COMPILE, LKED, and GØ steps, respectively. T000.JCL provides the necessary JCL for doing this plus instructions for copying statements from T000.JCL to a terminal work area. The file now contains Job Control Language for the following applications:

Added May 1968

1. Saving the following types of output in a data set on the 2314 disk, to be referenced later from a terminal:

   FORTRAN IV Level H compiler listing,

   Linkage map and error diagnostics,

   FORTRAN IV Level H program execution listing,

   WATFOR compiler and program execution listing,

   Assembler Level F listing,

   Assembler Level F execution listing,

   PL/1 compiler listing,

   BMD program execution listing.

2. Reading a card deck in from a batch job and saving it on a data set on the disk for future reference from WYLBUR.

3. Printing or punching data sets created by WYLBUR.

4. JCL for SØRT, with input from a data set stored in card format on the disk and output to the disk.

5. JCL for transferring a source copy of library programs from SYS2.SØRCELIB into a data set for modification and use from WYLBUR.

The first part of T000.JCL contains a list of the various JCL applications available. Users should list the file offline before using, as the line numbers for the various DD statements may have been changed.

3.6.6.4 WYLBUR Files Blocked

WYLBUR files are saved in blocked format. Files which are saved in card format will be blocked 44 records/block; in print format, 26 records/block. In order to save files in unblocked format, the user must specify a blocking factor of 1 in his save command:

```
SAVE dsname CARD (1)
```

If files are saved with the format specified by the 'LRECL=' option, each block will contain one record.

When retrieving files that have been saved by WYLBUR, you should specify the following parameters in the data control block field of the DD card:

Added May 1968

WYLBUR command used	DCB field
SAVE dsname CARD	DCB=(RECFM=FB,LRECL=80,BLKSIZE=3520)
SAVE dsname PRINT	DCB=(RECFM=FB,LRECL=133,BLKSIZE=3458)
SAVE dsname LRECL=nnn	DCB=(RECFM=F,LRECL=nnn,BLKSIZE=nnn)

The blocking factors used by WYLBUR allow storage of 88 cards/track, in card format, and 52 lines/track in print format. Since this is very efficient use of space on the 2314 disk, we suggest that WYLBUR users use the same blocking factors when creating files to be referenced through WYLBUR. For example, to create an output file from FORTRAN H in print format, with 300 lines of output expected, you would use the following DD statement:

```
//GØ.FT06F001  DD SYSØUT=,DISP=(NEW,KEEP),UNIT=2314,             X
//               VØLUME=SER=SYSnn,                                X
//               DCB=(RECFM=FB,LRECL=133,BLKSIZE=3458),           X
//               DSNAME=Annn.DSNAME,SPACE=(TRK,(6,2),RLSE)
```

Note that 6 tracks are allocated in the SPACE parameter. This allocation is enough for 312 lines of output. The secondary allocation of 2 tracks will allow space for up to 780 more lines.

3.6.6.5 &T000.SPACE

WYLBUR users needing to know the amount of space available on various volumes may examine &T000.SPACE LRECL=121. In it they will find information concerning the space available on each of the volumes where the storage of data sets is permitted.

The program which creates this file of information is run at least once a day and is generally as current as the VTOC listing posted daily outside of Room 184, Pine Hall. (See page 3-34.)

Users are cautioned that this information may be out of date by the time they go to use the particular volume selected. Prudent use of the information, however, should be an aid in finding space for the storage of programs and data run in the batch stream.

4. LANGUAGE PROCESSORS

4.1 FØRTRAN

4.1.1 Description of the Language

FØRTRAN is one of the most commonly used programming languages. It is especially useful in writing programs for scientific and engineering applications that involve mathematical computations. FØRTRAN is usually available in some form at even the smallest computer installations, and a variety of library and applications programs are available for easy use. In fact, the development of FØRTRAN is so widespread that the American Standards Association has defined a basic FØRTRAN language (ASA FØRTRAN) which specifies the fundamental elements of the language.

FØRTRAN IV for the Operating System 360 is compatible with and encompasses ASA FØRTRAN, including its mathematical subroutine provisions. It is described in the IBM System/360 FØRTRAN IV Language Manual (Form No. C28-6515) and is supplemented by the mathematical subroutines in IBM's Scientific Subroutine Package (described in the form H20-0205). The SSP contains a number of commonly used routines for data screening, regression, analysis of variance, non-parametric statistics, matrix operations, integration and differentiations, etc.

In addition, the Stanford Extrinsic Program Library and the Applications Program Library have been locally developed to provide additional facilities to the FØRTRAN user. Programs for plotting, numerical integration, linear system solution, matrix inversion, linear least squares solution, the UCLA Bio-Medical Computer Programs (BMD's) and the Electronic Circuit Analysis Program (ECAP) are among those included in these libraries.

The major library development and consulting effort of the Campus Facility is directed toward the support of FØRTRAN IV (levels G and H).

4.1.2 Documentation

1. IBM System/360 FORTRAN IV Language, Form No. C28-6515.

   This publication describes the IBM System/360 FORTRAN IV Language for the IBM System/360 Operating System. It provides a quick definition and syntactical reference to the various elements of the language.

2. IBM System/360 Operating System FORTRAN IV (G) Programmer's Guide, Form No. C28-6639.

   This publication describes how to compile, link edit, and execute a program written in FORTRAN IV using the G-level compiler. It gives details on the implementation of the FORTRAN IV language and how the programmer communicates with the system to execute his job.

3. IBM System/360 Operating System FORTRAN IV (H) Programmer's Guide, Form No. C28-6602.

   This publication describes how to compile, link edit, and execute a program written in FORTRAN IV using the H-level compiler. It gives details on the implementation of the FORTRAN IV language and how the programmer communicates with the system to execute his job.

4. IBM System/360 Operating System FORTRAN IV Library Subprograms, Form No. C28-6596.

   This publication describes the library subprograms supplied with FORTRAN IV and provides the information necessary to use the library subprograms in either a FORTRAN IV or an assembler language program. In addition, the publication contains algorithms, accuracy statistics and timing estimates, descriptions of error and interruption procedures, storage estimates, and sample storage printouts.

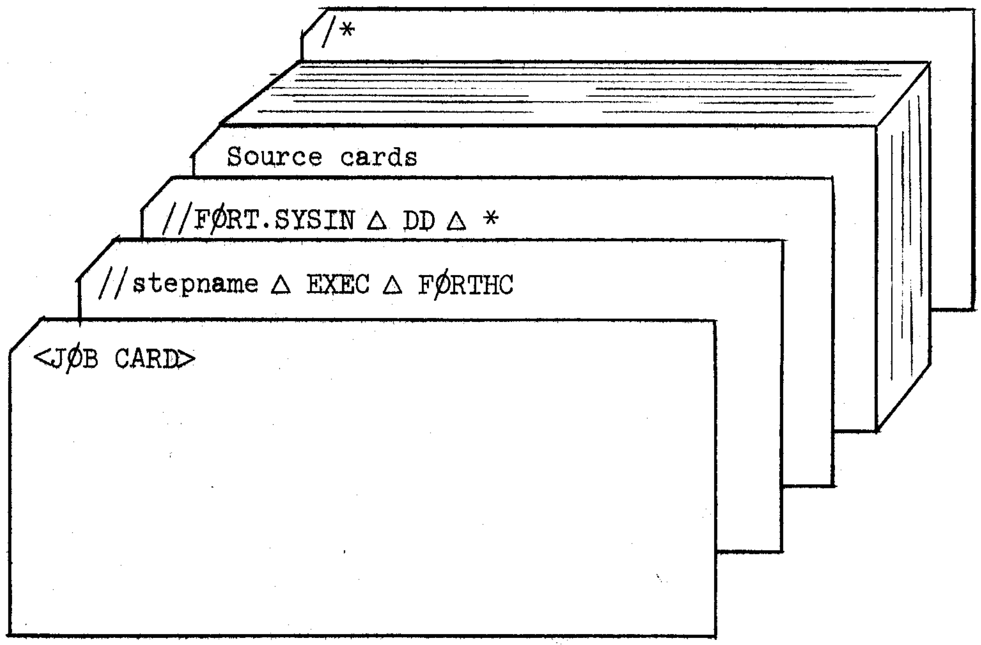
4.1.3 Sample Deck Set-ups

Note 1: All of the control cards (i.e., // and /*) must be punched starting in column 1. Blanks, at least one, are required where the symbol Δ appears. Blanks must not appear in other fields on the card.

Note 2: The "stepname" information on the EXEC card can be used at the option of the programmer. If it is used, the programmer may choose any stepname that he wishes -- however, it cannot be longer than 8 characters. The first character of this name must immediately follow the slash -- there can be no blanks. If the stepname is omitted, there must be at least 1 blank after the second slash before the word EXEC.

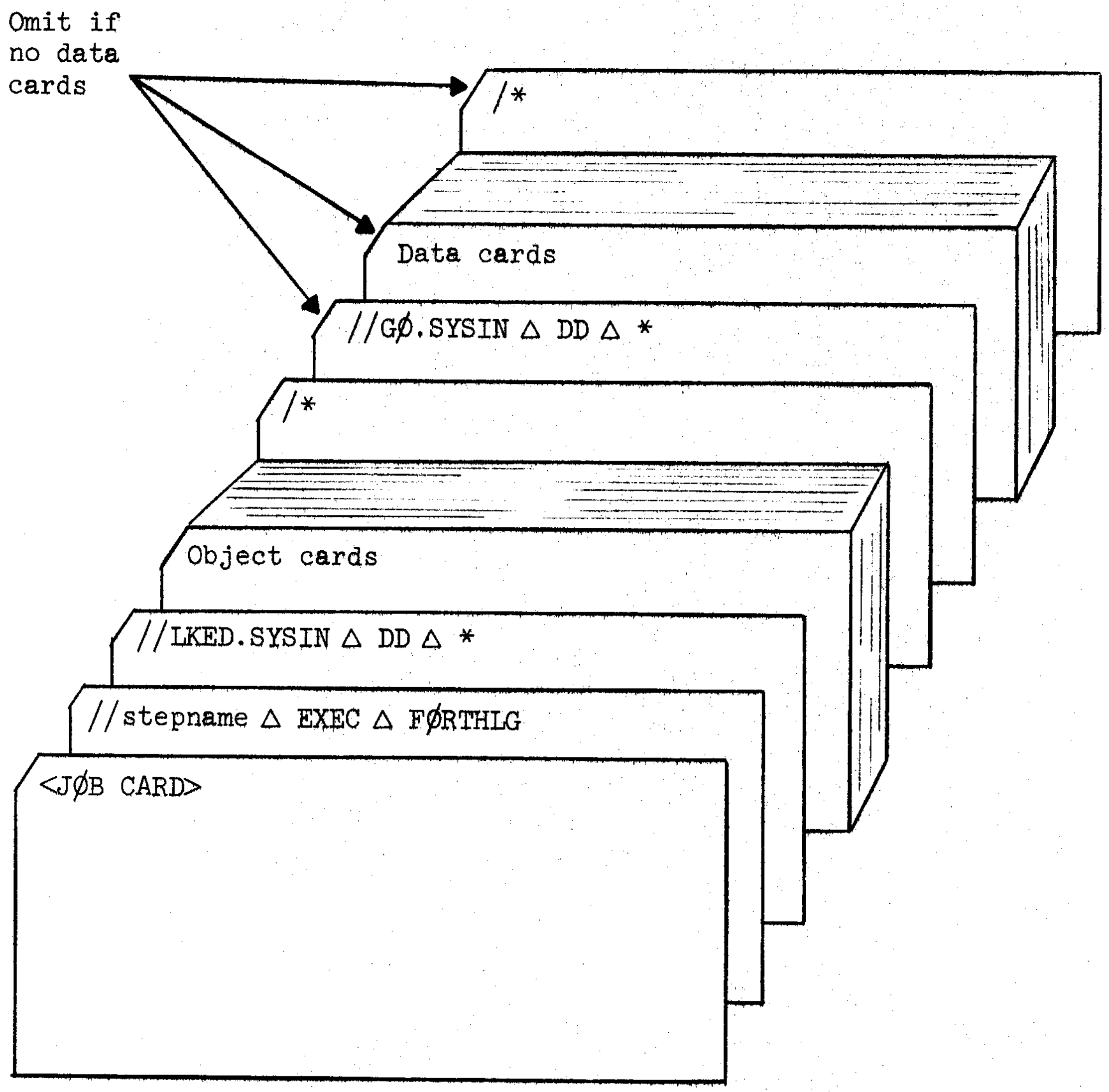
Note 3: The deck set-ups illustrated can also be used with FORTRAN G. To do so, change the catalogued procedure in the EXEC statement to the corresponding form for FORTRAN G (i.e., change FØRTHC to FØRTGC, etc.)

1. Deck set-up for a compile only job:

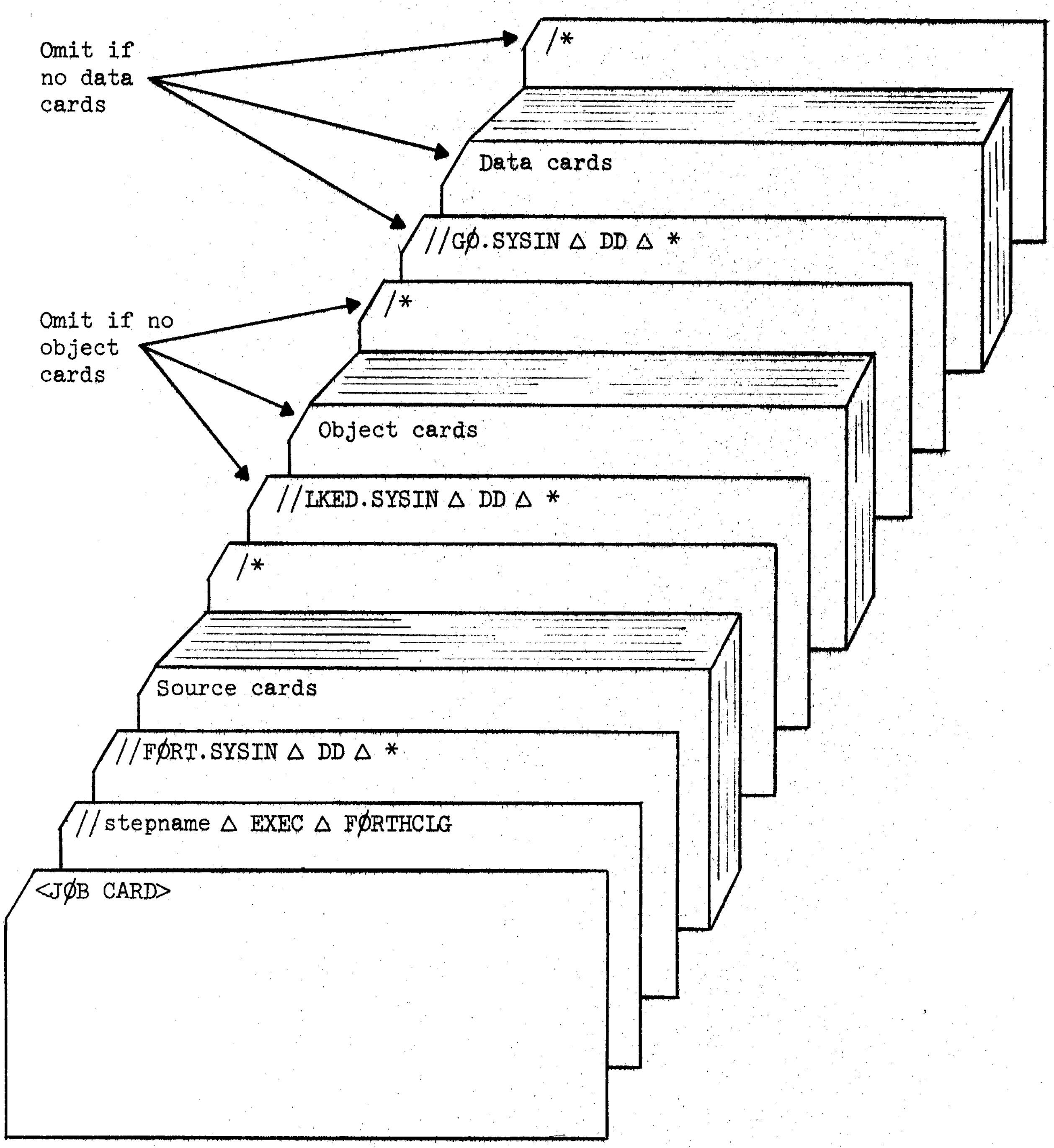


To obtain an object deck from the above, the execute card should be changed as follows: //stepname Δ EXEC Δ FØRTHC,PARM.FØRT='DECK'

2. To link and go previously compiled decks the following deck set-up should be used:



3. To compile, link and go the following deck set-up should be used:



4.1.4 Helpful Hints

1. There have been several significant additions to the language of which the user should be aware. Some of these are:

   - Use of quotes to enclose a Hollerith constant (BCD string) instead of requiring a character count. (The count may still be used.)
   - The IMPLICIT statement used to define and type variables according to their first character.
   - The Explicit type statements used to type variables and, optionally, to assign values to them.
   - Two new format codes, G and T.
   - Addition of double precision complex variables and constants.
   - Allowing mixed mode expressions.
   - The READ statement has been extended to include transfers on end-of-data and/or redundancy error.

2. In converting programs or writing programs that are compatible with both versions of FORTRAN IV (7090 and 360 versions), the major areas of difficulty are:

   - Any use of the format specification A6 must be checked. The corresponding I/O list item must be either redefined as a REAL with length 8 (double precision variables are now termed REAL*8) or the specification must be changed to A4 and the I/O list expanded correspondingly (which may also necessitate changing DIMENSION statements).
   - All COMMON and/or EQUIVALENCE statements must be checked to see that the variables are listed so that they observe the hardware boundaries.
   - Octal constants must be replaced by hexadecimal constants and the use of AND or OR functions must be changed to be consis-

tent with the new word length and character set.

- o Implied DO's must be removed from DATA statements.
- o If 7 significant figures are not sufficient, programs must be converted to double precision. Then constants with less than eight significant digits must be rewritten with a D exponent.
- o Binary tapes written by 7090 FORTRAN cannot be read directly by 360 FORTRAN. (Binary tape conversion routines are being provided. These are discussed in a document prepared by the SLAC Computing Facility of the Computation Center. This document, "7090/7094 to OS/360 Magnetic Tape Conversion," is available from the Systems Documentation Office, Room 185, Pine Hall, Ext. 4877.)
- o Blank fields read with I, E, F, or D format are no longer converted to -0. They are converted to +0.
- o Shifts effected by multiplication or division by 2 must be checked to assure that:
  - a. The shift is no more than 32 positions.
  - b. The shift is compatible with the new representation for negative integers. (Negative integers are stored as the two's complement of their absolute value with the sign position negative.)
  - c. If BCD characters are being manipulated, the number of bits/character is 8 rather than 6.
- o All MAP subprograms must be rewritten.
- o End-of-file trap routines are no longer necessary.

3. Users who wish to convert their own programs from the 026 character set to the corresponding 029 characters may do so with the program CHARCØNV. A sample set-up of this program is in Section 3.4, "Utilities". (Note: This conversion will <u>not</u> correctly convert the special characters of the ALGOL Extended Character Set.)

4. Use the FORTRAN G compiler for debugging during program development and for short, one-shot programs. Once a program has reached the production stage, use FORTRAN H with ØPT=2. The reason for this is that FORTRAN G is a small fast compiler, but it produces a relatively slow object code. On the other hand, FORTRAN H is a larger but slower compiler, though it produces a fast object code.

5. When using exponents use X**2 rather than X**2.0 -- in general, use <expression> ** < integer> or <expression> ** <integer expression> rather than <expression> **<real>. This method will save time and accuracy.

6. Remember, underflow usually occurs when exponentiating numbers or expressions < 1. (e.g., .0005**20).

7. Make sure arguments to subroutines, functions, or arithmetic statement functions are all of correct type and length.

8. Never try (by accident or on purpose) to reset a constant:

```
    CALL ASUB(5)              SUBRØUTINE ASUB (Z)
                              Z = 4
                              RETURN
                              END
```

9. All characters and character handling routines should be written in integer variables rather than real.

10. From the FORTRAN supplied subprograms, pick the function you mean -- use either single precision or double precision, and use the right form of the argument, etc., (e.g., SIN, for single precision variables; DSIN, for double precision variables; the argument in radians, etc.)

11. On input, decimal points override specified field specifications. Otherwise, numbers must be correctly justified. For example: If the format specified is F8.3 and the number on the data card in the first eight columns is 17.0014, then the number of decimal places designated by the format statement (3) will be overridden and the number will appear as on the data card. However, in the

format statement I5, the number must be right justified as embedded and trailing blanks are treated as zeros. Integers on input and output are <ins>always</ins> right justified.

4.2 PL/1

4.2.1 Description of the Language

PL/1 is a multipurpose programming language for use not only by commercial and scientific programmers but by the real-time programmer and the systems programmer as well. PL/1 contains substantially all of the capabilities of FØRTRAN, ALGØL, and BALGØL with some additional features (i.e., list processing capabilities, bit and character string manipulation).

One of the primary aims in the design of the language was modularity, that is, providing different levels of the language for different applications and different degrees of complexity. This means that a programmer experienced in FØRTRAN or ALGØL need not learn all of PL/1, but only a subset of it that closely resembles the language he has been using. The subset that he learns will provide the programmer with the capability to operate efficiently in PL/1.

4.2.2 Documentation

1. IBM System/360 Operating System PL/1 Language Specifications, Form No. C28-6571.

   This manual gives a complete description of the PL/1 language, independent of any particular implementation. It contains a very detailed explanation of the language, though the wording is often unclear and there are few examples given. It is not a manual for inexperienced programmers wishing to learn PL/1.

2. IBM System/360 Operating System PL/1 (F) Programmer's Guide, Form No. C28-6594.

   This manual describes the IBM F level implementation of PL/1 and gives information on how to use it. It is not a language description. It includes:

   a) Language features not supported in the current (2nd) release of PL/1. These features are listed in Appendix H of the manual.

b) Implementation conventions and restrictions (i.e., limits on array bounds, limits on size of an individual statement). These are listed in Appendix B.

c) A detailed description of the implementation related input/output features -- most specifically in the area of Record I/O.

d) A list of the diagnostic messages the compiler produces and other information about compiler output.

e) Debugging information.

f) A discussion of cataloged procedures and job control.

3. A Guide to PL/1 for FORTRAN Users, Form No. C20-1637.

This manual gives a good introduction to PL/1 for programmers who know FØRTRAN. It is not a complete description of the language, but it is clearly presented and contains many examples.

4. A PL/1 Primer, Form No. C28-6808.

This is a good manual for novice programmers who wish to learn PL/1 or for more experienced programmers who want a quick overview of the language. It does not contain a complete description of the language. The material, though, is well presented an includes numerous examples.

5. PL/1 Subset Language Specifications, Form No. C28-6809.

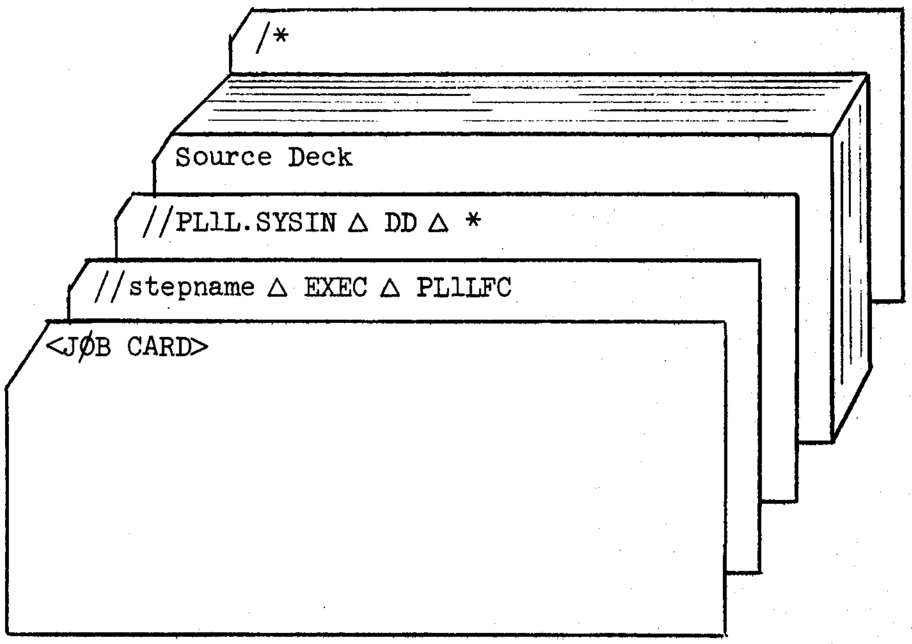
This is a scaled down version of the Language Specifications manual. It describes a subset of PL/1, not a commercial or scientific subset, just a subset. Also, the subset described is not related to any particular implementation of PL/1. It uses examples sparingly and is not an easy manual to read.

4.2.3 Sample Deck Setup

Note 1: All of the control cards (i.e., // and /*) must be punched starting in column 1. Blanks, at least one, are required where the symbol Δ appears. Blanks must not appear in other fields on the card.

Note 2: The "stepname" information on the EXEC card can be used at the option of the programmer. If it is used, the programmer may choose any stepname that he wishes -- however, it cannot be longer than 8 characters. The first character of this name must immediately follow the slash -- there can be no blanks. If the stepname is omitted, there must be at least 1 blank after the second slash before the word EXEC.

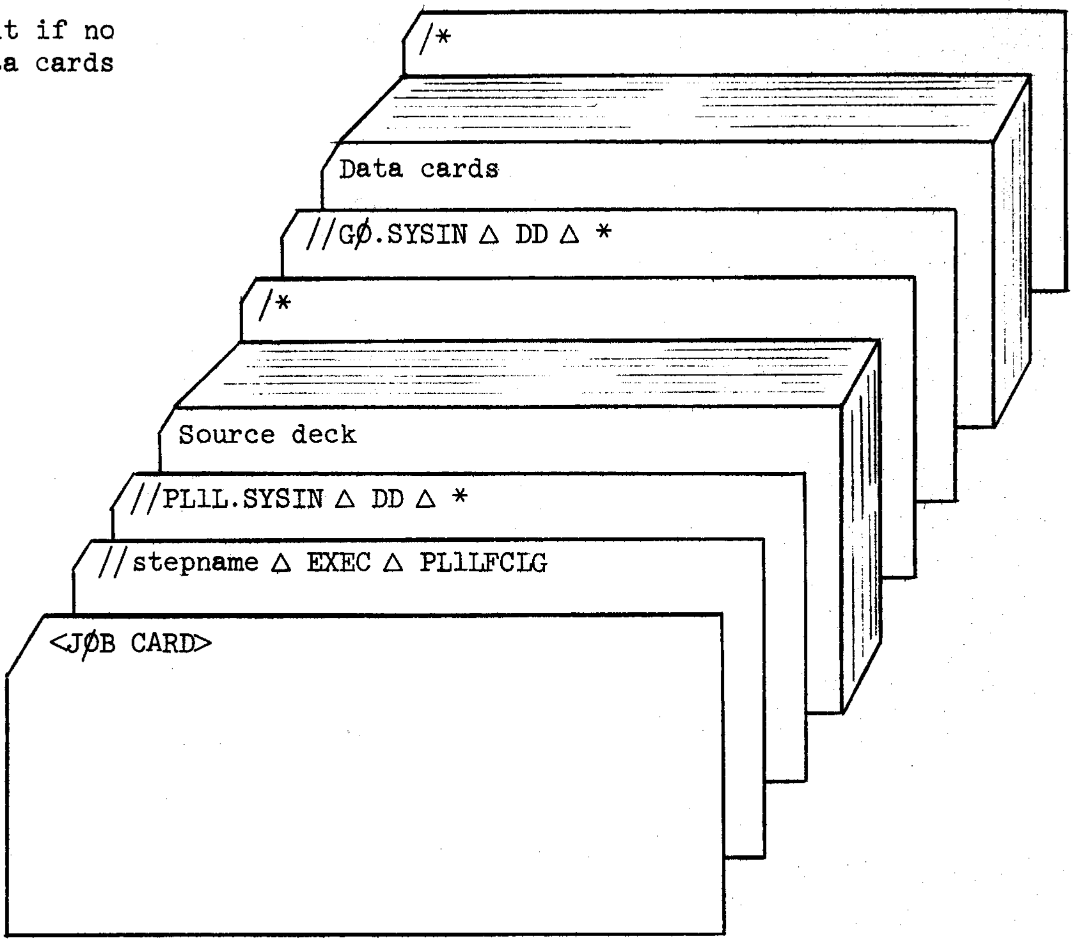
1. Deck setup for a compile only job:



To obtain an object deck from the above, the execute card should be changed as follows: //stepnameΔEXECΔPL1LFC,PARM.PL1L='DECK'

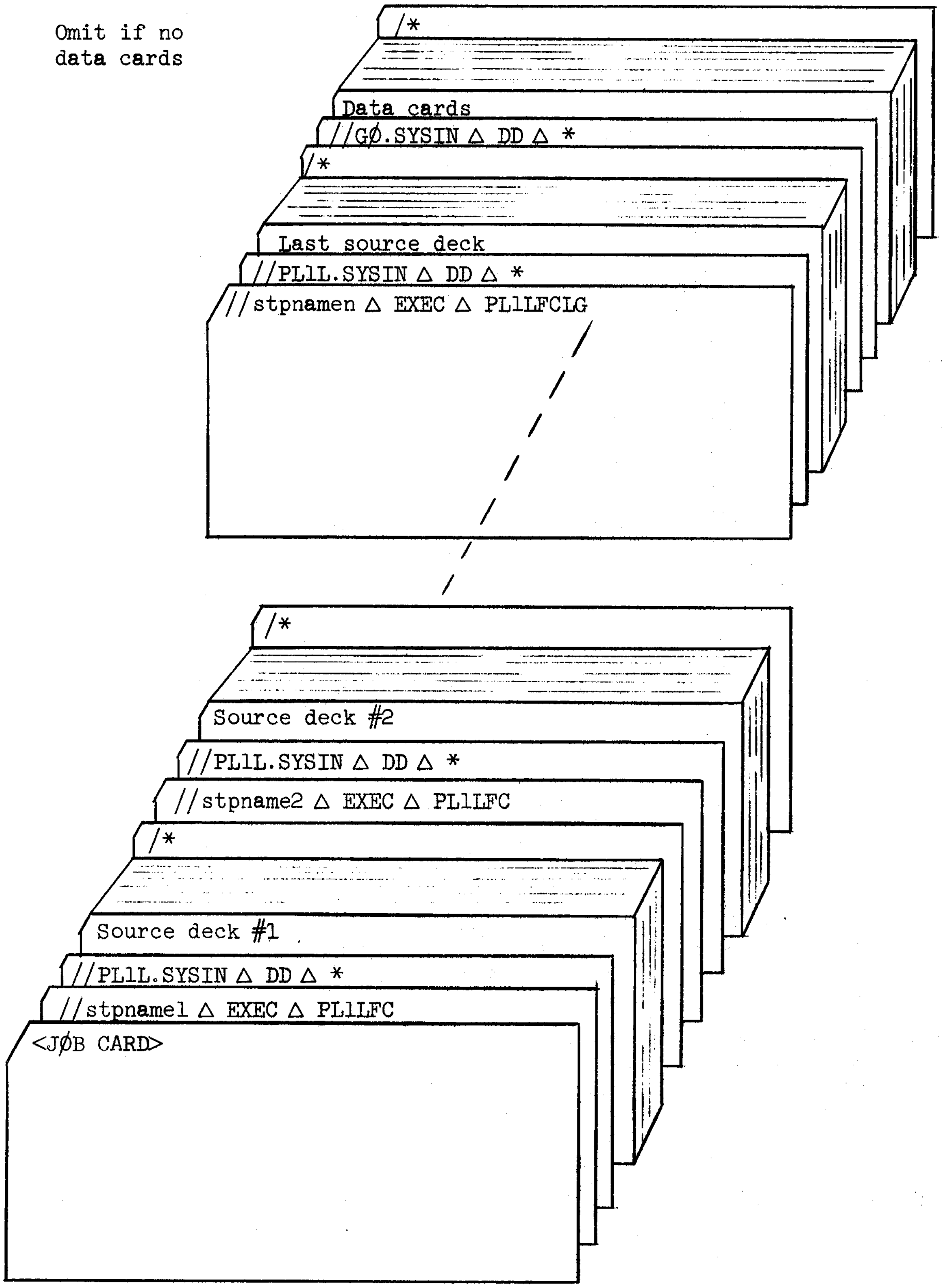
2. Deck setup for a compile, link edit, and go job (single compilation):

Omit if no data cards

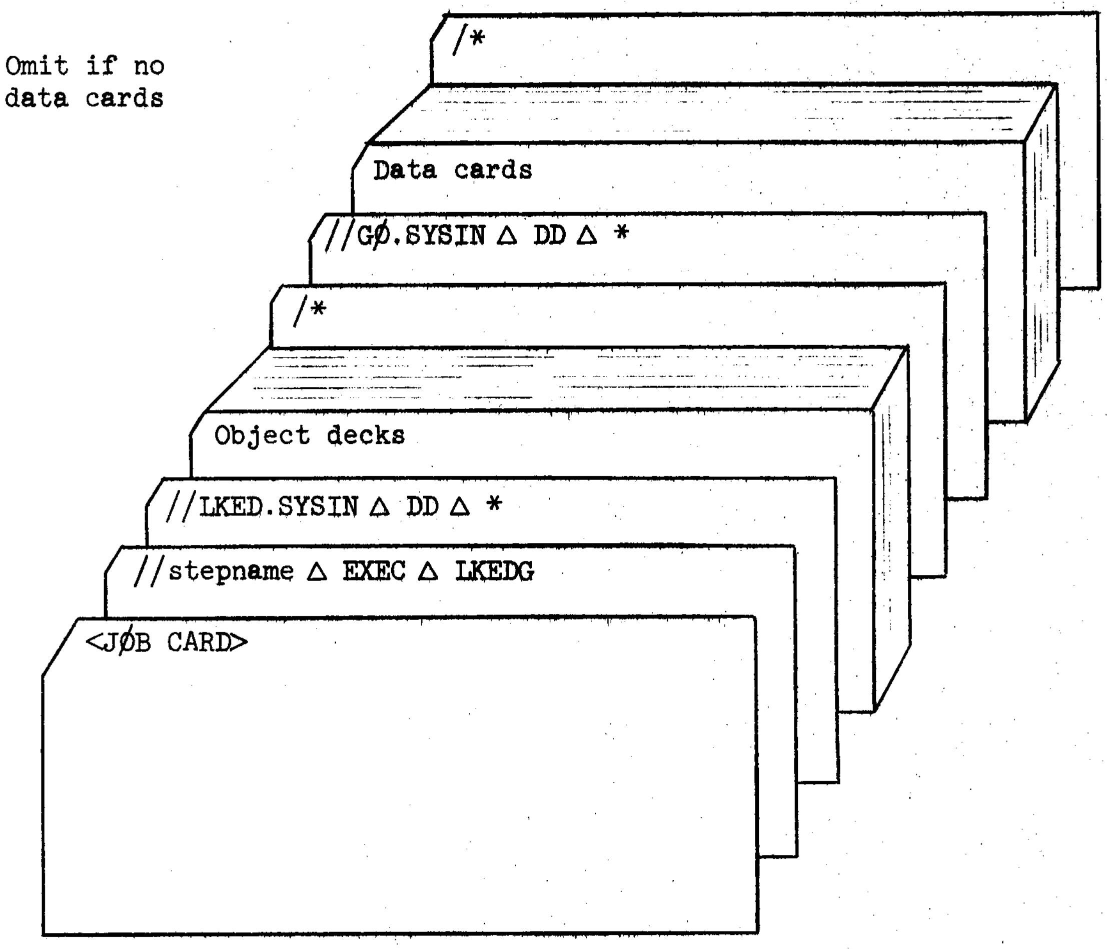


To obtain an object deck from the above, the execute card should be changed as follows: //stepname△EXEC△PL1LFCLG,PARM.PL1L='DECK'

3. Deck setup for a compile, link edit, and go job (multiple compilations):



4. Deck setup for link edit and go job (assuming object decks read from the card reader):



DD cards describing data sets to be used in the users GØ step should be placed in the deck just before the //GØ.SYSIN△DD△* card.

4.2.4 Helpful Hints

The chapter on "Programming Techniques" in the PL/1 (F) Programmer's Guide can be most helpful to the PL/1 user. It contains the section 'Common Errors and Pitfalls' which is an excellent discussion on the snags most likely to trip up a new, or not so new, PL/1 programmer. It also contains a section entitled 'Programming for Increased Efficiency' which gives information for improving both compile and execution times.

4.2.5 Input/Output Capabilities

There are two types of I/O in PL/1 -- stream and record. In stream I/O a data set is regarded as a continuous stream of characters. In record I/O, the data set consists of discrete records, and no conversions occur during transmission.

The devices that may be accessed using stream I/O are: card reader, card punch, printer, disk and tape. The devices which may be used with record I/O are disk and tape.

Stream I/O implies sequential access of data. When record I/O is used, data may be accessed sequentially or randomly. For random access, the device type must be disk.

Programmers who plan to use record I/O should read the chapter, "Managing Data," in the PL/1 Programmer's Guide. It contains information essential to the understanding of F level PL/1 record I/O. A few features of record I/O have not yet been implemented. These are listed in Appendix H of the Programmer's Guide.

The PL/1 catalog procedures provide DD cards only for the printer, card reader (compile and go step), and punch (compile step only). If any other data sets are referenced by a program, the user must supply appropriate DD cards in the GØ step of the job. Information on how to make DD cards can be found in the PL/1 (F) Programmer's Guide in the chapter entitled, "Managing Data", and also in the publication, IBM System/360 Operating System Job Control Language, Form No. C28-6539.

4.3 Assembler Language

4.3.1 Description of the Language

The IBM System/360 Operating System Assembler Language is a symbolic programming language which permits the programmer to use all machine functions as if he were coding machine language.

The assembler program that processes the language translates symbolic instructions into machine-language instructions, assigns storage locations, and performs auxiliary functions necessary to produce an executable machine-language program.

4.3.2 Documentation

1. IBM System/360 Operating System Assembler Language, Form No. C28-6514.

   This publication contains specifications for the Operating System Assembler Language. Part I of the manual describes the assembler language and Part II describes an extension of the assembler language - the macro language - used to define macro-instructions. (145 pages)

2. IBM System/360 Operating System Assembler (F) Programmer's Guide, Form No. C26-3756.

   This publication compliments the IBM System/360 ØS Assembler Language manual. It provides a guide to program assembling, linkage editing, executing, interpreting listings, and assembler programming considerations. Included in Appendix A of the manual is a description of the Assembler Language diagnostic messages. (48 pages)

3. IBM System/360 Principles of Operation, Form No. A22-6821.

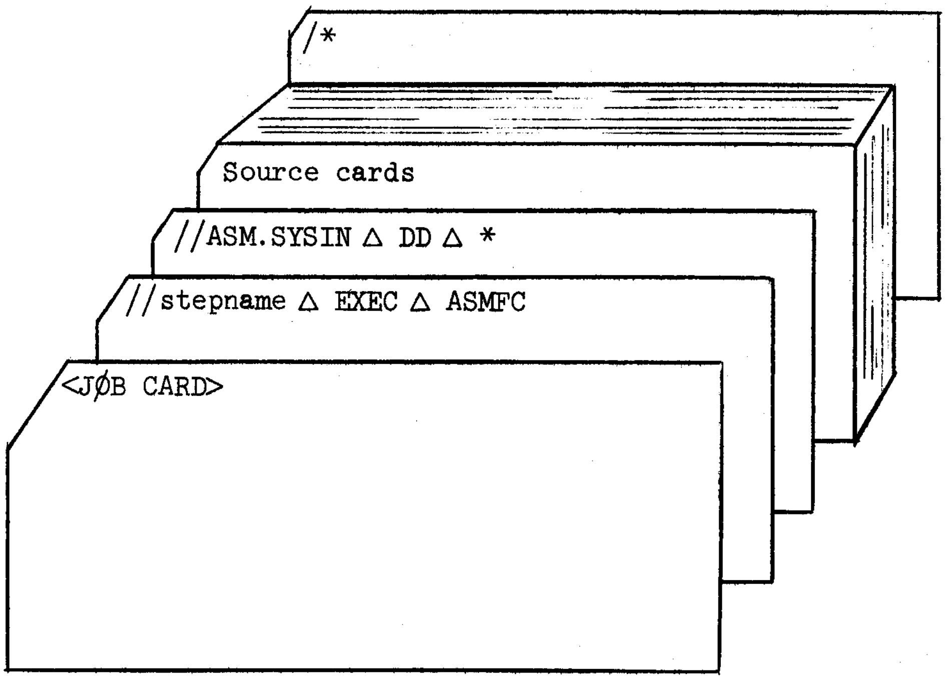
   This publication is the machine reference manual for the IBM System/360. It provides a direct, comprehensive description of the system structure - of the arithmetic, logical branching, status switching, and input/output operations, and of the interruption system. It is the only manual that gives a detailed description of the various machine instructions. (172 pages)

4.3.3 Sample Deck Setups

Note 1: All of the control cards (i.e., // and /*) must be punched starting in column 1. Blanks, at least one, are required where the symbol Δ appears. Blanks must not appear in other fields on the card.

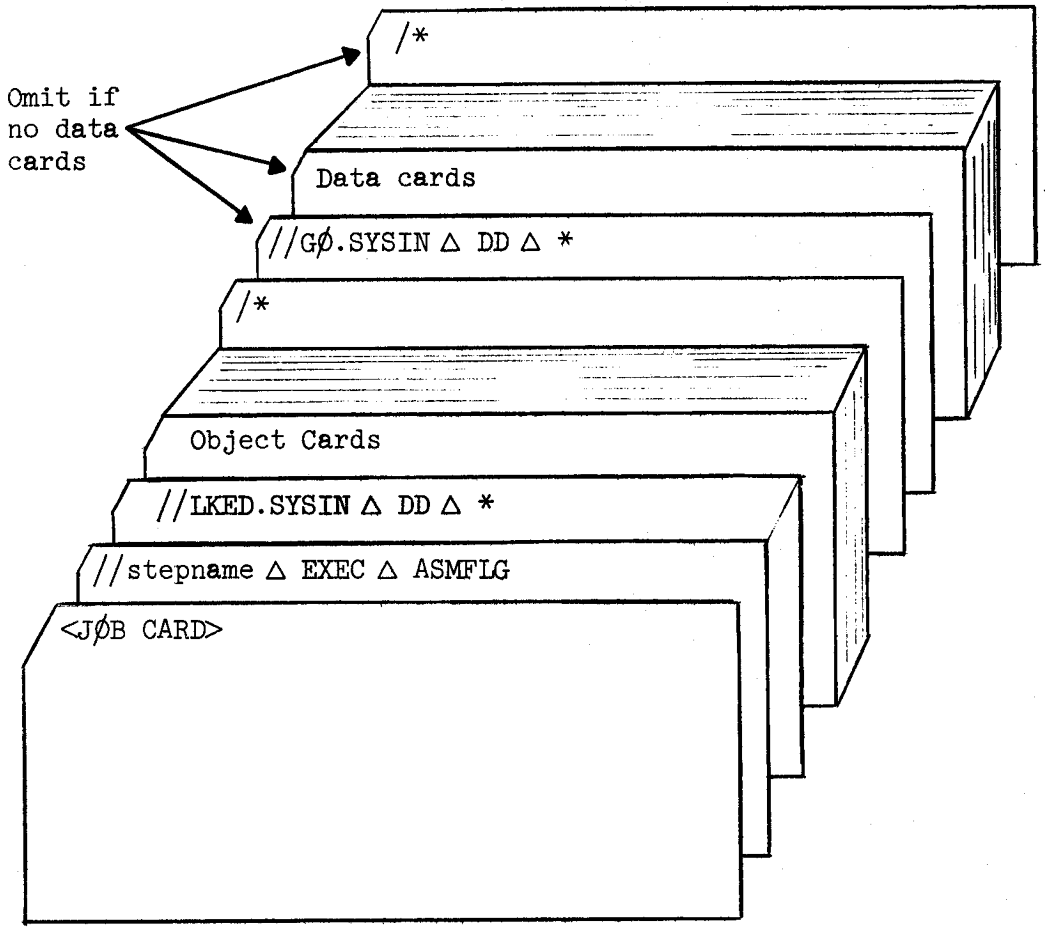
Note 2: The "stepname" information on the EXEC card can be used at the option of the programmer. If it is used, the programmer may use any stepname that he wishes. The first character of this name must immediately follow the slash -- there can be no blanks. If the stepname is omitted, there must be at least 1 blank after the second slash before the word EXEC.

1. To assemble only the following deck setup should be used:

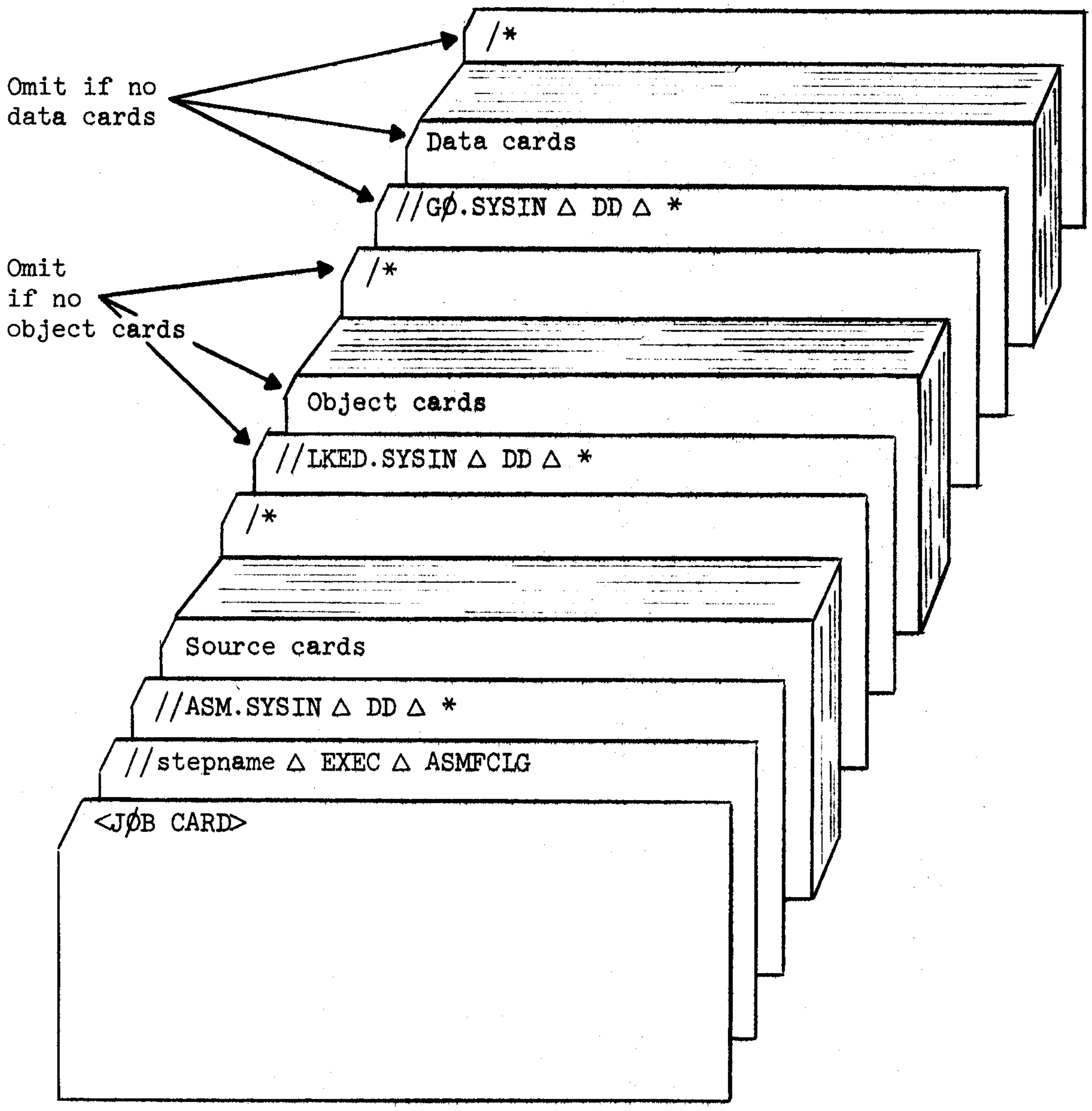


To obtain an object deck from the above, the execute card should be changed as follows: //stepnameΔEXECΔASMFC,PARM.ASM='DECK'

2. To link and go previously assembled decks, the following deck setup should be used:



3. To assemble, link and go the following deck setup should be used:



4.3.4 Helpful Hints

The Operating System/360 Assembler Language offers a greater flexibility to the machine-language programmer than was available in earlier large-scale assembly systems such as FAP and MAP. This is due to the fact that the ØS Assembler, unlike most traditional two-pass assemblers, makes four logically distinct passes over the source program. The third and fourth passes perform the usual two-pass assembly process. The first pass consists of macro editing and global dictionary collection, and the second pass involves macro expansion and conditional assembly. Some of these operations are occasionally included in the first pass of a standard assembler, but with considerably less capability than is found in the ØS Assembler.

The following examples illustrate some situations often found in the use of the Assembler Language. They help to indicate some of the features of the Assembler as well as some of the problems involved in its use.

1. When using the CSECT assembler instruction, you may reference symbols from a control section other than the one in which it is defined. This means that (1) an individual symbol can be used only once in an assembly step, and (2) because of the base-displacement addressing scheme used in the System/360, it apparently may be possible to address a symbol, whereas under later relocation by the linkage editor the address will be incorrect. This occurs because a USING instruction which applies in one control section will apply to code which references symbols in that control section from another, even though no provision may have been made for loading an appropriate base register before the reference. In general, the capability of assembling multiple control sections must be used with care. This situation is somewhat alleviated since the Assembler does not do multiple assemblies -- all cards following the END card are not processed as a new assembly.

2. One feature of the four-pass assembly is that some attributes of certain symbols can be used during the conditional assembly

pass to determine what quantities are to be generated for conventional assembly during the last two passes. The general rule is that a symbol whose type, length, integer, and scale attributes are defined without using ordinary or variable symbols can have those attributes used during conditional assembly. A notable exception to this rule is the EQU instruction. It might seem that the statement A EQU 5 should define the attributes of A. It must be remembered, however, that this equivalence does not become operative until the last two passes, so that all symbols defined in EQU statements are undefined (have type attribute U) during pass two. This apparent discrepancy is easily circumvented by the use of a variable symbol, as in the statement &A SETA 5. The programmer will find that, in many situations where the use of ordinary symbols is either cumbersome or forbidden, variable symbols provide the necessary capability.

3. The variety of methods available for testing macro-instruction operands (the actual parameters) is quite extensive. Such operands may be treated as character strings, symbols, or lists of either. Their properties may be tested in a number of ways, such as character comparison, symbol equality, and so on, giving considerable flexibility in the design and use of macro-instructions.

4.3.5 Helpful Hints in Other Manuals

Since one of the major uses of machine language programs will be writing special purpose routines to be called from higher-level languages, the prospective programmer of such codes should become familiar with the appropriate sections of manuals which describe linkage conventions in the desired language. A recommended starting point is the first part of the publication IBM System/360 Operating System Supervisor and Data Management Services, Form No. C28-6646. The first sections contain a description of linkage conventions commonly used in ØS/360. For FØRTRAN programmers, a brief example of the use of assembly-language routines with FØRTRAN programs is given in the IBM System/360 Operating System FORTRAN IV (H) Programmer's Guide (Form No. C28-6602) and in the IBM System/360 Operating System FORTRAN IV (G) Programmer's Guide (Form No. C28-6639). For programmers using PL/1, a description of the linkage and parameter-passing conventions used is given in the IBM System/360 Operating System PL/1 (F) Programmer's Guide, Form No. C28-6594. Additional information of possible interest will be found in the IBM System/360 Operating System PL/1 Subroutine Library-PLM, Form No. Y28-6801. The PL/1 conventions are sufficiently complex so that most programmers will prefer not to use assembler-language programming except in extreme cases. It should be noted that in the case of routines with only scalar arguments (no arrays, structures, or strings) the PL/1 linkage and argument-passing conventions are the same as the regular ØS/360 conventions. If it is desired to maintain multiple entry points, it may be necessary to go to extra lengths.

Routines which will later be used as part of a much larger program are occasionally difficult to debug because no I/O is to be performed for many levels of routine calls. In such a situation the PRINTØUT macro-instruction is often helpful, since virtually any quantity may be printed in an easily readable format, without worrying about destroying registers or the necessity for providing linkage registers. Similarly, a simplified form of card input may be performed using the READCARD macro-instruction. Both macros use the FØRTRAN I/Ø package to perform the necessary formatting and conversion so that extra space will be required in the program's load module. Once the program has been debugged and the PRINTØUT and READCARD statements removed, the extra routines will not be required.

5. DEBUGGING

5.1 Introduction

The Computation Center realizes the need to present to the user some of the available resources and methods for debugging. The term debugging is meant here in the most general sense of trying to determine why a particular job did not execute. At the present, this section will briefly describe some of the resources available to the user for debugging, some of the common errors made by 360 programmers, and some helpful hints for avoiding potential errors. In a future update to the User's Manual, this section will be elaborated to include more complex debugging procedures.

OS/360 provides a number of debugging aids as follows:

1. A message reports an exceptional condition such as a programming or processing error. Messages appear in the output, informing the programmer of what is wrong in the program.

2. A completion code is given to indicate why a task was abnormally terminated.

3. A storage dump following abnormal termination is either (1) an indicative dump, which gives control information including the completion code or (2) an abnormal termination dump (ABDUMP), which includes control information as well as the contents of main storage at the time the job was abnormally terminated.

4. In addition, a listing of object code produced and a map of modules loaded by the linkage editor can be very useful to the programmer in debugging his code. Listings and maps can be obtained by specifying the LIST and MAP options in the EXEC statement for the compiler or catalogued procedure involved. (Refer to Section 3 of this manual) Examples of formats for listings and maps are given in the "System Output" section of the appropriate Programmer's Guides.

5.2 Documentation

1. IBM System/360 Operating System Messages, Completion Codes, and Storage Dumps, Form No. C28-6631.

   This document lists and explains the messages and completion codes produced by all IBM- supplied components of the operating system. It also describes the format of storage dumps. It is the basic reference for all debugging. Format of the diagnostic messages is explained in the Introduction. Messages are arranged alphabetically by their codes and are grouped according to the operating system component which issues them. For example, all messages from the FORTRAN IV (H) compiler start with the code IEK, all messages from the PL/1 (F) compiler with the code IEM. If the message text provided in the program output is not self-explanatory, consult this manual for a more detailed explanation.

2. The Programmer's Guides for the language processor of interest to the user contains a list of messages produced by the processor in one of the Appendices.

FORTRAN H	IBM System/360 Operating System FORTRAN IV (H) Programmer's Guide, C28-6602. Appendix D
FORTRAN G	IBM System/360 Operating System FORTRAN IV (G) Programmer's Guide, C28-6639. Appendix D
Assembler	IBM System/360 Operating System--Assembler Programmer's Guide, C26-3756. Appendix A
PL/1	IBM System/360 Operating System PL/1 (F) Programmer's Guide, C28-6594. Appendix G

5.3 Notes

1. Programmers using FØRTRAN G should check Appendix F of the Programmer's Guide (C28-6639) for a list of the debugging facilities available.

2. MSGLEVEL=1 should always be specified in the JØB statement. This parameter causes all job control statements to be printed as part of the user's output, including the entire contents of the catalogued procedure(s) used.

3. Users should double check all job control statements to make sure that the required blanks are really there. For example,

```
//FØRT.SYSIN  DD*
```

   gives the error message IEF121I CANNOT IDENTIFY VERB.
   The correct form of the statement would be `//FØRT.SYSIN  DD  *`

4. Don't try to WRITE on unit 5, or to READ unit 6.

5. When using continuation cards for job control statements, the rules specified in Section 3.2.6 must be rigidly adherred to. In particular, failure to include the comma after a parameter before going to the next card, will cause that card to be read as a comment, and all parameters on that card will be lost. For example:

```
//FIRSTGØ  EXEC  FØRTHCLG,PARM.FØRT='DECK'                X
//                PARM.LKED='XREF'
```

   The above statement would give a deck, but the cross-reference list specified on the second card would be ignored. The correct set-up for this statement should be

```
//FIRSTGØ  EXEC  FØRTHCLG,PARM.FØRT='DECK',               X
//                PARM.LKED='XREF'
```

6. Users should be aware that on the 360/67 core is not set to zero prior to job execution. To 7090 FORTRAN users, this means that uninitialized variables have unpredictable contents. If these variables are used as subscripts, program termination can result. The following program and the actual output show the danger of such a situation.

These statements

```
ISN  0002          REAL A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z
ISN  0003          B = 1
ISN  0004          A = B+C+D+E+F+G+H+I+J+K+L+M+N+O+P+Q+R+S+T+U+V+W+X+Y+Z
ISN  0005          PRINT 100, A
ISN  0006     100  FORMAT(' ',' THE VALUE OF A IS', E20.7)
ISN  0007          RETURN
ISN  0008          END
```

give this result

```
THE VALUE OF A IS       0.1007571E 55
```

6. LIBRARIES

6.1 Introduction

As one of its responsibilities to the users, the Computation Center has undertaken the development and maintenance of program libraries in support of the user's computational needs. These libraries are divided into two general categories; i.e., the applications program library classification and the subprogram library classification.

The first of these will contain complete programs and their usage will only require the users to furnish the appropriate job control language set-up, any program control or option specifications required, and the data that he wishes to process.

The second type of library will contain all of those programs which require the additional requirement of a driver or calling program. A subprogram generally performs only one task or a set of similar tasks and is used by the programmer as one of the "building blocks" within his program. A subprogram can either be a function subprogram or a subroutine subprogram. The function subprogram returns only the value of the function to the driver program while a subroutine subprogram can return the values of more than one argument.

6.1.1 Sources of Programs

In order to serve the current and future expanding needs of the users, the libraries will be in a continuous state of modification and expansion. The initial structure of the libraries was dictated by the requirements of the Center's past and current users. The Computation Center is also aware that deficiencies exist within the libraries. Consequently, provisions have been made for the users to submit any of their programs which receive frequent use or are standard operational routines for inclusion within the libraries. The programming standards and conventions are specified within this section of the Users Manual. Any program that is submitted must conform to these standards and be certified by the Center before it can be included within one of the libraries. Anyone wishing to submit a program may obtain a write-up form from the Users Services office, Room 156, Polya Hall.

The user may also request that a program from another facility be considered for inclusion in the Stanford Libraries. The aforementioned qualifiers for any such program must also be met.

In the event that a program does not exist or cannot be located that will perform a required task or set of tasks for the user, a written request can be made to the Computation Center specifying the need for such a program. If methodology is significant, it should be explicitly stated. All such requests will be reviewed by the Computation Center and the user will be notified of the action taken.

6.1.2 Program Standards and Conventions

Introduction

In general, a program should be written in such a manner that anyone can read it, understand what it does, and make modifications to it with a minimum of effort. Even straightforward coding should contain profuse comments and tricky coding should be explained in even greater detail. As experience has shown, it is often easier to rewrite a whole program than to make a small modification or correction to a program written by someone else. This duplication of effort must be eliminated for library programs -- the solution is to make the programs fully self-explanatory. The standards and conventions listed below will aid in accomplishing this goal.

When preparing a program for inclusion in the Program Library, keep three things in mind. First, another individual who may wish to use the program will probably be thoroughly unfamiliar with it. Such an individual will want to be able to use the program with a minimum of study. He will want to know how to incorporate the program into his deck and what sort of accuracy, error messages, etc., he can expect. Second, an individual might wish to modify the program slightly for a specific use. In order for the program to be most useful to him it must be thoroughly documented. Third, individuals using the program library expect programs to be computationally sound and efficient. Programs included in the Program Library must utilize good programming practices.

Comments

Comments are the most important part of any program which is to be used by someone other than the original programmer (and even for your own programs if you want to read them at some later date). Remember, the listing of the program is generally the only reliable source. Comments should be profuse and not cryptic. They should be placed at least in the following places in a higher level language such as FØRTRAN:

1. The first few cards in any program or subprogram deck should

be a comment giving the following information:

a. Title and name of the routine.

b. Name of programmer (affiliation).

c. The date.

d. The modification level.

For example, the first three cards of a deck could be:

```
C    LINEAR SYSTEM SUBRØUTINE, DECMP1
C    DECØMPØSE A INTØ LU
C    JØHN P. JØNES (SCC) 2 MARCH 1967, MODIFICATION 3
```

Any programmer who subsequently makes a modification to the source deck should <u>add</u> a comment giving his name and the date and a brief description of the modification. This provides an easy mechanism to keep historical records on library programs.

2. Comments at the beginning of each program or subprogram. The comment should outline the purpose and method of the subsequent code and describe all parameters.
3. Comments at the beginning of each block or segment of code that performs a particular task. For example, each non-trivial DO-loop should be commented.
4. Comments at each branch point in a program. The reason for taking each branch and what will be done after branching there should be explained.
5. Wherever else a comment can be placed to make <u>absolutely clear</u> what a particular statement is doing. The program with too many comments has not yet been written.
6. At each CALL or function reference (except standard transcendental functions), there should be an indication of what actions are performed by the subroutine or function.

When coding routines using an assembly language the above rules should be applied in addition to the following:

1. Put a comment on every card.
2. Any tricky coding should be <u>fully</u> commented.
3. To reiterate what is specified above, be sure to put comments at the beginning of each program or subprogram, at the beginning

of each major segment of code, and at all branch points to make the program completely readable.

Variables

1. Declarations: All variables and arguments must be declared at the head of the subprogram, even when using compilers that declare by default.
2. Identifiers: The identifiers (names) given to variables should be mnemonic with as much meaning as possible. The identifiers used in subprogram definitions should agree with those used on the write-up. Some examples of mnemonic variable names are: an increment in the X direction is usually identified as DX; in an eigenvalue procedure the matrix is usually denoted by A and an eigenvalue would be LAMBDA. In general, use long (6 characters for FØRTRAN) rather than short names. A good practice is to use the normal English names for the quantity.

Names of Subroutines

Names of subprograms should be AAAAAd where A is an alpha character and d is a numeric digit. This allows updating a routine by adding one to the digit 'd'. Thus two (or more) versions of a routine can be maintained in the library simultaneously without confusion.

Parameter Lists Ordering

The ordering of parameters in a subprogram call should be:

Input parameters first.

Output parameters second.

Label parameters last.

For example: A matrix product routine (MPROD) could be called as

```
MPROD(N, A, B, C)
```

where A and B are square matrices of dimension N and their product is stored in the matrix C . Thus, N, A, and B are input parameters and C is the output parameter.

Indentation

Indentation should be used to indicate the scope of various constructs in a program. This can be applied to FØRTRAN code as well as other languages. For example, consider the following nonsense program segment:

```
        DØ 10 I = 1, N
          X = X + DX
          DØ 9 J = 1, N
            Y = Y + DY
            Z = Z + DZ
 9          ANS(I,J)=Y**4 + Z**4
          B(I) = C(I) * X
 10       CØNTINUE
 11     CØNTINUE
```

I/O in Subprograms

Avoid any input or output within a subprogram. Instead, pass the information through the parameters or arguments.

Use label parameters to indicate errors or non-standard occurences within subprograms. For example, instead of printing an error message in the subprogram return to the calling program via a label parameter. This allows the calling program to decide what action to take in the case of an error.

Register Usage in 360 Subprograms

For register usage conventions refer to the following IBM documents. For use with FØRTRAN:

> IBM System/360 Operating System FORTRAN IV (H) Programmer's Guide, (Form No. C28-6602).

For use with PL/1:

> IBM System/360 Operating System PL/1 (F) Programmer's Guide, (Form No. C28-6594).

Material to be included for each program or subprogram for the Program Library

1. Completed write-up form.
2. The source deck.
3. A binary deck (if applicable).
4. A listing of the source deck.
5. A driver program which does as many test cases as necessary to completely check the operation of the program.
6. A listing of a successful run with the driver program with enough comments that a person can tell whether the printed results are what should have been produced.

6.2 Stanford Extrinsic Program Library

6.2.1 Introduction

The Stanford Extrinsic Program Library is a library of subprograms developed and written by the Computation Center. These subprograms may supplement as well as overlap those of the Scientific Subroutine Package. The intent of those programs which duplicate any tasks of the Scientific Subroutine Package is that of providing a more efficient and more current state-of-the-art program.

All subprograms in the Extrinsic Program Library are entered by the standard FORTRAN calling statements. The user is reminded that he must furnish, as part of his calling program, all I/O and other operations necessary for the total solution of his problem.

Any subprogram contained within the Stanford Extrinsic Program Library has gone through a process of certification. This process implies the successful completion of a sufficient number of test runs which, in general, utilize all the significant options and controls of the subprogram. Within the structure of these test cases, the subprogram also has demonstrated no syntactical or logical errors. The final step of the certification procedure requires that the subprogram be tested by a programmer other than the source programmer. This testing includes such qualifiers as:

1. The documentation explaining the subprogram is adequate, well written and descriptive.
2. The subprogram performs as advertised in the documentation.

6.2.2 Obtaining Library Programs and Documentation

All of the announced subprograms exist on the 2314 disk in source form. The user can obtain these programs <u>directly</u> by using the utility program IEBPTPCH. Either source decks and/or source listings are available. Details on the use of IEBPTPCH and the job control language set-up for obtaining source decks and/or listings can be found in Section 3.4, 'Utilities', of this manual.

Write-ups for these subprograms are available from the Systems Documentation Office, Room 185, Pine Hall, Ext. 4877.

Copies of any of the programs contained in the Stanford Computation Center Program Library may also be ordered through the Campus Facility User Services Group, Room 156, Polya Hall. There is a charge for this service as follows.

- A service charge of $50.00 will be levied for each order regardless of the number of routines requested.
- In addition, a charge of $1.00 per 1000 card images will be levied for output on tape.
- If the buyer fails to provide his own tape, we will supply tape at the rate of $27.00 per 2400 foot reel.
- If card output is requested, a charge of $5.00 for the first 1000 cards plus $1.25 per each 1000 additional cards, plus actual postage, will be levied, in addition to the $50.00 service charge.

Persons wishing more information on the programs available through this service should contact the Campus Facility User Services Group, Room 156, Polya Hall, Stanford Computation Center, Stanford, California.

6.2.3 Available Programs

The following Extrinsic Library subprograms are available. The member name(s) are also given for each program.

Program Number	Member Name(s)	Title and Description
C 001	GCD1	Greatest Common Divisor - This function subprogram finds the absolute value of the greatest common divisor of two integer arguments using Euclid's algorithm.
C 002	NXP1	Next Prime Number Routine - NXP1 finds the next prime number larger than the integer argument supplied by the calling program.
C 003	VPR2	Double Precision Inner Product Accumulation - This routine forms the double-long product of two long floating-point arguments and adds it to a double-long sum, and provides a simple indication of exponent overflow and underflow.
C 004	DPR1R DPR1U	Double Precision Product Accumulation with Optional Rounding - This subroutine subprogram computes the long precision of two long precision floating-point arguments with optional rounding and provides a simple indication of exponent overflow and underflow.
C 005	DPR2	Product Accumulation for Short and Long Floating-Point Operands - This subroutine subprogram computes the long precision product of a short and a long precision floating-point number; the product replaces the long precision operand, and a simple indication is given of exponent underflow and overflow.
C 007	EDNUM	Convert Integer Variables to an Edited Print Format - This program converts integer variables to EBCDIC (alphanumeric) characters, suppressing leading zeroes and inserting commas and dollar signs or asterisks, as desired.
C 008	MARQ1	Least Squares Estimation of Nonlinear Parameters (Marquardt, D. W.) - The Fortran subroutine can be used to (1) estimate nonlinear parameters in a statistical model, or (2) solve a system of simultaneous nonlinear equations by minimizing $\sum_{i=1}^{k} f_i^2$

Revised May 1968

Program Number	Member Name(s)	Title and Description
C 008 (continued)		with the independent variables as parameters. The subroutine uses the first partial derivatives of the function(s) which are evaluated by a function subprogram which is a parameter to MARQ1 .
C 009	MULLR1	Muller's Method - Subroutine MULLR1 finds complex roots of an arbitrary function of one complex variable.
C 011	QUADS1	Integration of a Real Function of One Variable by Classical Quadrature - This function subprogram approximates the integral of a real function of one variable by a given classical quadrature rule: $\int_{LOWER}^{UPPER} FUNCT(x)dx \quad \sum_{j=1}^{N} w_j \; FUNCT(t_j)$ The kind of quadrature rule (e.g., Gauss, Hermite, Newton-Cotes, etc.), the weights and abscissas are supplied by the User in the simplest form.
C 013	QUADM1	Multiple Integration of a Real Function of N variables by Product Rule Quadrature - The function of subprogram QUADM1 approximates the multiple integral of the form $\int_{L_1}^{U_1} F_1(x_1) \quad \int_{L_2}^{U_2} F_2(x_1, x_2, \ldots$ $\int_{L_N(x_1, x_2, \ldots, x_{N-1})}^{U_N(x_1, x_2, \ldots, x_{N-1})} F_N(x_1, x_2, \ldots, x_N)dx_N \ldots dx_2 dx_1$ by repeating the quadrature rule supplied a specified number of times for each integration (i.e., producing a product quadrature rule). The lower and upper limit functions and the integrands are supplied by REAL FUNCTION subprograms.

Program Number	Member Name(s)	Title and Description
C 014	QUADS3	Numerical Integration of Adaptive Quadrature - This real function subprogram approximates the integral of the function FUNCT between the limits of LOWER and UPPER by applying Simpson's rule and Romberg correction to various length subintervals as dictated by the integrand and the tolerance EPSLØN .
C 015	DFEQS1	Solve a System of First Order Ordinary Differential Equations by the Kutta-Merson Method - This subroutine subprogram uses the single step method of Kutta-Merson to approximate the solution of a system of n first order differential equations, $\frac{dY}{dX} = F(X,Y)$. Automatic error control (step size control) is available if requested.
C 016	INTRP1	Newton's Forward and Backward Interpolation for Equally Spaced Points - Given a table of values of a function F(x) at a set of equally spaced values of x, this procedure uses Newton's Forward and Backward Interpolation formulas to determine the value of F(x) for any value of x in the allowable range.
C 017	BESSL1	Bessel's Interpolation with Equally Spaced Points - Given a table of values of a function F(x) at a set of equally spaced values of x, this procedure uses Bessel's interpolation formula to determine the value of F(x) for any value of x in the range.
C 018	HRMIT1	Hermite Interpolation - This procedure evaluates, at a given abscissa, a (2N+1)th-degree Hermite polynomial passing through N+1 points. The first derivative of the polynomial at each of the N+1 points must be given to the procedure.
C 019	LAGRN1	Lagrange Interpolation - This procedure evaluates, at a single abscissa, an Nth-degree Lagrange polynomial passing through N+1 points.
C 021	DCOMP1 LSSQS1 SOLV1	Linear Least Squares Problem Solver - These subroutines solve the linear least squares problem $\| A\underline{x} - \underline{b} \|_2 = \text{minimum}$,

Revised May 1968

Program Number	Member Name(s)	Title and Description
C 021 (continued)		where $\lVert \cdot \rVert_2$ indicates the euclidean vector norm and A is an mxn $(m \geq n)$ real matrix of rank n . A matrix decomposition based on orthogonal Householder transformation is used rather than solving the normal equations $$A^T A\underline{x} = A^T \underline{b} .$$ Several vectors, $\underline{b}$, may be given for solution at once to increase efficiency.
C 022	DECMP1 DETER1 IMPRV1 INVRT1 LINSY1 SOLVE1	Subroutine Package for Linear Systems Solution, Matrix Inversion and Determinants - By decomposing the given square matrix A into two triangular matrices L and U such that LU = A, these subroutines supply approximate solutions to the problems: 1. find the vector $\underline{x}$ such that $A\underline{x} = b$, where $\underline{b}$ is a given vector; 2. find the matrix V such that AV = 1, where 1 is the identity matrix; 3. find determinant (A) . Problem 1 can be solved efficiently for many right-hand sides $\underline{b}$ after the initial decomposition of A . Iterative improvement of each solution $\underline{x}$ may be specified. Matrix singularity and near singularity are reported; an accuracy measure is supplied following iterative improvement and exponent overflow and underflow during the determinant calculation is conveniently handled.
C 023	AXIS1 ENDP1 FACTR1 LINE1 NUMBR1 OFFST1 PLOTS1 PLOT1 SCALE1 STRTP1 SYMBL1 WHERE1	OS/360 Plotting on the CALCOMP Plotter - These twelve subroutine subprograms perform a general purpose set of plotting tasks. They use the 570 CALCOMP System to output the resultant plot(s).

Revised May 1968

Program Number	Member Name(s)	Title and Description
C 024	CURVE1 GRAPH1 POINT1 TITLE1	Graphing of One Variable Functions (Curve Plotting) on the CALCOMP Plotter - These subroutines use the basic CALCOMP plot subroutines to draw curves and label graphs of points supplied as (X,Y) pairs. The points for each curve are collected by the subroutine POINT1 ; the subroutine CURVE1 is called to give the style and symbol to be used for each curve to be on the next graph. Subroutine TITLE1 may be called to place a heading on the next graph. A call to subroutine GRAPH1 labels the axis, if requested, scales and plots all curves as specified, then moves the paper to prepare for the next graph.
C 025	LØGAX1 LIMIT1	Logarithmic Axis Plotting for the CALCOMP Plotter - These subroutines can be used to plot a scaled axis on the CALCOMP Plotter with logarithmically placed tic marks.
C 026	HEX2	Hexadecimal to EBCDIC Conversion Routine for Extended Arguments - This routine converts a string of hexadecimal digits into a string of EBCDIC characters which represent the hex digits.
C 027	SHF1LA SHF1LC SHF1LL SHF1RA SHF1RC SHF1RL	Shift Routines for Fullword Operands - These function subprograms provide the Fortran programmer with the capability of performing arithmetic, logical and circular shifts of any number of binary positions on any fullword operands.
C 028	SHF2L SHF2R	Shift Routines for Doubleword Operands - These function subprograms provide the Fortran programmer with the ability to perform logical shifting operations on doubleword operands by an arbitrary number of binary positions. It is closed, re-entrant, double-precision type of function subprograms, standard OS/360 entry.
C 029	LGO1AN LGO1CM LGO1OR LGO1XR	Logical Operations of Fullword Operands - This function subprogram provides the logical operations AND, OR, EXCLUSIVE OR, and ONE'S COMPLEMENT (BITWISE COMPLEMENT) on fullword operands.
C 030	MVC1	Fortran Move-a-byte Routine - This subroutine subprogram allows the Fortran programmer to move any byte in storage to any other byte.
C 031	IRND1L IRND1S RND1LS	Rounding Functions for Long and Short Precision Arguments - These subprograms provide the three functions:

Program Number	Member Name(s)	Title and Description
C 031 (continued)		1. round a long precision floating-point number to short precision, 2. round a short precision floating-point number to the nearest integer, and 3. round a long precision floating-point number to the nearest integer.
C 033	ABD1	Abnormal-End Program Terminator - This short program allows the Fortran programmer to terminate his program with an ABEND macro-instruction, and therefore to obtain a dump of storage at the time of the call.
C 034	FIO99X FIO999	Fortran Read-Write Simulation and Internal I/O Buffer Control - This routine allows a Fortran programmer to perform I/O conversions to and from specified buffer areas in memory without an accompanying I/O operation on a physical device.
C 035	JREPRNTO JREPRNTU PRINTOUT	Printout Macro-instruction - The PRINTOUT macro-instruction provides the machine language programmer with a simple means of obtaining debugging, diagnostic, and other useful output with an absolute minimum of effort.
C 036	WTOPARM	PARM Field Type Out Routine - WTOPARM is a stand alone program which will display on the typewriter the contents of the PARM field from the EXEC statement of the Job Control Language.
C 037	JREREADC READCARD	Readcard Macro-instruction - READCARD provides a simple means of reading 80 bytes from a data card into a buffer in the Users' program.
C 039	CLOCK1	Date and Time Fortran Function - This integer type function subprogram will supply the year, day, and time of day in three formats (HHMMSSth, hundredths of a second, and timer units).
C 040	RAN1 RAN1A	Uniform Random Number Generator - This subroutine generates a sequence of uniformly distributed random numbers, either integer (on the interval $(0, 2^{31}-1)$) or real (on the interval (0, 1)).
C 041	TRC1 TRC1T TRC1S	Fortran Run-Time Error Trace and Diagnostic Routine - This routine provides the Fortran programmer with diagnostic information at the time certain errors occur, in a format which simplifies program debugging and error tracing.

Program Number	Member Name(s)	Title and Description
C 043	A2INT	Alpha to Integer Conversion - This routine converts a string of one to four byte characters, read in under an alphanumeric format, to an integer.
C 044	EPILOGUE JREPREPO PROLOGUE	Prologue and Epilogue Macro-instructions - These macro-instructions will considerably simplify program debugging for beginning Assembler language programmers; useful diagnostic information is provided for all program interruptions.
C 045	SYMVV1	Eigenvalue and Eigenvectors of a Symmetric Real Matrix by the QR Method - This subroutine finds all the eigenvalues and eigenvectors of a symmetric real matrix by Householder Tridiagonalization, and the QR method.
C 046	SYMV1	Eigenvalues of a Symmetric Real Matrix by the QR Method - This subroutine finds all the eigenvalues of a symmetric real matrix by Householder Tridiagonalization, followed by a stable square-root free version of the QR method.
C 047	DTSHF1 QRMTH1 SPCTM1 SUBDG1	Eigenvalues of a Nonsymmetric Real Matrix by the QR Method - This subroutine finds all the eigenvalues of an arbitrary real matrix. The matrix is reduced to Hessenberg form; then the eigenvalues are found by the QR method. If the matrix is known to be symmetric, use subroutine SYMV1 (Library Program No. C 046).
C 048	PRINTLINE	PRINTLIN Macro-instruction for Simplified Single or Multiple-line Output from Assembler Language Programs - This macro-instruction allows a programmer to write preformulated character strings on a printer or other output medium with a minimum of worry about details of data management and Job Control Language.

6.3 Scientific Subroutine Package

6.3.1 Introduction

The Scientific Subroutine Package is a manufacturers supplied item. It is a collection of over 200 FORTRAN subroutines divided into two groups: statistics and mathematics. The subroutines are input/output-free computational building blocks that can be combined with a user's input, output, or computational routines to meet his needs.

All subroutines in this package are entered by means of the standard FORTRAN CALL statement. Since the SSP is available in the system, no external decks or control cards are required.

These subroutines are purely computational in nature and do not contain any references to I/O devices. The user must furnish, as part of his calling program, all I/O and other operations necessary for the total solution of his problem. In addition, the user must define all matrices to be operated on by SSP subroutines as well as those matrices utilized in his program. These matrices must conform to the requirements set forth in the SSP Programmers Manual. All of the normal rules for FORTRAN concerning subroutines must be observed.

6.3.2 Documentation

The subroutines contained in the package are described in the System/360 Scientific Subroutine Package - Programmers Manual (Form No. H20-0205). Personal copies of this manual can be obtained from the Systems Documentation Office, Room 185, Pine Hall.

6.3.3 Available Programs

The following SSP programs are available for use.

Data Screening

TALLY	totals, means, standard deviations, minimums, maximums
BOUND	selection of observations within bounds
SUBST	subset selection from observation matrix
ABSNT	detection of missing data
TAB1	tabulation of data (1 variable)
TAB2	tabulation of data (2 variables)
SUBMX	build subset matrix

Elementary Statistics

MOMEN	first four moments
TTEST	tests on population means

Correlation

CORRE	means, standard deviations, and correlations

Multiple Linear Regression

ORDER	rearrangement of intercorrelations
MULTR	multiple regression and correlation

Polynomial Regression

GDATA	data generation

Analysis of Variance

AVDAT	data storage allocation
AVCAL	sigma and delta operation
MEANQ	mean square operation

Discriminant Analysis

DMATX	means and dispersion matrix
DISCR	discriminant functions

Factor Analysis

TRACE	cumulative percentage of eigenvalues
LOAD	factor loading
VARMX	varimax rotation

Time Series

AUTO	autocovariances
CROSS	crosscovariances
SMO	application of filter coefficients (weights)
EXSMO	triple exponential smoothing

Nonparametric Statistics

CHISQ	Chi-square test for a contingency table
UTEST	Mann-Whitney U-test
TWOAV	Friedman two-way analysis of variance
QTEST	Cochran Q-test
SRANK	Spearman rank correlation
KRANK	Kendall rank correlation
WTEST	Kendall coefficient of concordance
RANK	rank observations
TIE	calculation of ties in ranked observations

Canonical Correlation

CANOR	canonical Correlation
NROOT	eigenvalues and eigenvectors of a special non-symmetric matrix

Special Matrix Operations

MINV	matrix inversion
EIGEN	eigenvalues and eigenvectors of a real, symmetric matrix
MFGR, DMFGR(1)	matrix factorization and rank determination

Matrices

GMADD	add two general matrices
GMSUB	subtract two general matrices
GMPRD	product of two general matrices
GMTRA	transpose of a general matrix

GTPRD	transpose product of two general matrices
MADD	add two matrices
MSUB	subtract two matrices
MPRD	matrix product (row into column)
MTRA	transpose a matrix
TPRD	transpose product
MATA	transpose product of matrix by itself
SADD	add scalar to matrix
SSUB	subtract scalar from a matrix
SMPY	matrix multiplied by a scalar
SDIV	matrix divided by a scalar
RADD	add row of one matrix to row of another matrix
CADD	add column of one matrix to column of another matrix
SRMA	scalar multiply row and add to another row
SCMA	scalar multiply column and add to another column
RINT	interchange two rows
CINT	interchange two columns
RSUM	sum the rows of a matrix
CSUM	sum the columns of a matrix
RTAB	tabulate the rows of a matrix
CTAB	tabulate the columns of a matrix
RSRT	sort matrix rows
CSRT	sort matrix columns
RCUT	partition row-wise
CCUT	partition column-wise
RTIE	adjoin two matrices row-wise
CTIE	adjoin two matrices column-wise
MCPY	matrix copy
XCPY	copy submatrix from given matrix
RCPY	copy row of matrix into vector
CCPY	copy column of matrix into vector
DCPY	copy diagonal of matrix into vector
SCLA	matrix clear and add scalar
DCLA	replace diagonal with scalar
MSTR	storage conversion

MFUN	matrix transformation by a function
RECP	reciprocal function for MFUN
LOC	location in compressed-storage matrix
CONVT	single precision-double precision conversion
ARRAY	vector storage-double dimensioned storage conversion

<u>Integration and Differentiation</u>

QTFE, DQFG	integration of monotonically tabulated function by trapezoidal rule
QTFE, DQFE	integration of equidistantly tabulated function by trapezoidal rule
QSF, DQSF	integration of equidistantly tabulated function by Simpson's rule
QHFG, DQHFG	integration of monotonically tabulated function with first derivative by Hermitian formula of first order
QHFE, DQHFE	integration of equidistantly tabulated function with first derivative by Hermitian formula of first order
QHSG, DQHSG	integration of monotonically tabulated function with first and second derivatives by Hermitian formula of first order
QHSE, DQHSE	integration of equidistantly tabulated function with first and second derivatives by Hermitian formula of second order
QATR, DQATR	integration of a given function by trapezoidal rule together with Romberg's extrapolation method
QG2 QG3(2) QG4 QG5 QG6 QG7 QG8 QG9 QG10 DQG4 DQG8 DQG12 DQG16 DQG24 DQG32	integration of a given function by Gaussian quadrature formulae
QL2 QL3 QL4 QL5 QL6 QL7 QL8 QL9 QL10 DQL4 DQL8 DQL12 DQL16 DQL24 DQL32	integration of a given function by Gaussian-Laguerre formulae

Integration and Differentiation (continued)

QH2	QH3	
QH4	QH5	
QH6	QH7	
QH8	QH9	
QH10	DQH8	
DQH16	DQH24	integration of a given function by Gaussian-Hermite
DQH32	DQH48	quadrature formulae
DQH64		

QA2	QA3	
QA4	QA5	
QA6	QA7	
QA8	QA9	
QA10	DQA4	
DQA8	DQA12	integration of a given function by associated Gaussian-
DQA16	DQA24	Laguerre quadrature formulae
DQA32		

Ordinary Differential Equations

RK1	solution of first-order differential equation by Runge-Kutta method
RK2	tabulated solution of first-order differential equations by Runge-Kutta method
RKGS, DRDGS	solution of system of first order ordinary differential equations with given initial values by the Runge-Kutta method
HPCG, DHPCG	solution of general system of first order ordinary differential equations with given initial values by Hamming's modified predictor-corrector method
HPCL, DHPCL	solution of linear system of first order ordinary differential equations with given initial values by Hamming's modified predictor-corrector method
LBVP, DLBVP	solution of system of linear first order ordinary differential equations with linear boundary conditions by method of adjoint equations

Fourier Analysis

FORIF	Fourier analysis of a given function
FORIT	Fourier analysis of a tabulated function
HARM, DHARM	complex three-dimensional analysis
RHARM, DRHARM	real one-dimensional analysis

Special Operations and Functions

GMMMA	gamma function
BESJ	J Bessel function
BESY	Y Bessel function
BESI	I Bessel function
BESK	K Bessel function
EXPI	exponential integral
SICI	sine cosine integral
CS	Fresnel integrals
CEL1, DCEL1	complete elliptic integral of the first kind
CEL2, DCEL2	complete elliptic integral of the second kind
ELI1, DELI1	generalized elliptic integral of the first kind
ELI2, DELI2	generalized elliptic integral of the second kind
JELF, DJELF	Jacobian elliptic functions

Linear Equations

SIMQ	solution of simultaneous linear, algebraic equations
GELG, DGELG	system of general simultaneous linear equations by Gauss elimination
GELS, DGELS	system of general simultaneous linear equations with symmetric coefficients
GELB, DGELB	system of general simultaneous linear equations with band structured coefficients
LLSQ, DLLSQ	solution of linear least squares problems

Non Linear Equations

RTWI, DRTWI	refine estimate of root by Wegstein's iteration
RTMI, DRTMI	determine root within a range by Mueller's iteration
RTNI, DRTNI	refine estimate of root by Newton's iteration
FMFP, DFMFP	unconstrained minimum of a function of several variables -- Davidson method
FMCG, DFMCG	unconstrained minimum of a function of several variables -- conjugate gradient method

Roots of Polynomials

POLRT	real and complex roots of a real polynomial
PRQD, DPRQD	roots of a real polynomial by QD algorithm with displacement

Special Polynomials

CNP, DCNP	value of Nth Chebyshev polynomial
CNPS, DCNPS	value of series expansion in Chebyshev polynomials
TCNP, DTCNP	transform series expansion in Chebyshev polynomials to a polynomial
CSP, DCSP	value of Nth shifted Chebyshev polynomial
CSPS, DCSPS	value of series expansion in shifted Chebyshev polynomials
TCSP, DTCSP	transform series expansion in shifted Chebyshev polynomials to a polynomial
HEP, DHEP	value of Hermite polynomial
HEPS, DHEPS	value of series expansion in Hermite polynomials
THEP, DTHEP	transform series expansion in Hermite polynomials to a polynomial
LAP, DLAP	value of a Laguerre polynomial
LAPS, DLAPS	value of series expansion in Laguerre polynomials
TLAP, DTLAP	transform series expansion in Laguerre polynomials to a polynomial
LEP, DLEP	value of Legendre polynomial
LEPS, DLEPS	value of series expansion in Legendre polynomials
TLEP, DTLEP	transform a series expansion in Legendre polynomials to a polynomial

Polynomial Operations

PADD	add two polynomials
PADDM	multiply polynomial by constant and add to another polynomial
PCLA	replace one polynomial by another
PSUB	subtract one polynomial from another
PMPY	multiply two polynomials
PDIV	divide one polynomial by another
PQSD	quadratic synthetic division of a polynomial
PVAL	value of a polynomial
PVSUB	substitute variable of polynomial by another polynomial
PCLD	complete linear synthetic division
PILD	evaluate polynomial and its first derivative
PDER	derivative of a polynomial
PINT	integral of a polynomial

PGCD	greatest common divisor of two polynomials
PNORM	normalize coefficient vector of polynomial
PECN, DPECN	economization of a polynomial for symmetric range
PECS, DPECS	economization of a polynomial for unsymmetric range

Approximation, Interpolation, Table Construction

ALI, DALI	Aitken-Lagrange interpolation
AHI, DAHI	Aitken-Hermite interpolation
ACFI, DACFI	continued fraction interpolation
ATSG, DATSG	table selection out of a general table
ATSM, DATSM	table selection out of a monotonic table
ATSE, DATSE	table selection out of an equidistant table

Convergence Accelerating Operations

TEAS, DTEAS	limit of a given sequence
TEUL, DTEUL	sum of a given function sequence

Random Number Generators

RANDU	uniform random numbers
GAUSS	normal random numbers

NOTES

(1) Where there are two subroutine names separated by a comma, the first is the single precision version and the second is the double precision version. This notation also serves to identify the Version 2 additions to SSP/360.

(2) In each family of quadrature formulae, the factor that differs is the number of points used. Also, Q designates single precision subroutines and D designates double precision subroutines.

6.4 Applications Program Library

6.4.1 Biomedical Computer Programs (BMD)

6.4.1.1 General Information

The Biomedical Computer Programs (BMD) are a set of complete programs designed to serve the following needs:

1. To provide programs for the commonly used tasks of data processing and statistical analysis.
2. To provide programs in a "package" form so that researchers may achieve their desired computations with simple coded instructions.
3. To provide "package" programs in a general form so that a wide variety of problems may be handled by each program simply by specifying the appropriate parameters of the problem.

The BMD programs were obtained from the UCLA Health Sciences Computing Facility. Since this was not a Stanford developed system, it was necessary to verify the reputed performance of the programs. They were run with the test data provided in the BMD manual and the resultant output checked against the manual's test results. An agreement of four significant places was an average for these comparisons. The dissimilarity in the results could be caused by the difference in word length size between the IBM 7090 and the IBM 360/67, i.e., a 32-bit word for the 360/67 and a 36-bit word for the 7090. The other noticeable difference is that missing integer data was represented by a -0 on the 7090, but appears as a 0 on the 360/67. This occurs because there is no possible representation for an integer -0 on the IBM 360/67.

Although we are relying on UCLA for maintenance of these programs, users are encouraged to report any difficulties encountered to Arline Kapphahn, User Services Group, Room 185, Pine Hall.

7.4.1.2 Documentation

These programs are described in the Biomedical Computer Programs publication which is available from the Stanford Bookstore.

Reference copies of this manual are available in the Computer Science Library and the study room in the Dispatch lobby.

6.4.1.3 Available Programs

The following BMD programs are now available for use on the Campus Facility's 360/67 computer.

Class D-Description and Tabulation

BMD01D	Simple data description
BMD02D	Correlation with transgeneration
BMD03D	Correlation with item deletion
BMD04D	Alphanumeric frequency count
BMD05D	General plot including histogram
BMD06D	Description of strata
BMD07D	Description of strata with histograms
BMD08D	Cross-tabulation with variable stacking
BMD09D	Cross-tabulation, incomplete data
BMD10D	Data patterns for dichotomies
BMD11D	Data patterns for polychotomies

Class M-Multivariate Analysis

BMD01M	Principal component analysis
BMD02M	Regression on principal components
BMD03M	Factor analysis
BMD04M	Discriminant analysis for two groups
BMD05M	Discriminate analysis for several groups
BMD06M	Canonical analysis
BMD07M	Stepwise discriminant analysis

Class R-Regression Analysis

BMD01R	Simple linear regression
BMD02R	Stepwise regression
BMD03R	Multiple regression with case combinations
BMD04R	Periodic regression and harmonic analysis

BMD05R	Polynomial regression
BMD06R	Asymptotic regression

Class S-Special Programs

BMD01S	Life table and survival rate
BMD02S	Contingency table analysis
BMD03S	Biological assay: probit analysis
BMD09S	Transgeneration
BMD10S	Transposition of large matrices

Class T-Time Series Analysis

BMD01T	Amplitude and phase analysis
BMD02T	Autocovariance and power spectral analysis

Class V-Variance Analysis

BMD01V	Analysis of variance for one-way design
BMD02V	Analysis of variance for factorial design
BMD03V	Analysis of covariance for factorial design
BMD04V	Analysis of covariance with multiple covariates
BMD05V	General linear hypothesis
BMD06V	General linear hypothesis with contrasts
BMD08V	Analysis of variance (new program)

Class X-Supplementary Programs

BMDX63	Multivariate general linear hypothesis
BMDX64	General linear hypothesis
BMDX65	Substitute means for missing values
BMDX69	Multivariate analysis of variance and covariance
BMDX70	T-test and F-test program
BMDX71	Non-linear least squares estimation
BMDX72	Factor analysis
BMDX73	Multiple time series spectral estimation
BMDX74	Identification of outliers
BMDX75	Canonical analysis
BMDX76	Life tables and survival rate

BMD05R	Polynomial regression
BMD06R	Asymptotic regression

Class S-Special Programs

BMD01S	Life table and survival rate
BMD02S	Contingency table analysis
BMD03S	Biological assay: probit analysis
BMD09S	Transgeneration
BMD10S	Transposition of large matrices

Class T-Time Series Analysis

BMD01T	Amplitude and phase analysis
BMD02T	Autocovariance and power spectral analysis

Class V-Variance Analysis

BMD01V	Analysis of variance for one-way design
BMD02V	Analysis of variance for factorial design
BMD03V	Analysis of covariance for factorial design
BMD04V	Analysis of covariance with multiple covariates
BMD05V	General linear hypothesis
BMD06V	General linear hypothesis with contrasts
BMD08V	Analysis of variance (new program)

Class X-Supplementary Programs

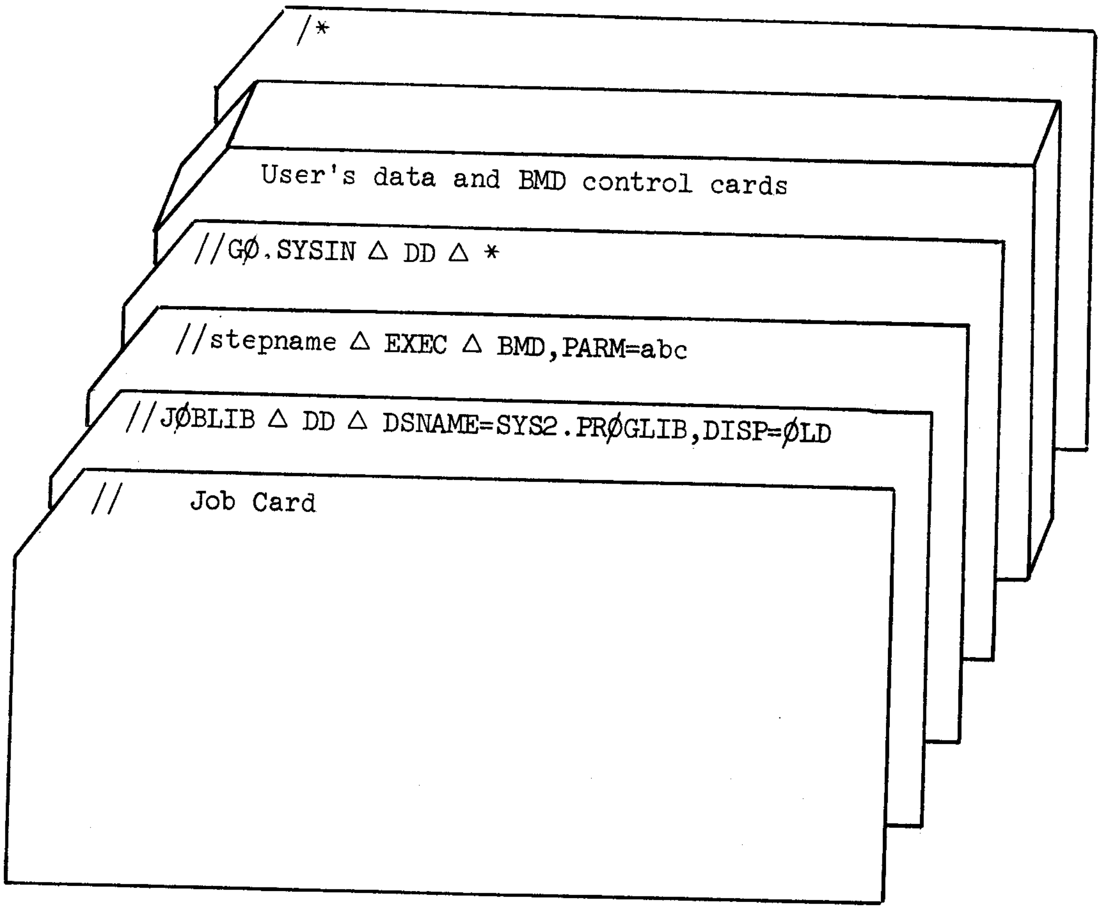
BMDX63	Multivariate general linear hypothesis
BMDX64	General linear hypothesis
BMDX65	Substitute means for missing values
BMDX69	Multivariate analysis of variance and covariance
BMDX70	T-test and F-test program
BMDX71	Non-linear least squares estimation
BMDX72	Factor analysis
BMDX73	Multiple time series spectral estimation
BMDX74	Identification of outliers
BMDX75	Canonical analysis
BMDX76	Life tables and survival rate

BMDX77	Transgeneration
BMDX80	Mean frequency epoch analysis
BMDX84	Asymetrical correlation with missing data
BMDX85	Nonlinear least squares

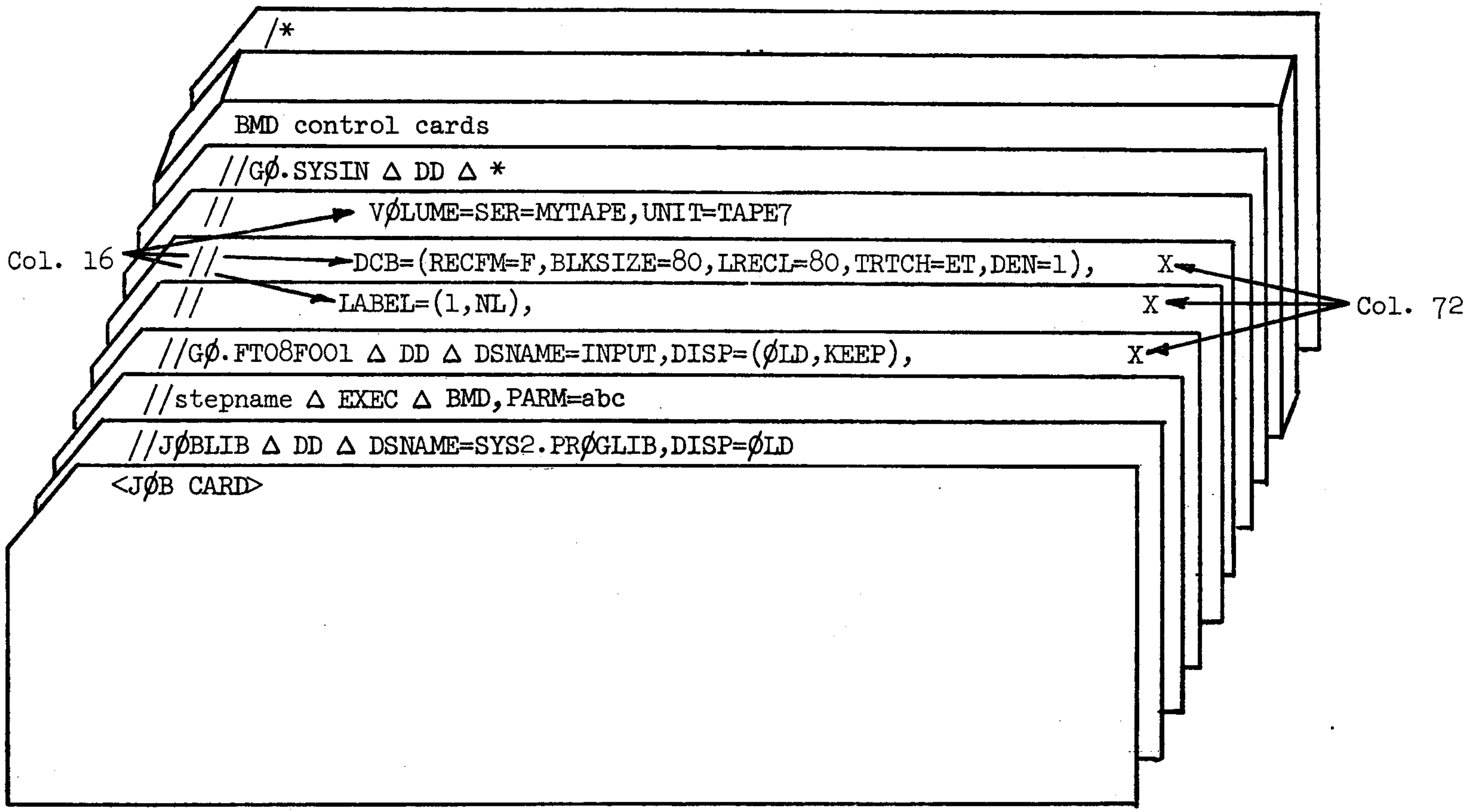
6.4.1.4 BMD Job Set-up

The BMD programs are included in the Applications Program Library and the Source Library. The following job set-ups will allow you to process data from cards or tape, respectively.

1. Sample deck set-up for executing a BMD program with the input data and control statement on cards.


/*
User's data and BMD control cards
//GØ.SYSIN △ DD △ *
//stepname △ EXEC △ BMD,PARM=abc
//JØBLIB △ DD △ DSNAME=SYS2.PRØGLIB,DISP=ØLD
// Job Card

2. Sample deck set-up for executing a BMD program with the input data on tape number 8 and the BMD control statements on cards. The tape DD card describes the tape as a BCD card-image tape in a 7090 compatible format. The data is in the first file on the tape.



```
/*
BMD control cards
//GØ.SYSIN Δ DD Δ *
//             VØLUME=SER=MYTAPE,UNIT=TAPE7
//             DCB=(RECFM=F,BLKSIZE=80,LRECL=80,TRTCH=ET,DEN=1),      X
//             LABEL=(1,NL),                                          X
//GØ.FT08F001 Δ DD Δ DSNAME=INPUT,DISP=(ØLD,KEEP),                   X
//stepname Δ EXEC Δ BMD,PARM=abc
//JØBLIB Δ DD Δ DSNAME=SYS2.PRØGLIB,DISP=ØLD
<JØB CARD>
```

Note: In order to process tapes whose formats differ from the above, or which are referred to by a tape number other than 8, the user will have to provide an appropriate DD statement. DD statements are described in Section 3.2.4 of this manual and in the publication IBM System/360 Operating System Job Control Language (Form No. C28-6539).

6.4.2 Electronic Circuit Analysis Program (ECAP)

The IBM Electronic Circuit Analysis Program, ECAP, is a general circuit analysis program allowing DC, AC, and transient analyses of circuits with up to 50 nodes and 200 branches. Use of the program can facilitate the analysis of most any system for which an electrical analog can be specified. Node voltages, element currents, and power dissipation are typically determined from a circuit-oriented description of the network.

Currently the program can be entered into the batch stream from either the WYLBUR terminals or the card readers.

Documentation for this program is the IBM manual Electronic Circuit Analysis Program User's Manual (Form No. H20-0170). The documentation is available from the Systems Documentation Office, Room 185, Pine Hall.

6.4.3 Linear Programming System (LPS)

A Linear Programming System, which is a part of IBM's Mathematical Programming System, is available for use on the 360/67. This system consists of a compiler and an executor. The compiler accepts programs written in a special linear programming language which consists of:

- Procedures which execute and maintain problem files consisting of coefficient matrices, multiple right hand sides, multiple objective functions, multiple bounds, rows and range columns, and starting bases.
- Procedures which find an optimum solution. Primal and dual algorithms are provided. The primal uses a revised simplex (product form of the inverse) algorithm with bounded variables and range constraints.
- Procedures which do post-optimal studies. Ranging, perturbing the objective function, the right hand side, or any row or column is provided. These things can be done simultaneously.
- Statements which change certain constants and sub-procedures within the system.

The executor takes the output of the compiler and executes the indicated procedures using data provided by the user in the input stream and other data which the user has placed in the system from past runs.

The system will handle problems with at least 2,048 rows and virtually infinite columns. The optimization algorithms are extremely efficient. A small problem (14 constraints and 21 variables) was optimized in less than .01 minutes.

Documentation is available from the Systems Documentation Office, Room 185, Pine Hall.

The following deck set up may be used for this program.

```
//      Job Card
//JØBLIB △ DD △ DSNAME=SYS2.LP,UNIT=2314,DISP=(ØLD,PASS),VØLUME=SER=SYS06
//STEP1 △ EXEC △ LPSYS
//CØMP.SYSIN △ DD △ *

        MPS Program

/*
//GØ.SYSIN △ DD △ *

        Data

/*
```

6.4.4 Continuous System Modeling Program (CSMP)

IBM's Continuous System Modeling Program (CSMP) is a problem-oriented program designed to facilitate the simulation of continuous processes on the 360. The program provides an application-oriented language that allows these problems to be prepared directly and simply from either a block-diagram representation or a set of ordinary differential equations.

Typical applications might be a control engineer's study of the effectiveness of various control system designs, a physiologist's simulation of the cardio-vascular system, and a mechanical engineer's investigation of the effects of damping and backlash in a proposed mechanical device.

The following is a sample deck set-up for CSMP programs:

```
//   Job Card
//JØBLIB  DD  DSNAME=SYS2.PRØGLIB,DISP=(ØLD,PASS)
//STEP1   EXEC  CSMP1
//CSMP1.SYSIN  DD  *
     Data cards for CSMP
ENDJØB
/*
```

Documentation:

1. S/360 Continuous System Modeling Program Application Description (Form No. H20-0240).
2. S/360 Continuous System Modeling Program User's Manual, (Form No. H20-0367).

6.4.5 TRANSPORT

The Physics Application program, TRANSPORT, calculates the properties of particles in a beam traveling through a magnetic deflection system.

The following is a sample deck set-up for TRANSPORT programs:

```
//   Job Card
//JØBLIB  DD  DSNAME=SYS2.PRØGLIB,DISP=(ØLD,PASS)
//STEP1   EXEC  PGM=TRNSPØRT
//FT04F001  DD  UNIT=SYSCP,DCB=(RECFM=F,BLKSIZEE=80)
//FT06F001  DD  SYSØUT=A
//FT05F001  DD  *
     Data cards for Transport
/*
```

Documentation for this program is available from the Systems Documentation Office.

6.4.6 General Purpose System Simulator/360 (GPSS/360)

IBM's General Purpose System Simulator/360 is suitable for modeling and examining the behavior of systems in the engineering, management science, and computer systems areas. GPSS/360 is a significant extension of the GPSS III language on the 7090.

There are several IBM manuals available for this system.

H20-0186	GPSS/360 Application Description
H20-0304	GPSS/360 Introductory User's Manual
H20-0326	GPSS/360 User's Manual

The documentation is available from the Systems Documentation Office, Room 185, Pine Hall.

The following is a sample deck setup for GPSS users.

```
//      Job Card
//JØBLIB △ DD △ DSNAME=SYS2.PRØGLIB,DISP=(ØLD,PASS)
//NØW △ EXEC △ PGM=DAG01,PARM=C
//DØUTPUT △ DD △ SYSØUT=A
//DINTERØ △ DD △ DSNAME=SYS1.UT1,DISP=ØLD
//DSYMTAB △ DD △ DSNAME=SYS1.UT2,DISP=ØLD
//DREPTGEN △ DD △ DSNAME=SYS1.UT3,DISP=ØLD
//DINTWØRK △ DD △ DSNAME=SYS1.UT4,DISP=ØLD
//DINPUT1 △ DD △ *

          GPSS Problem Formulation

/*
```

Note: The symbol "△" indicates one or more blanks.

6.4.7 NWAY

NWAY is a tabulation and statistical processing program for social science research. It is a general statistical program with the following special features:

- o A mixture of different computation can be requested at one time and will be executed, to the extent of storage facilities available on the computer, in the same pass through the user's data.
- o Storage in the computer is dynamically allocated by the program so that the user rarely need be concerned with the storage requirements of the tasks that he requests.
- o The program is designed to be as efficient as possible in the handling of large data files. This feature makes it feasible to perform analyses on such large bodies of data as population censuses.
- o The user is given control of the data input, enabling the use of any kind of transgenerative or recoding process desired.
- o The program can simultaneously perform calculations on different units of data. For example, the program could, on a single pass of a population census file, perform analyses of both variables measured on individuals and on variables measured on their households.
- o Facilities are provided for the labeling of both variables and categories within variables (i.e., values of the variable).
- o NWAY is written in an elementary subset of FORTRAN IV, and should be readily adaptable to most FORTRAN IV processors. Its current implementation is in OS/360 FORTRAN IV (H).

NWAY may be executed using a catalogued procedure. The following job control is required for execution:

```
//   Job Card
//JØBLIB  DD  DSNAME=SYS2.PRØGLIB,DISP=ØLD
//   EXEC   NWAY
//GØ.SYSIN  DD  *
         NWAY input:  includes parm cards, vars and varl cards, format
                      cards, data, etc.
/*
```

(NOTE: If your data is not on cards but on a disk or tape data set, an additional DD card defining the data set is required for the GØ step.)

Documentation for NWAY is available from the Systems Documentation Office, Room 185, Pine Hall, for $1.00.

6.4.8 MDSCAL

MDSCAL is a multi-dimensional scaling program written by Joseph Kruskal, Bell Telephone Laboratories. It has been converted to FORTRAN IV (Level H) and is available in the applications program library, SYS2.PRØGLIB.

Multi-dimensional scaling is a statistical technique for cluster analysis. It constructs a configuration of points in space from information about the distance between the points. A wide variety of options is available. For example, the data may be in the form of either an entire matrix or a lower halfmatrix. The matrix need not be symmetric, and the diagonal may be absent. Either similarities or dissimilarities are acceptable, replicated measurements are permitted, data values may be missing, and weighting is available. A maximum of 60 objects may be scaled in up to 10 dimensions, with a maximum of 1800 data values.

MDSCAL documentation is available from the Systems Documentation Office, Room 185, Pine Hall, for $1.75.

6.5 IBM CATALOG OF PROGRAMS

6.5.1 Introduction

Another source of programs available to the user are those listed in the document Catalog of Programs for IBM/360, (Form No. C20-1619). This publication is available for reference in the Systems Documentation Office, Room 185, Pine Hall, Ext. 4877.

The publication lists four types of available programs:

Type I
Programming Systems: Programs conceived and developed by IBM as integral parts of the data processing system for which they are written.

Type II
Application Programs: Carefully selected solutions by IBM of data processing problems. They are supported by well planned documentation and tested procedures.

Type III
IBM-Contributed Programs: Programs contributed by IBM employees to aid the programming and system community.

Type IV
Customer Contributed Programs: Aids to the programming community supplied by members of customer organizations and individual users of IBM Data Processing Systems.

6.5.2 How to Obtain a Program

Users interested in obtaining a copy of a program listed in the Catalog of Programs may do so through the Systems Documentation Office. If the program requested is distributed on magnetic tape, the user will be required to supply the tape. Programs of this type take about three weeks for delivery.

6.6 SLAC Computer Program Library

6.6.1 Introduction

This program library serves the needs of the SLAC Facility for programs from the fields of physics applications, graphics and picture processing as well as an archive to document and distribute programs referenced in SLAC publications; systems programs used only at SLAC are also included. The library is meant to supplement the Stanford Extrinsic Program Library described in Section 6.2 of this manual, and is less formal in its acceptance requirements, documentation and management. A list of the entries in the SLAC library is included in this section.

6.6.2 Documentation

Program submittal forms, write-ups, distribution decks, and other information may be obtained from the SLAC program librarian, Linda Lorenzetti at SLAC, Ext. 8689.

A KWIC index of the SLAC library is also available for reference in the Systems Documentation Office, Room 185, Pine Hall, Ext. 4877.

6.6.3 Available Programs

Each program entry includes the following information:

Addd(SC.L) Descriptive Title with Keywords. (subroutine names)
Machine and language

where

Addd	is the SLAC program library number
SC	is the SHARE Classification code e.g., J5 Output - plotting
L	is the language code,

A = B5500 ALGOL
F = IBM7090 FORTRAN II
G = IBM7090 FORTRAN IV
H = IBM360 FORTRAN IV (H)
M = IBM360 Assembler Language Macro
S = IBM7090 SUBALGOL
Z = Miscellaneous

A001(J5.S)	GRAPH (CURVE-PLOTTING) AND CONTOUR OF FUNCTIONS ON THE CALCOMP PLOTTER. (AXAS,GRAPH,SKALE,CONTOUR) 7090 SUBALGOL PROCEDURES
A002(S15.S)	ERROR-FUNCTION BY HASTING'S APPROXIMATION. (ERF) 7090 SUBALGOL PROCEDURE
A003(M1.S)	LEXICOGRAPHIC ORDERING OF THE ROWS OF A MATRIX. (ORDER) 7090 SUBALGOL PROCEDURE
A004(T1.F)	MAGNETIC-FIELD CALCULATION FOR AN AIR-GAP COIL. (COILB) 7090 FORTRAN II PROGRAM
A005(E1.S)	QUADRATIC-FORM CONTOURS CALCULATION. (EPOINTS) 7090 SUBALGOL PROCEDURE
A006(F4.S)	LEAST-SQUARES FITTING OF FUNCTIONS TO MEASURED DATA. (CURVE) 7090 SUBALGOL PROGRAM
A007(T1.S)	TRANSPORT: A PROGRAM TO SOLVE 1ST AND 2ND ORDER BEAM-TRANSPORT OPTIC PROBLEMS. (TRANSPORT) IBM7090 SUBALGOL AND FORTRAN IV AND IBM360 FORTRAN IV (H)
A008(J4.A)	BASIC-ALGOL: OUTPUT PROCEDURES FOR ALGOL. (TAB,TAB1,HEAD, PRINT,LINEQ,INTEGRAL,LEBESQUE) B5500 ALGOL PROCEDURES
A009(J5.A)	CURVE-PLOTTING (GRAPH-PROCEDURES) ON THE CALCOMP PLOTTER. (POINT,CURVE,TITLE,GRAPH) B5500 ALGOL PROCEDURES

A010(T1.A)	CALCULATION OF AXIAL MAGNETIC-FIELD FROM SOLENOIDS AND IRON RINGS ALONG THE FIELD AXIS. (IRONSUM,COILSUM) B5500 ALGOL
A011(T1.A)	CALCULATION OF MAGNETIC-FIELD FROM SOLENOIDS AND IRON AT ANY POINT IN SPACE. (MARKIV) B5500 ALGOL PROGRAM
A012(G5.A)	UNIFORM AND NORMAL RANDOM-NUMBER GENERATORS. (SETUPRANDOM, RANDOM,NORMAL) B5500 ALGOL PROCEDURES
A013(T .A)	CALCULATION OF DEFLECTION, MOMENT, AND MAXIMUM BENDING-STRESS OF AN END-SUPPORTED-BEAM. B5500 ALGOL PROGRAM
A014(T1.A)	BEAM-TRACE THROUGH A SYSTEM OF MAGNETIC-LENS ELEMENTS. (BEAMTRACE) B5500 ALGOL PROGRAM
A015(M1.A)	SORT AND SHUFFLE THE ROWS OF AN ARRAY. (SORT, SHUFFLE) B5500 ALGOL PROCEDURES
A016(J5.A)	CURVE-PLOTTING USING THE LINE-PRINTER. (OUTPLOT) B5500 ALGOL PROCEDURE
A017(E5.A)	MINIMIZE A FUNCTION OF ONE VARIABLE. (MINIMIZE B5500 ALGOL PROCEDURE
A018(Z .A)	PRINT THE CALENDAR FOR ANY YEAR BC OR AD (CALENDAR) B5500 ALGOL PROGRAM
A019(D1.A)	NUMERICAL INTEGRATION BY ADAPTIVE SIMPSON'S-RULE. (SIMPS5) B5500 ALGOL PROCEDURE
A020(J5.A)	CONTOUR PLOT OF A FUNCTION OF TWO VARIABLES ON THE CALCOMP PLOTTER. (CONTOUR B5500 ALGOL PROCEDURE
A021(T .A)	PROGRAM FOR VARIABLE PERMEABILITY MAGNETOSTATIC-FIELD PROBLEMS. (NUTCRACKER) B5500 ALGOL PROGRAM
A022(F4.A)	NONLINEAR LEAST-SQUARES WITH CONSTRAINTS. (SOLVE) B5500 ALGOL PROCEDURE
A023(T .A)	THICK TARGET, HIGH ENERGY BREMSSTRAHLUNG. B5500 ALGOL PROGRAM
A024(T .A)	HIGH ENERGY SECONDARY-PARTICLE PHOTOPRODUCTION CALCULATION FOR SLAC-USER'S-HANDBOOK. (FIVE PROGRAMS) B5500 ALGOL PROGRAMS
A025(J4.H)	2250-SCOPE OUTPUT FOR BPS FORTRAN. (STTSCP,RSETUP,AXES,ETC.) IBM360 FORTRAN SUBROUTINES
A026(D1.A)	NUMERICAL INTEGRATION BY RECURSION. (SIMPS6) B5500 ALGOL PROCEDURE

A027(Q1.H)	READ INTERVAL TIMER AND CONVERT TO MILLISECONDS. (TIM1) IBM360 FORTRAN SUBROUTINE
A028(M4.H)	SHIFT-OPERATIONS FOR FULLWORD OPERANDS. (SHF1:SHF1RA,SHF1LA, SHF1RL,SHF1LL,SHF1RC,SHF1LC) IBM360 FORTRAN SUBROUTINES
A029(M4.H)	SHIFT-OPERATIONS FOR DOUBLEWORD OPERANDS. (SHF2:SHF2R, SHF2L) IBM360 FORTRAN SUBROUTINES
A030(R1.H)	LOGICAL-OPERATIONS ON FULLWORD OPERANDS. (LG01:LG01AN, LG01OR,LG01XR,LG01CM) IBM360 FORTRAN SUBROUTINES
A031(M2.H)	HEXADECIMAL TO EBCDIC CONVERSION. (HEX1) IBM360 FORTRAN SUBROUTINE
A032(A1.H)	GREATEST-COMMON-DIVISOR. (GCD1) IBM360 FORTRAN SUBROUTINE
A033(A1.H)	NEXT PRIME-NUMBER). (XP1) IBM360 FORTRAN SUBROUTINE
A034(M2.H)	FORTRAN MOVE-A-CHARACTER ROUTINE. (VC1) IBM360 FORTRAN SUBROUTINE
A035(J9.M)	PRINTOUT-MACRO INSTRUCTION FOR SIMPLE OUTPUT AND DEBUGGING. (PRINTOUT) IBM360 ASSEMBLER LANGUAGE
A036(19.M)	READCARD-MACRO INSTRUCTION FOR SIMPLE INPUT. (READCARD) IBM360 ASSEMBLER LANGUAGE
A037(T .A)	BREMSSTRAHLUNG TABLES FOR POSITRON-HYDROGEN ATOM COLLISION. B5500 ALGOL PROGRAM
A038(M5.H)	BINARY SEARCH-FUNCTION. (IBS) IBM360 FORTRAN SUBROUTINE
A039(19.H)	FORTRAN INTERNAL INPUT/OUTPUT BUFFER-CONTROL. (F10999, F1099X) IBM360 FORTRAN SUBROUTINES
A040(T5.F)	DIREC-GAMMA-MATRIX TRACE REDUCTION PROGRAM. (FTRACE,GLEANE, TALKER,CHODED) IBM7090 MAP AND FORTRAN IV PROGRAM
A041(L7.A)	EXTENDED ALGOL-TO-FORTRAN TRANSLATOR. (ALTRAN/1 B5500 ALGOL PROGRAM
A042(Q1.H)	ABNORMAL-END PROGRAM TERMINATOR. (ABD1) IBM360 FORTRAN SUBROUTINE
A043(J9.H)	PROGRAMMED CHARACTER-GENERATOR FOR 2250-SCOPE OUTPUT-BUFFER. BCDVE1,BCDVE2,BCDVE3) IBM360 FORTRAN IV AND ASSEMBLER LANGUAGE SUBROUTINES
A044(J9.H)	INPUT/OUTPUT INTERFACE FOR 2250-SCOPE. (CRTIOA,CRTIOB) IBM360 FORTRAN SUBROUTINES

A045(J9.H)	EXPAND AND TRANSLATE GRAPHIC-ELEMENTS FOR THE IBM 2250-SCOPE. (ZOOM1,EXPND1) IBM360 FORTRAN SUBROUTINES
A046(A1.H)	DOUBLE PRECISION PRODUCT-ACCUMULATION WITH OPTIONAL ROUNDING. (DPR1R,DPR1U) IBM360 FORTRAN SUBROUTINES
A047(A1.H)	PRODUCT-ACCUMULATION FOR SHORT AND LONG FLOATING-POINT OPERANDS. (DPR2) IBM360 FORTRAN SUBROUTINES
A048(M2.H)	HEXADECIMAL TO EBCDIC CONVERSION ROUTINE. (HEX2) IBM360 FORTRAN SUBROUTINE
A049(T .S)	POISSON-EQUATION SOLVING PROGRAM. IBM 7090 SUBALGOL PROGRAM
A050(J9.H)	FORTRAN INTERFACE SUBROUTINES FOR BUFFERED IBM2250-SCOPE MOD 1 INPUT/OUTPUT. (GRDBUF,GRDSTR,ETC.) IBM360 FORTRAN SUBROUTINES
A051(Q .H)	UTILIZATION OF AVAILABLE-MEMORY WITH FORTRAN PROGRAMS. (#AVAILM) IBM360 SORTRAN SUBROUTINES
A052(Q .H)	SUBROUTINE TO RELEASE-A-PARTITION UNDER OS/360 VIA THE WAITR MACRO. (WAITR) IBM360 FORTRAN SUBROUTINE
A053(J9.H)	SUBROUTINE TO SET THE BUFFER-SWITCH ON THE IBM 2250-SCOPE MOD 1. (SETBUF) IBM360 FORTRAN SUBROUTINE

APPENDIX A: GLOSSARY

access method: Any of the data management techniques available to the user for transferring data between main storage and an input/output device.

alias: An alternate name that may be used to refer to a member of a partitioned data set; an alternate entry point at which execution of a program can begin.

allocate: To grant a resource to, or reserve it for, a job or task.

attribute: A characteristic; e.g., attributes of data include record length, record format, data set name, associated device type and volume identification, use, creation date, etc.

auxiliary storage: Data storage other than main storage.

basic access method: Any access method in which each input/output statement causes a corresponding machine input/output operation to occur. (The primary macro instructions used are READ and WRITE.)

batch processing: (See stacked job processing.)

block (records):
1. To group records for the purpose of conserving storage space or increasing the efficiency of access or processing.
2. A physical record so constituted, or a portion of a telecommunications message defined to be a unit of data transmission.

buffer (program input/output): A portion of main storage into which data is read, or from which it is written.

cataloged procedure: A set of job control statements that has been placed in a special data set named SYS1.PROCLIB and that can be retrieved by naming it in an execute (EXEC) statement.

concatenated data set: A collection of logically connected data sets. Concatenation is a fancy word for joining collections of information (i.e. data sets, libraries) end-to-end.

control block: A storage area through which a particular type of information required for control of the operating system is communicated among its part.

control program: A collective or general term for all routines in the operating system that contribute to the management of resources, implement the data organization or communications conventions of the operations.

CPU (central processing unit): The unit of a system that contains the circuits that control and perform the execution of instructions.

data control block: A control block through which the information required by access routines to store and retrieve data is communicated to them.

data definition name (ddname): A name appearing in the data control block of a program which corresponds to the name field of a data definition statement.

data definition (DD) statement: A job control statement that describes a data set associated with a particular job step.

data management: A general term that collectively describes those functions of the control program that provide access to data sets, enforce data storage conventions, and regulate the use of input/output devices.

data set: The major unit of data storage and retrieval in the operating system, consisting of a collection of data in one of several prescribed arrangements and described by control information that is accessible by the system.

data set control block (DSCB): A data set label for a data set in direct-access storage.

data set label (DSL): A collection of information that describes the attributes of a data set, and that is normally stored with the data set; a general term for data set control blocks and tape data set labels.

device-independence: The ability to command input/output operations without regard to the characteristics of the input/output devices.

device name: Usually, the general name for a kind of device, specified at the time the system is generated. For example, 2311 or 2400 or TAPE.

direct access: Retrieval or storage of data by a reference to its location on a volume, rather than relative to the previously retrieved or stored data.

dump (main storage):
1. To copy the contents of all or part of main storage onto an output device, so that it can be examined.
2. The data resulting from 1.
3. A routine that will accomplish 1.

entry point: Any location in a program to which control can be passed by another program.

execute (EXEC) statement: A job control statement that designates a job step by identifying the load module or cataloged procedure to be fetched and executed.

input stream: The sequence of control statements and data submitted to the operating system on an input unit especially activated for this purpose by the operator.

job: A total processing application comprising one or more related processing programs, such as a weekly payroll, a day's business transactions, or the reduction of a collection of test data.

job control statement: Any one of the control statements in the input job stream that identifies a job or defines its requirements.

job library: A concatenation of user-identified partitioned data sets used as the primary source of load modules for a given job.

job management: A general term that collectively describes the functions of the job scheduler and master scheduler.

job processing: The reading of control statements from an input stream, the initiating of job steps defined in these statements, and the writing of SYSOUT messages.

job scheduler: The control program function that controls input job streams and system output, obtains input/output resources for jobs and job steps, attaches tasks corresponding to job steps, and otherwise regulates the use of the computing system by jobs. (See reader/interpreter, initiator/terminator, output writer.)

job (JOB) statement: The control statement in the input job stream that identifies the beginning of a series of job control statements for a single job.

job step: That unit of work associated with one processing program and related data. A cataloged procedure can comprise many job steps.

language translator: A general term for any assembler, compiler, or other routine that accepts statements in one language and produces equivalent statements in another language.

library:

1. In general, a collection of objects (e.g., data sets, volumes, card decks) associated with a particular use, and the location of which is identified in a directory of some type. In this context, see job library, link library, system library.
2. Any partitioned data set.

link library: A generally accessible partitioned data set which, unless otherwise specified, is used in fetching load modules referred to in execute (EXEC) statements and in ATTACH, LINK, LOAD, and transfer control (XCTL) macro instructions.

linkage: The means by which communication is effected between two routines or modules.

linkage editor: A program that produces a load module by transforming object modules into a format that is acceptable to fetch; combining separately produced object modules and previously processed load modules into a single load module; resolving symbolic cross references among them; replacing, deleting, and adding control sections automatically on request; and providing overlay facilities for modules requesting them.

load: To fetch, i.e., to read a load module into main storage preparatory to executing it.

load module: The output of the linkage editor; a program in a format suitable for loading into main storage for execution.

logical record: A record from the standpoint of its content, function, and use, rather than its physical attributes; i.e., one that is defined in terms of the information it contains.

macro instruction: A general term used to collectively describe a macro instruction statement, the corresponding macro instruction definition, the resulting assembler language statements, and the machine language instructions and other data produced from the assembler language statements; loosely, any one of these representations of a machine language instruction sequence.

main storage: All addressable storage from which instructions can be executed or from which data can be loaded directly into registers.

master scheduler: The control program function that responds to operator commands, initiates actions requested thereby, and returns requested or required information; thus, the overriding medium for controlling the use of the computing system.

module (programming): The input to, or output from a single execution of an assembler, compiler, or linkage editor; a source, object, or load module; hence, a program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading.

multiprogramming: A general term that expresses use of the computing system to fulfill two or more different requirements concurrently.

name: A 1- to 8-character alphameric term that identifies a data set, a control statement, a program, or a cataloged procedure. The first character of the name must be alphabetic.

object module: The output of a single execution of an assembler or compiler, which constitutes input to the linkage editor. An object module consists of one or more control sections in relocatable, though not executable, form and an associated control dictionary.

operator command: A statement to the control program, issued via a console device, which causes the control program to provide requested information, alter normal operations, initiate new operations, or terminate existing operations.

partitioned data set: Independent groups of sequentially organized data sets, each identified by a member name in the directory.

physical record: A record from the standpoint of the manner of form in which it is stored, retrieved, and moved; i.e., one that is defined in terms of physical qualities.

private library (of a job step): Any partitioned data set which is neither the link library nor any part of the job library.

problem program: Any of the class of routines that perform processing of the type for which a computing system is intended, and including routines that solve problems, monitor and control industrial processes, sort and merge records, perform computation, process transactions against stored records, etc.

processing program: Any program capable of operating in the problem program mode. This includes IBM-distributed language processors, application programs, service and utility programs, and user-written programs.

qualified name: A control statement term that comprises one or more names, each qualifying the name that follows it. Levels of qualification are separated by periods. For example, the term stepname.procstepname represents a procedure step name qualified by a job step name.

record: A general term for any unit of data that is distinct from all others when considered in a particular context.

reenterable: The attribute of a load module that allows the same copy of the load module to be used concurrently by two or more tasks.

resource: Any facility of the computing system or operating system required by a job or task and including main storage, input/output devices, the central processing unit, data sets, and control and processing programs.

secondary storage: Auxiliary storage.

segment:
1. The smallest functional unit (one or more control sections) that can be loaded as one logical entity during execution of an overlay program.
2. As applied to telecommunications, a portion of a message that can be contained in a buffer of specified size.

service program: Any of the class of standard routines that assist in the use of a computing system and in the successful execution of problem programs, without contributing directly to control of the system or production of results, and including utilities, simulators, test and debugging routines, etc.

source module: A series of statements (in the symbolic language of an assembler or compiler) which constitutes the entire input to a single execution of the assembler or compiler.

supervisor: As applied to the operating system, a routine or routines executed in response to a requirement for altering or interrupting the flow of operations through the central processing unit, or for performance of input/output operations, and, therefore, the medium through which the use of resources is coordinated and the flow of operations through the central processing unit is maintained; hence, a control routine that is executed in supervisor state.

SYSIN: A name conventionally used as the data definition name of a data set in the input job stream.

SYSOUT: An indicator used in data definition statements to signify that a data set is to be written on a system output unit.

system library: The collection of all cataloged data sets at an installation.

SYS1.LINKLIB: The partitioned data set that contains the IBM-supplied processing programs and part of the nonresident portion of the control program. It may also contain user-written programs.

SYS1.PROCLIB: The partitioned data set that contains cataloged procedures.

task: A unit of work for the central processing unit from the standpoint of the control program; therefore, the basic multiprogramming unit under the control program.

task management: A general term that collectively describes those functions of the control program that regulate the use by tasks of the central processing unit and other resources (except for input/output devices).

telecommunications: A general term expressing data transmission between a computing system and remotely located devices via a unit that performs the necessary format conversion and controls the rate of transmission.

teleprocessing: A term associated with IBM telecommunications equipment and systems.

turn-around time: The elapsed time between submission of a job to a computing center and the return of results.

user: Anyone who requires the services of a computing system.

volume: All that portion of a single unit of storage media which is accessible to a single read/write mechanism.

APPENDIX E: WYLBUR REFERENCE MANUAL

Second Edition

February 29, 1968

Campus Facility <u>USERS MANUAL</u>

PREFACE

This second edition of the WYLBUR users manual updates the first edition to reflect the new features which have been added to WYLBUR since its introduction to the Stanford University Community. The developers of the WYLBUR system wish to offer their thanks to the early users who suffered through the many system crashes which occurred while producing the stable system enjoyed today.

WYLBUR was produced by the programming staff of the Stanford Computation Center, Campus Facility. Credits are due all of the Staff of the Campus Facility without whose individual efforts WYLBUR could not have been born or reached its young manhood. Warranting particular mention are Roger Fajman, John Borgelt, John Gilman, Leighton Allen, Janet Gallo, Jim Moore and Bill Riddle upon whose shoulders the bulk of the design, implementation, and documentation fell. We also wish to expess a special word of thanks to the Director of the Stanford Computation Center, Professor E. A. Feigenbaum, for his support during the development of the system.

WYLBUR is a subsystem residing under what is called the Stanford Terminal Processor (STP) and is the first of many terminal oriented facilities we hope to develop here at Stanford.

The design goal was to produce a terminal oriented text editor and remote job entry facility with fairly strong data management facilities to be used as an aid in program development. WYLBUR is not a time sharing facility in the sense that that word means interactive computation. It is, however, a system which provides quick response to many users simultaneously sitting at terminals doing text editing.

Additional power is provided through the integration of STP with HASP which is the spooling submonitor used to control the batch services here at Stanford. Modification to this piece of work provides for the low overhead introduction of programs into the batch stream, in an easy and natural way, from the WYLBUR terminal. Output from batch stream programs can be left on disk for later review from the terminal. While this does not provide time sharing in the formal sense, it does provide a more favorable facility for getting work done than that which can be provided by a batch system alone.

Rod Fredrickson
Associate Director,
Stanford Computation Center,
Campus Facility

WYLBUR

Stanford University
Computation Center
Text Editor

by William E. Riddle

WYLBUR is a Text Editor designed to operate in either an O/S or HASP environment in order to provide users at 2741 terminals a comprehensive text editing facility with prompt response, while otherwise allowing the computer to proceed with processing of the normal batch job stream.

WYLBUR allows the user to change the contents of a line without retyping all of it, to insert, delete or replace lines, to copy and move lines, to renumber lines, and to list lines. In addition, the user may retrieve external data sets in order to work upon them and save data sets which have been constructed. The user may also insert the data set into the monitor's input stream for processing as a batch job.

This manual is intended to serve a dual purpose. First it should serve as a planning guide for potential users -- an introduction to what the user can and cannot accomplish using WYLBUR. Second, it should serve the user while he is operating WYLBUR -- a reference text.

The format of the manual is as follows. In order to make the later discussion more readable, the definitions of all terms are gathered into an initial section. The remaining sections introduce the WYLBUR commands in order of their increasing difficulty. In introducing the commands, the form of the manual is conversational with examples given showing what will happen during a session with WYLBUR when the command under discussion is used. In the examples, all material which is typed at the terminal by WYLBUR is underlined to differentiate it from the typing done by the user. Also, the typing of the non-printing keys ATTN and RETURN are designated by ATTN and CR.

DEFINITION OF BASIC TERMS

Before outlining the set of operations which is available in WYLBUR, it is necessary to define some basic terms which will be needed in the discussion.

The elementary "atomic unit" from which all data sets are ultimately derived is the CHARACTER. The set of valid WYLBUR characters - the WYLBUR CHARACTER SET - consists of those on the keyboard of an IBM 2741 teletypewriter, a picture of which appears in Appendix A. This is a standard typewriter keyboard with a few special purpose keys, the function of which will be explained shortly, added.

The 2741 contains some characters which are not available on the 1403 high speed printers, ! and ¢. These extra characters will be printed as the blank character when a WYLBUR created data set is printed on the 1403.

On a 2741 keyboard there is also a set of keys which do not cause any printing to occur. These include the special-purpose keys which were mentioned previously. Some of these have a special meaning when used under WYLBUR and hence a full description of their function is deferred until the WYLBUR commands are explained. However, a rough description of the function of these <u>NON-TYPING</u> <u>KEYS</u> is given now in order to complete the discussion of the WYLBUR character set.

RETURN - this key returns the printing head to the left-hand margin. In WYLBUR it is used during a session to signal that the user has finished typing a line and wishes WYLBUR to inspect the line and effect any operations which are called for.

BACK-SPACE - this key moves the printing head one space to the left. In WYLBUR this effectively erases this character from the line of text which the user is currently typing.

TAB - this moves the print head forward to the next tab setting. The user must follow a set of very rigid rules, which will be explained subsequently, concerning its use.

CLR and SET toggle switch - this is used to clear and set the tabs. There are also rigid rules concerning the use of this switch which will be explained below.

MAR REL - margin release key used as on a standard typewriter. This key serves no purpose whatsoever in WYLBUR.

SHIFT and LOCK - used to shift into upper case as on a standard typewriter.

ON and OFF toggle switch - since the 2741 is basically an electric typewriter, this switch serves the function of turning on and off the terminal. It should not be used during a WYLBUR session since the results might be disastrous.

ATTN - used during a session to tell WYLBUR that it should suspend whatever it is currently doing for the user and prepare to accept a new directive.

It should be reiterated that the above explanation of the RETURN, ATTN, and BACK-SPACE keys is inadequate for a full understanding of their function in WYLBUR. The full story on the use of these keys will unfold as the various WYLBUR commands are explained, and then be brought together in a succint outline in the appendix.

Characters, the elementary building blocks, are used to construct the following increasing hierarchy of forms.

> Character String - any group of consecutive characters which occurs on one line.
>
> Line - a character string which begins at the left margin and ends with a carriage return.
>
> Range - a group of lines, each of which possesses some distinguishing characteristic.
>
> Data Set - a group of lines which, taken as a whole, forms some logical unit of interest.

The rest of this section is devoted to a fuller description of each of these constructs, their forms and attributes, and their functions in WYLBUR.

<u>CHARACTER STRINGS</u> are the basic unit of interest in WYLBUR since it is these entities which a WYLBUR user generally wishes to manipulate. A character string is represented by a group (of at least one) of consecutive characters from the WYLBUR character set. Blanks are significant - a group of three (say) blanks appears in the string as three blank characters. The significance of blanks also applies to leading (those before any other characters in the string) and terminal (those following all

other characters in the string) blanks.

A LINE is a special character string which begins at the left-hand margin and continues to a carriage return. It is also differentiated from a normal character string in that it forms a logical unit. Two types of lines may be distinguished by the type of logical unit they form. The first is a COMMAND LINE which has the following format

```
        <WYLBUR REQUEST> <USER RESPONSE> CR
```

Essentially, WYLBUR asks the user a question by typing the <WYLBUR REQUEST> prompt at the user's terminal. Then the user completes the line by typing his <USER RESPONSE> answer. As an example, WYLBUR may ask the user a question and then, on the next line, prompt the user for an answer to the question that has been posed

```
      REPLY ?
```

The user answers by completing the line with his answer and then finishing the line with a carriage return

```
      REPLY ? YES CR
```

The second type of line is a LINE OF TEXT -- a logical unit from the data set which the user is working on with WYLBUR's help. The general form of this type of line is

```
    <LINE NUMBER> <SINGLE BLANK> <CONTENT STRING> CR
```

The <LINE NUMBER> is an attribute of the line, used by WYLBUR to determine the position of the line relative to the rest of the lines in the data set. For example, all lines before a line with line number seven will have a line number less than seven and all lines following that line will be numbered greater than seven. A line number is a mixed decimal number with four integer digits and three decimal digits

0013.098   9999.999   0065.100

Leading zeros may be dropped so that

0013.098 and 13.098

are equivalent. Terminal zeros may also be dropped so that

0065.100 and 65.1

are equivalent. Embedded blanks are not equivalent to zeros so that

6 5. 1 and 605.01

are not equivalent. More comments will be made concerning blanks within line numbers when the <SINGLE SPACE> is discussed below. If the line number is an integer, the decimal point need not be typed so that

0098.000, 98.0, 98., and 98

are all equivalent.

The second major part of a line of text is the <CONTENT STRING> (or simply content) of the line. This may be a string of from 0 to 133 consecutive characters. If no characters are present, then the line is a line of blanks. It may be useful to the user to have WYLBUR monitor the number of characters which are contained in the content of a line. Consider, for example, the construction of a data set which is a Fortran program. Since a maximum of 72 characters may be in any line, it is useful to the user to know when he has typed more than that number of characters in a line. For this reason, WYLBUR recognizes an attribute of lines called LENGTH - a count of the number of characters in the line, up to and including the last non-blank character. This attribute is assumed by WYLBUR to have a value of 72 unless the user takes definitive action (by using the SET command described below) to change it to some other value. When a line with more than the designated number of characters is typed, WYLBUR will warn the user that he has exceeded the limit which has been set. The line will still be accepted by WYLBUR and placed in its appropriate place in the data set -- the message that WYLBUR sends is a warning only. The user must change the line so that it conforms to the length which he desires. Note that LENGTH is only an attribute of the contents of a line - the characters used for the line number are not included in the length of a line.

The remaining part of a line of text is the <SINGLE BLANK>. This is used by WYLBUR to recognize the dividing point between the line number and the content of the line. Hence anything appearing before the blank is interpreted as a line number and anything after the blank is considered part of the content of the line. If three (say) blanks follow the line number, WYLBUR will use the first as a signal for the end of the line number and the next two blanks will cause the content of the line to begin with two blank characters. This is the reason that blanks are unacceptable in a line number and that the line of text

    1 0.1 THIS IS A DUMMY LINE

will not be line 100.1 with content

            THIS IS A DUMMY LINE

but rather will be line 1.000 with content

            0.1 THIS IS A DUMMY LINE

RANGE has already been defined as a group of lines which all possess some distinguishing characteristic. The most common instance of this is the _EXPLICIT TYPE OF RANGE_ where the characteristic is that the line number lies within some range of numbers (and hence the name range). This type of range is denoted in WYLBUR by giving two explicit line numbers separated by the character /

1/10

In the example above, the lines of interest are those having a line number greater than or equal to 1.000 and less than or equal to 10.000. A special case is that in which there is only one line in the range

78.098/78.098

which may be abbreviated to just the single line number

78.098

In defining this type of range, the user may employ the special line numbers FIRST and LAST, which have as values the line numbers of the first and last lines in the data set on which the user is working. (F and L are legal abbreviations for FIRST and LAST.) This leads to the following possible ranges

```
          FIRST/10
          1.09/LAST
       or FIRST/LAST
```

As a further convenience the user may also specify the range FIRST/LAST by using the special explicit range specification ALL. It should be emphasized that these special line numbers and the special range ALL are only legal within an explicit range specification.

A second form of an explicit range is just a list of line numbers.

```
F,10.2,75,114.125,126.01
```

WYLBUR requires that there be no more than ten numbers in such a list. Also, it is required that the line numbers are in ascending sequence.

The second type of range which may be specified is the <u>ASSOCIATIVE RANGE</u> in which the distinguishing characteristic is an instance of some <CHARACTER STRING>. This type of range is specified by enclosing the <CHARACTER STRING> within quote marks. A 2741 keyboard has two types of quote marks available - a single quote (') and a double quote ("). Either of these may be used to enclose the <CHARACTER STRING> but consistency must be maintained. Thus the following are valid associative ranges

```
      "VARIABLE"
   or 'VARIABLE'
```

whereas the following is not

```
      "VARIABLE'
```

Also take note that blanks are significant in an associative range specification, so that

```
        "   BLANKS ARE PART OF THE STRING   "
    and "BLANKS ARE PART OF THE STRING"
```

are not equivalent range specifications.

If a quote mark appears in the <CHARACTER STRING>, then the user must follow some additional rules. It is easiest to enclose the <CHARACTER STRING> in quote marks of the opposite type from those appearing in the <CHARACTER STRING>. If the user finds that he must use the same type of quote marks to enclose the string, then he should type those quote marks which appear in the <CHARACTER STRING> as two consecutive instances of the quote mark. For example

```
        "DON'T"
    and 'DON''T'
```

both designate that range of lines which contain the character string

```
        DON'T
```

As an example of the meaning of various associative ranges consider the following text

```
 1.   DON'T WORRY IF YOU HAVEN"T UNDERSTOOD
 2.   EVERYTHING SO FAR.  WYLBUR IS VERY UNDERSTANDING
 3.   AND WILL HELP YOU WHEN YOU MAKE A MISTAKE
 4.   DURING A SESSION BY GIVING YOU SOMEWHAT
 5.   INFORMATIVE ERROR MESSAGES INDICATING
 6.   WHAT WAS WRONG AND SUGGESTING A POSSIBLE METHOD
 7.   OF CORRECTING THE ERROR.  TRYING TO
 8.   OPERATE WYLBUR WILL BE THE BEST WAY
 9.   YOU HAVE OF LEARNING HOW TO USE IT.  BUT
10.   DON'T THINK FROM THIS THAT YOU SHOULD
11.   NOT READ THE REST OF THIS MANUAL BEFORE
12.   ATTEMPTING TO USE WYLBUR.
```

The following associative range specifications give the following sets of lines

```
"DON'T"                             <1,10>
'TH'                                <2,6,7,8,10,11>
' TH'                               <7,8,10,11>
"DON'T WORRY IF YOU HAVEN""T"       <1>
```

The user may wish to restrict the lines included in an associative range to a certain subset of all lines which contain the <CHARACTER STRING>. He may do this by appending a series of digits which specify the ordinal number of each line which is to be in the subset. Thus, using the above example

```
'TH'                                <2,6,7,8,10,11>
'TH' (1,3/5)                        <2,7,8,10>
'TH' (1,3/6)                        <2,7,8,10,11>
```

The user may further restrict the chosen lines by specifying an explicit range in which the search should be made

```
'TH' IN 7/10                        <7,8,10>
'TH' (1,3) IN 7/10                  <7,10>
'TH' IN ALL                         <2,6,7,8,10,11>
```

It should be noted that if the user knows that the string of interest occurs in only a restricted section of his data set, he is doing WYLBUR a service to tell him this additional information. With this extra information, WYLBUR won't need the perhaps lengthy time required to search the full data set.

The user may further restrict an associative range by suffixing the string with column positions within which the string must be contained in any given line.

```
'TH' 6/8                            <2,10>
'TH' 6/8 (1)                        <2>
'TH' 6/8 (1) IN 7/10                <10>
```

Note that the string must begin <u>and</u> end within the specified column positions. If only one column position is specified, the string is required to start in that column position.

'TH' 6 <2>

Blank lines are considered by WYLBUR to be lines with null content. Thus the associative range '' or "" would retrieve all blank lines in the working data set. But the range ' ' or " " would retrieve all lines having one blank <u>and</u> at least one non-blank character, i.e. not blank lines.

The final hierarchical form which may be constructed is the <u>DATA SET</u> - the ultimate logical unit which WYLBUR can be used to construct. A data set is a group of lines, ordered in increasing sequence according to their line number attributes, and corresponds to some logical unit such as a Fortran or PL/I program or merely some data set of text. A data set has many attributes, the first of which is the name of the data set - the data set's <u>DSNAME</u>. The DSNAME can be up to 39 characters long and is of the form

A.B. ... .Z

where A through Z must be character strings of less than eight characters and must begin with an alphabetic character.

The next attribute is the place of residence of the data set. This is called the <u>VOLUME</u> attribute of the data set and denotes the alphameric name of some data set storage device such as a disk. Current issues of the Users' Bulletin give the VOLUME names which are available for use in storing private data sets.

The final attribute of a data set is the FORMAT in which it is written. This may be one of the following forms - EDIT, CARD, or PRINT. It should be emphasized that these formats only apply to the manner in which data sets are stored on external storage devices. When a user is working on one of his data sets with WYLBUR, he is actually working on a copy of the data set and this copy is held by WYLBUR in EDIT format.

For a data set to be in EDIT FORMAT means that the line number attributes are attached to the lines of text and that the content is stored in a special form.

CARD FORMAT consists (as the name implies) of card images - 80 character contents for the lines of text. If any line originally had a length greater than 80 characters, the 81st and greater characters will be lost. Line numbers are not stored with the line contents.

PRINT FORMAT is the same as CARD format, with the single exception that the length of the lines of text is 133 characters.

EDIT, CARD, and PRINT are the standard formats available in WYLBUR. In addition, the user may specify the exact number of characters in each record by using the LRECL type of format

LRECL=80 is equivalent to CARD format
LRECL=133 is equivalent to PRINT format
LRECL=122 would access records each having
  122 characters

The normal WYLBUR user will ordinarily not have to concern himself with the use of LRECL format.

WYLBUR COMMANDS

There are three modes in which WYLBUR can be employed. The first is called COLLECT MODE and is that mode in which the user is having WYLBUR construct a new data set or add lines of text to an existing data set. The second is EDIT MODE in which the user is having WYLBUR make corrections to lines in an existing data set. This mode is the most important one of the three since it is that for which WYLBUR is primarily designed and intended. The third mode is APPLY MODE and is that in which the user asks, through WYLBUR, that some processing be done on the data set which the user has created. The following discussion of the WYLBUR commands is split into three sections corresponding to these three modes of WYLBUR operation.

In the following discussion, the data set upon which the user is making modifications is called the WORKING DATA SET. This is in contrast to EXTERNAL DATA SETS - all other data sets which exist in the system. External data sets are therefore those data sets external to the scope of attention of the user and his terminal.

SIGN ON PROCEDURE

When the user sits down at a terminal, he should first check that the COM-LOC switch on the side of the unit is in the COM position (switch up), enabling the terminal to communicate to the central processor. The other position of this switch, LOC (switch down), puts the terminal into local mode of operation in which it acts as a conventional electric typewriter. Next the user should check to see that the terminal is on by depressing the RETURN key or ATTN key. If nothing happens, this means that the ON-OFF toggle switch is in the OFF position and should be depressed to the ON position. When the terminal is on, depressing the RETURN key or ATTN key causes STP (the Stanford Terminal Processor - a systems program which acts as the interface between the user and the WYLBUR program) to start up the sign-on procedure, during which it asks the user a series of initial questions so that the validity of the user and his account number may be checked.

STP first types

```
STP    NN    MM/DD/YY    HH:MM:SS
```

STP signifies that the Stanford Terminal Processor is processing the information as an interface between the user and WYLBUR. NN shows that the user is at a terminal which is attached to the central processor through a port having number NN. The date is given as month, day, and year - MM/DD/YY - and the current time is given as hour (24 hour clock), minute, and second - HH:MM:SS. STP follows this line with another giving the message of the day - special directions of which the user should be aware. Then STP requests that the user supply his name and the user should answer by typing his last name and a carriage return

NAME ? RIDDLE CR

STP asks the user to identify himself further by giving his account number

ACCOUNT ? T000 CR

The user is then asked to supply a keyword in order to determine that he has a right to use the account number which he just gave

KEYWORD ? FOO CR

(If the user has not set a keyword, then CR - a blank keyword - is the correct response.) As a protection feature, STP overprints both the account number and keyword with a series of miscellaneous characters. Unauthorized use of an account can therefore not be made by a person finding output which has been left in a public area. Provided that the correct response is made the user is asked to supply the number of the terminal which he is using

TERMINAL ? H15 CR

The response to this prompt is found on a label attached to the terminal at the right of the keyboard. The user's response to this prompt is used by STP to collect statistics on the usage of various terminal units.

Associated with each valid account number is a list of <u>ASSOCIATED TERMINAL NUMBERS</u>. These are used for the purpose of insuring that each user can get his fair share of the computer's services. Looking at these lists in the reverse direction, there is a list of valid account numbers associated with each associated terminal number. These lists insure a uniformity of service since STP will allow only one user from each group assigned to an associated terminal number to be signed on at any one time. If a second user attempts to sign on he will receive the message

<u>ASSOCIATED TERMINAL NUMBER ALREADY BEING USED</u>

In this manner, one terminal which is rented for the use of many people, can not block the use of the system by other users who are renting other terminals.

If a user's account number has more than one associated terminal number, then, when he signs on, STP will consider him as using the first (the associated terminal numbers are stored in collating sequence) associated terminal number for his account number which is not already being used. If the user wishes to override this default decision he may explicitly specify an associated terminal number after his account number

<u>ACCOUNT ? </u>T000 H26

It should be reiterated that the user's response to the <u>TERMINAL?</u> prompt is used by STP to direct the collecting of statistics on terminal usage and <u>is not used</u> to determine the associated terminal number which the user is considered to be using.

During the sign-on procedure, the user is conversing with STP. As a final step, STP asks the user which facility is to be used

<u>SYSTEM? </u>WYLBUR

Note that if only one system is available for use, STP does not ask the user this question.

This completes the sign-on procedure and STP now calls in WYLBUR as is indicated by the next prompt

<u>COMMAND ? </u>

by which WYLBUR signals that it is ready to accept directives from the user.

<u>COLLECT MODE OF OPERATION</u>

Now that the user has signed onto the WYLBUR system, let us suppose that he wishes to construct a new data set using WYLBUR. Since every session starts with a blank working data set, what the user wants to do is to input consecutive lines of text into this initially blank working data set. This mode of operation is what is called COLLECTing lines of text. The only thing which a user need do to enter COLLECT mode is give the COLLECT command

<u>COMMAND ? </u>COLLECT

WYLBUR will now prompt the <LINE NUMBER> and <SINGLE SPACE> parts of consecutive lines of text and expect that the user will fill in the <CONTENT STRING> part for each line

```
     1.    ? THIS IS AN EXAMPLE
     2.    ? OF THE PROMPTS AND
     3.    ? ANSWERS WHICH TAKE PLACE
     4.    ? DURING THE COLLECT MODE
     5.    ? OF OPERATION.
```

As a convenience, the COLLECT command may be given in one of two shortened forms. The first (as with most of the WYLBUR commands -- watch for further examples below) is just the first three characters of the command word

```
COMMAND ? COL
```

The second is a command which is just an ATTN character

```
COMMAND ? ATTN***
```

Before considering other features of the collect mode of operation, it is appropriate to explain the function of the non-typing keys when they are used during the typing of lines of text.

Carriage return signals that the user has finished typing the contents into the line of text and wishes WYLBUR to prompt with the <LINE NUMBER> of the next line of text.

Back-spacing over a character erases it from the line of text which is being typed.

```
     1.    ? AN EXMPLE OF USE
                     AMPLE OF THE USE OF THE BACK-SPACE KEY
```

results in a line numbered 1.000 and having content

```
             AN EXAMPLE OF THE USE OF THE BACK-SPACE KEY
```

since the user first typed out to the USE, then back-spaced to the beginning of MPLE and then typed the rest of the line. If the user back-spaces into the <LINE NUMBER> or

<SINGLE BLANK> part of the line of text (both of which were prompted by WYLBUR), he will lose all characters which he may subsequently type into these areas. The example

```
    1.068?
       98  ATTEMPT TO CHANGE THE LINE NUMBER
```

where the user has attempted to change the line number which WYLBUR prompted, will result in a line numbered 1.068 with content

```
           ATTEMPT TO CHANGE THE LINE NUMBER
```

Likewise

```
    1.068?
         ATTEMPT TO OVERWRITE THE LINE NUMBER
```

results in a line numbered 1.068 with content

```
           EMPT TO OVERWRITE THE LINE NUMBER
```

The attention key has two functions during COLLECT mode. If any characters are typed before hitting the ATTN, then the ATTN key signals to WYLBUR that the user wishes to erase the line which has just been partially typed

```
    1.    ?THIS SHOWS WHAT HAPPENS WHEN
    2.    ?THE ATTENTION KEY IS HIT BEFORE
    3.    ?COMPLETING A LINE WITH A
    4.    ?CARRIAGE RETURN ATTN***
    4.    ?
```

The ATTN at the end of line 4.000 causes WYLBUR to disregard what was typed into the line, and WYLBUR prompts again with the same line number. (It should be noted here that whenever characters are typed into a line -- command type as well as a line of text -- and the line is terminated with an ATTN, WYLBUR pays no attention to that line and returns the prompt that was just previously made.)

The second function of the ATTN key during COLLECT mode is to signal to WYLBUR that the user wishes to leave COLLECT mode. To give this signal, the user must type the ATTN key as the only character in the line. WYLBUR will then forget about the line numbered with the last prompt and ask the user for a new command

```
    1.    ? THIS SHOWS WHAT HAPPENS WHEN WYLBUR
    2.    ? IS SIGNALED THAT THE USER WISHES
    3.    ? TO LEAVE THE COLLECT MODE.
    4.    ? ATTN***
COMMAND ?
```

At line 4.000 the user typed ATTN, WYLBUR then responded with the three asterisks, returned to a new line, and asked the user for his next directive. The data set will consist of three lines numbered 1.000, 2.000, and 3.000. A duality should be apparent -- an ATTN command is sufficient to get the user into COLLECT mode and an ATTN answer to a line number prompt while in COLLECT mode is sufficient to remove the user from COLLECT mode.

The above examples show that WYLBUR COLLECTs new lines of text into a blank working data set by beginning with a line numbered 1.000 and obtains line numbers for subsequent lines by adding 1.000. These values are under the control of the user and he may append modifying phrases to the COLLECT command in order to cause WYLBUR to use different values. Directing WYLBUR as follows

```
COMMAND ? COLLECT 2.098
```

will cause WYLBUR to collect lines of text with numbers 2.098, 3.098, 4.098, 5.098, etc. -- start at 2.098 and add

1.000 to get subsequent line numbers. Further modifying the command

```
   COMMAND ? COLLECT 2.098 BY .001
```

causes the COLLECTing of lines of text numbered 2.098, 2.099, 3.000, 3.001, etc. -- start at 2.098 and add .001 to get subsequent line numbers. It should be noted that these modifying phrases may not be appended to the COLLECT command when the ATTN short form is used.

Suppose now that the user has constructed a data set (i.e. that his working data set is not empty), and that he wishes to add more lines of text. Suppose further that the working data set consists of seven lines of text numbered 1.1, 2.1, 3.1, 4.1, 5.1, 6.1, and 7.1. If the user issues a COLLECT directive, WYLBUR will prompt for lines to be added to the end of the working data set

```
   COMMAND ? COLLECT
      8.1   ? COLLECTING MORE LINES INTO THE DATA SET,
      9.1   ? STARTING AT THE END OF THE CURRENT LINES.
     10.1   ? ATTN***
   COMMAND ?
```

The question arises of how WYLBUR derives the starting number of 8.1? The working data set has an attribute, LAST, which is defined to be the line number of the last line of text. 8.1 is therefore derived by taking this LAST attribute and adding the increment which is currently in effect (namely 1.000 unless the user specifies otherwise in the COLLECT command). As before, the user may change the increment

```
COMMAND ? COLLECT BY .1
    7.2  ? COLLECTING MORE LINES, STILL INTO THE END
    7.3  ? OF THE DATA SET, BUT NOW WITH A TEMPORARY
    7.4  ? INCREMENT OF .1
    7.5  ? ATTN***
COMMAND ?
```

The user may also collect lines into the interior of his working data set, but WYLBUR will not allow any existing lines to be overwritten or passed in the process. Consider the working data set as above and the command

```
COMMAND ? COLLECT 2.1 BY .1
```

Since collection of a line numbered 2.1 would cause an existing line to be overwritten, WYLBUR will return an error message and ask for a new command. If, on the otherhand, the command is

```
COMMAND ? COLLECT 2.2 BY .1
```

WYLBUR is quite happy to let the user add lines numbered 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, and 3.0; but will not collect a line numbered 3.1 since it would cause an existing line to be overwritten. Finally, if the command were

```
COMMAND ? COLLECT 2.2 BY .4
```

WYLBUR will collect lines numbered 2.2, 2.6, and 3.0, but will not pass the existing line 3.1 in order to collect a line numbered 3.4.

While COLLECTing lines the user may perhaps enter a line which contains more than LENGTH characters. Normally, WYLBUR will only issue a warning message, informing the user that this has occurred. However, the user may turn on an OVERFLOW option (see the SET OVERFLOW command below) which will cause WYLBUR to place all the extra characters into the next line. Suppose that LENGTH is set at 40 and that OVERFLOW is set on.

```
COMMAND ? COLLECT 90 BY 10
  90.    ? THIS IS A LINE WHICH HAS 44 CHARACTERS IN IT
 100.    ? IN IT AND ADD MORE
 110.    ? ATTN***
COMMAND ? LIST 90/110
  90.      THIS IS A LINE WHICH HAS 44 CHARACTERS
 100.      IN IT AND ADD MORE
COMMAND ?
```

Note first that breaks in the line containing excess characters are made only at blanks and never in the middle of a word. Also note that the overflowed characters are prompted by WYLBUR at the beginning of the next line. This overflow rectification process will work with any number of characters in the line - WYLBUR chooses successive groups of (approximately) LENGTH characters and makes them into new lines and then prompts for the next line, with the extra characters left over already present in the next line. If a string of more than LENGTH characters occurs without any blanks in it, WYLBUR will abort the overflow rectification process with an error message. The rectification done up to that point will be preserved, but the string itself and whatever follows it in the line just entered will be lost.

USING PREVIOUSLY DEFINED DATA SETS

Using WYLBUR would become rather tiresome if the only way in which a user could get a non-blank working data set would be to type in the full text while in COLLECT mode. Therefore, it is possible to give a command to WYLBUR to have it call in a copy of an external data set to use as the working data set. This could be a data set which the user constructed at an earlier session, a data set which has been constructed by another user, or a card deck which has been read into the system (the procedure for accomplishing this last example will be given in an Appendix).

It should first be emphasized that only a copy of the external data set is made available to the user. If the user seriously damages the working data set by making disastrous changes (or if the system dies during the session - thereby destroying all working data sets), a version of the unimpaired data set still exists.

In order to obtain a copy of the external data set, the user should issue a USE command

```
    COMMAND ? USE DUMMY
```

where DUMMY is the DSNAME of the data set which the user desires. The user's account number is appended to the front of the DSNAME to obtain the real name of the data set. In the above example, a data set named T000.DUMMY would be searched for on the external storage devices.

The user may help WYLBUR in its search for the data set by giving the VOLUME attribute

```
    COMMAND ? USE DUMMY ON SYS07
```

WYLBUR will ask the user to supply the format of the data set

```
COMMAND ? USE DUMMY ON SYS07
 FORMAT ? EDIT
COMMAND ?
```

and after getting this information, WYLBUR will get a copy of the data set to use as the working data set for the user. The user may preempt WYLBUR's standard request for the data set format by stating it in the USE command.

```
COMMAND ? USE DUMMY EDIT
COMMAND ? USE DUMMY ON SYS07 EDIT
```

When WYLBUR obtains the copy, it is automatically put into EDIT format if it is not already in that format. This means that for CARD and PRINT format data sets, WYLBUR will attach line number attributes to every line of text. The line numbers supplied will begin with 1.000 and have an increment of 1.000 (the increment may be changed - see section below on the DELTA global parameter).

A user may obtain a copy of a public data set belonging to some other user by giving that data set's full DSNAME preceded by the & character

```
COMMAND ? USE &****.DUMMY
```

where **** must be replaced with the account number of the user who constructed the data set named DUMMY. In no cases will the original copy of the data set be scratched or overwritten by anyone but the user who constructed it.

The user may specify the LRECL type of format

```
COMMAND ? USE DUMMY LRECL=121 ON SYS05
```

But note that LRECL may not be greater than the number of characters in the physical record or the record will be ignored. If the number of characters in the physical record is not an integral multiple of LRECL, those characters remaining after all possible lines of LRECL characters have been inserted into the working data set will be ignored.

When using data sets which are in PRINT, CARD, or LRECL format, the user may specify that a given number of records should be skipped when reading in the data set.

```
COMMAND ? USE DUMMY LRECL=80 SKIP=1000 ON SYS06
```

This will cause the first 1000 card images to be ignored and the card image, that without the SKIP phrase would have been line number 1001, will be line number 1.

GLOBAL PARAMETERS

Global parameters are adjustable features in WYLBUR which may be set to an appropriate value, and will remain set at that value for the duration of the session or until the user resets their value.

The first global parameter is the already explained LENGTH attribute of lines of text. The user may set this to any value between 1 and 133 (inclusive) by use of the SET command

```
COMMAND ? SET LENGTH = 80
```

If the user does not set the LENGTH attribute, it automatically has the value 72.

The second global parameter is the setting of the TABS at the terminal. These are used as on a standard typewriter, but the user must follow a rather rigid procedure for telling WYLBUR what the settings are. This rigid procedure is necessitated by the way in which a 2741 works. When the TAB key is pressed at the terminal, the typing head is released and travels to the right. A protruding bar stops the movement of the typing head at the next tab setting. However, the only signal that is sent to the computer is one which indicates that the TAB key was pressed - no signal is sent to tell the computer where the typing head stopped. Thus the tabs may be physically set at the terminal, but WYLBUR will not know what the settings are until the user completes the following sequence. The use initiates this sequence by asking WYLBUR to set the tabs

```
COMMAND ? SET TABS
```

WYLBUR then gives the user directions and a line of column numbers

```
COMMAND ? SET TABS
TYPE A "1" BENEATH EACH POSITION AT WHICH YOU SET A TAB.
          123456789A123456789B123456789C123456789D12...
  START ?
```

The first thing the user must do is to clear the physical tab settings which already exist at the terminal. This is done by successively tabbing across the line and depressing the CLR toggle switch at each tab until the right-hand margin is reached. Then the user should hit ATTN, which will cause WYLBUR to return to the beginning of the next line and re-prompt START?. The user then spaces over to each successive column in which he wants a tab set, first presses the SET toggle switch, and then types a "1"

```
COMMAND ? SET TABS
TYPE A "1" BENEATH EACH POSITION AT WHICH YOU SET A TAB.
          123456789A123456789B123456789C123456789D12...
  START ?
  START ?          1         1         1         1
```

The above example shows the successful setting of the tabs in columns 10, 20, 30, and 39.

Alternatively, the user may append to the SET TABS command a list of numbers representing the column numbers for the tab settings

```
COMMAND ? SET TABS = 10,20,30,39
COMMAND ?
```

But note that the physical tab settings at the terminal should correspond to the information given when using this form of the SET TABS command.

If the tabs have not been set and the tab key is used, WYLBUR will pay no attention to the line and will issue an error message. This occurs also if not enough tab stops have been set - for example, if only three tab stops have been set and the tab key is pressed four or more times.

The third global parameter is the increment which WYLBUR uses to derive one line number from its predecessor -- DELTA. The user may set DELTA

```
COMMAND ? SET DELTA = XXXX.XXX
```

where XXXX.XXX is any decimal number between .001 and 9999.999 (inclusive). If the user doesn't set DELTA, a default value of 1.000 is used. So far, the only use of DELTA which has been encountered is in the COLLECT mode where it was noted that a DELTA of 1.000 was used unless the user overrode this value by specifying a different value in the COLLECT command (the BY phrase). In actuality, the DELTA used is the current value of the global parameter -- it was said to be 1.000 because that is the default value.

The OVERFLOW global parameter, used only in COLLECT mode, is turned on and off by means of a SET command

```
COMMAND ? SET OVERFLOW = ON
COMMAND ? SET OVERFLOW = OFF
```

OVER is a valid abbreviation for the word OVERFLOW.

The final global parameter is the LAST attribute of a working data set, which the reader should remember is the line number of the last line of text in the data set. The user may not set the value of this parameter explicitly -- whenever new lines are added to the end of a data set, the value of LAST is implicitly and automatically reset.

All examples of the use of WYLBUR are typed in upper case letters in this manual. This is the normal mode in which WYLBUR operates - all alphabetic characters, whether typed in upper or lower case are recognized by WYLBUR as being in upper case. If the user wishes to use both upper and lower case, he must tell WYLBUR

```
COMMAND ? SET UPLOW
```

Then when the user wishes to revert to normal mode, he directs WYLBUR

```
   COMMAND ? SET UPPER
```

Note that regardless of the case mode which is in effect, all typing in a command line is recognized by WYLBUR as upper case except those characters delimited by quote marks (as in associative ranges). Also note that, at present, the 1403 high-speed printers are only capable of printing upper case. Thus any text listed on the 1403 will be printed in upper case regardless of its actual case.

The discussion above indicates the manner in which the global parameters may be set by the user. How can the user find out the values of the parameters if he has forgotten them? This is done by asking WYLBUR to SHOW their values.

```
   COMMAND ? COLLECT
        1.    ? THIS IS AN EXEMPLARY DATA SET USED
        2.    ? TO REVIEW THE SETTING AND SHOWING
        3.    ? OF GLOBAL PARAMETERS.
        4.    ? ATTN***
   COMMAND ? SHOW TABS
   CURRENT TAB SETTINGS ARE NONE.
   COMMAND ? SHOW LAST
   CURRENT LAST LINE NO. IS 3.
   COMMAND ? SHOW LENGTH
   CURRENT LENGTH IS 72.
   COMMAND ? SHOW DELTA
   CURRENT GLOBAL DELTA IS 1.
   COMMAND ? SET DELTA = .01
   COMMAND ? SHOW DELTA
   CURRENT GLOBAL DELTA IS 0.01
   COMMAND ? SET LENGTH = 80
   COMMAND ? SHOW LENGTH
   CURRENT LENGTH IS 80.
   COMMAND ? COLLECT
        3.01 ? EXTRA LINE ADDED
        3.02 ? ATTN***
   COMMAND ? SHOW LAST
   CURRENT LAST LINE NO. IS 3.01
   COMMAND ? SET TABS
   TYPE A "1" BENEATH EACH POSITION AT WHICH YOU SET A TAB.
                 123456789A123456789B123456789C123456789D12...
```

```
    START ?
    START ?          1         1          1        1
COMMAND ? SHOW TABS
CURRENT TAB SETTINGS ARE - 10 - 20 - 30 - 39 -
COMMAND ? SHOW CASE
UPPER CASE ONLY
COMMAND ? SET UPLOW
COMMAND ? ATTN***
     3.02 ? AB
     3.03 ? ab
     3.04 ? ATTN***
COMMAND ? LIST 'ab'
     3.03   ab
COMMAND ? SHOW CASE
UPPER-LOWER CASE
COMMAND ?
```

EDITING OF DATA SETS

This is the heart of WYLBUR --- the primary reason that it was designed and implemented. It is by using these commands that the user may have the computer help him in changing invalid character strings and lines which his data set may contain. Since data sets will most likely be programs, WYLBUR is, in a sense, a glorified keypunch. But as the user reads the rest of this manual, it is hoped that he will see that using this appellation is a gross underestimation, since, when used to its full capabilities, WYLBUR is capable of much more than mere string replacement.

WYLBUR's editing commands are of two distinct types, differentiated by whether the operation called for is effected upon full lines of text or merely upon strings which may appear in the content of a line of text -- INTER-LINE and INTRA-LINE editing.

INTER-LINE EDITING

The most common thing that the user will probably want to do with his data set is to list it. This he can do by commanding WYLBUR to LIST

```
COMMAND ? LIST
     1.       THIS IS A DUMMY DATA SET WHICH SHOWS
     2.       WHAT HAPPENS IN RESPONSE TO A LIST
     3.       COMMAND.  EACH LINE OF TEXT IS PRINTED
     3.1      P
     4.       WITH ITSASSOCIATED LINE NUMBER UNTIL
     5.       ALL THE LINES INTHE DATA SET ARE
     6.       LISTED, AT WHICH POINT WYLBUR PROMPTS
     7.       THE USER FOR ANOTHERDIRECTIVE.
COMMAND ?
```

The user may desire that only an explicit range within the data set be listed

```
COMMAND ? LIST 3/5
     3.       COMMAND.  EACH LINE OF TEXT IS PRINTED
     3.1      P
     4.       WITH ITSASSOCIATED LINE NUMBER UNTIL
     5.       ALL THE LINES INTHE DATA SET ARE
COMMAND ?
```

The user may further wish that his listing not contain the line numbers of the lines of text

```
COMMAND ? LIST UNNUMBERED
THIS IS A DUMMY DATA SET WHICH SHOWS
WHAT HAPPENS IN RESPONSE TO A LIST
COMMAND.  EACH LINE OF TEXT IS PRINTED
P
WITH ITS ASSOCIATED LINE NUMBER UNTIL
ALL THE LINES IN THE DATA SET ARE
LISTED, AT WHICH POINT WYLBUR PROMPTS
THE USER FOR ANOTHER DIRECTIVE.
COMMAND ? LIST 3/5 UNNUMBERED
COMMAND.  EACH LINE OF TEXT IS PRINTED
P
WITH ITS ASSOCIATED LINE NUMBER UNTIL
ALL THE LINES IN THE DATA SET ARE
COMMAND ?
```

Or as an alternative (albeit rather useless in this context) the user may wish to have just the line numbers listed

```
COMMAND ? LIST NOTEXT
     1.
     2.
     3.
     3.1
     4.
     5.
     6.
     7.
COMMAND ? LIST 3/5 NOTEXT
     3.
     3.1
     4.
     5.
COMMAND ?
```

This NOTEXT option is more useful when the user wants to find the line numbers of all lines which contain a certain character string - i.e. those lines in an associative range - a case which is discussed next.

Interesting uses of WYLBUR's listing capabilities arise when associative ranges are used

```
COMMAND ? LIST 'DATA SET' IN ALL
     1.      THIS IS A DUMMY DATA SET WHICH SHOWS
     5.      ALL THE LINES IN THE DATA SET ARE
COMMAND ?
```

The above example shows that WYLBUR can be used to find instances of some character string which is of interest to the user WYLBUR has found all occurrences in the specified range (in this case all) and printed their image for inspection. The user may also restrict the range in which WYLBUR is to search

```
COMMAND ? LIST 'DATA SET' IN 4/7
     5.      ALL THE LINES INTHE DATA SET ARE
COMMAND ?
```

The user could also ask for only the line numbers of the lines containing the string

```
COMMAND ? LIST 'DATA SET' IN ALL NOTEXT
    1.
    5.
COMMAND ?
```

Likewise, the user could append the phase UNNUMBERED instead of NOTEXT and receive the line contents only.

When listing with an UNNUMBERED option, the user may specify a MARKER, an instance of which in the first column of any line in the range will cause the listing to be temporarily suspended. After suspension, the listing may be resumed with a CR command.

```
COMMAND ? LIST ALL MARKER = P UNNUMBERED
THIS IS A DUMMY DATA SET WHICH SHOWS
WHAT HAPPENS IN RESPONSE TO A LIST
COMMAND. EACH LINE OF TEXT IS PRINTED
CR
WITH ITS ASSOCIATED LINE NUMBER UNTIL
ALL THE LINES IN THE DATA SET ARE
LISTED. AT WHICH POINT WYLBUR PROMPTS
THE USER FOR ANOTHER DIRECTIVE.
COMMAND ?
```

Note that the line containing the MARKER in column one is <u>not</u> printed and that the MARKER character is only recognized when it occurs in column one.

After a little use of WYLBUR, it should be apparent to the user that having it list all of a very large data set is not very convenient due to the 15 character per second speed of the 2741. Therefore the user may command

```
COMMAND ? LIST OFFLINE
TYPE YOUR BIN NUMBER FOR OFFLINE LIST.
 BIN NO. ? 333
 XXX IS THE JOB NUMBER FOR YOUR LIST OFFLINE.
COMMAND ?
```

This causes WYLBUR to append the necessary JCL cards to the data set and put the job into the job queue for HASP. The output will be marked as belonging in the bin which the user has specified, and the user may monitor the progress of his job by using the LOCATE command (see below). The user may preempt WYLBUR's standard request for a bin number by supplying it in the command prefaced by BIN or B

```
COMMAND ? LIST OFFLINE BIN 333
```

Normally the user will want the computer time required for the offline list to be charged to the same account number that was used in the sign on procedure. He may specify otherwise by giving an account number in the command prefaced by ACCOUNT, ACCT, or A

```
COMMAND ? LIST OFFLINE BIN 333 ACCT T000
```

If the user doesn't supply this ACCOUNT phrase, WYLBUR assumes that the account number given in the sign on procedure is the correct one to use.

When listing offline, the user may position his output on the printer paper by specifying a count of the number of blanks to be inserted at the left.

```
COMMAND ? LIST OFFLINE (5) B 333
```

will produce output which is shifted 5 positions to the right. The number of blanks specified must lie in the range 1 to 70. In addition, if an unnumbered offline list is done, a zero blank count may be specified thus allowing the user to list a file which contains its own printer control characters. (When a LIST OFFLINE UNNUMBERED (0) is done with a data set that has been created by any of the IBM assemblers, WYLBUR needs to be told with an MC phrase

```
   COMMAND ? LIST OFFLINE UNN (0) MC
```

since the carriage control characters output by the assemblers are not the standard ones used by the 1403 highspeed printers. WYLBUR passes this phrase on to the printers so that they can correctly interpret the carriage control characters.)

In a LIST OFFLINE command, the user may specify a title which will be printed at the top of the first page of the output.

```
   COMMAND ? LIST OFFLINE B 333 'DUMMY - AUG 8, 1967'
```

Note that the TITLE option is only valid in an OFFLINE list, and that there may not be more than 60 characters in the TITLE.

The user may obtain a punched card deck of his working data set by giving WYLBUR a PUNCH command

```
   COMMAND ? PUNCH 2/L UNNUMBERED B 333 ACCT T000
```

If the UNNUMBERED option is not specified, the line numbers will be punched into columns 73 - 80, overwriting whatever was in those columns. The BIN and ACCOUNT options work exactly as in the LIST OFFLINE command. The PUNCH command will automatically produce an offline listing of the cards punched.

The processing called for in both the LIST OFFLINE and PUNCH commands is twofold. First a short job is run to produce the print and/or punch file. Then the created file is printed and/or punched. The user may attach a priority - URGENT, PRIORITY, EXPRESS (only for the execution phase), STANDARD, IDLE, or OVERNITE - to each of these stages. STANDARD is the default priority for jobs entered by the card readers, EXPRESS is the default priority for jobs entered via WYLBUR, and the others are successively higher and lower. Note that EXPRESS is only valid when the execution time for the job does not exceed two minutes, otherwise the priority will be changed to STANDARD. (A fuller explanation of these priority levels and their concommitant charges is in the main text of this User's Manual.) The user specifies these priorities by attaching two phrases to the command.

```
    COMMAND ? LIST OFFLINE P B 333 U
```

Note first that the initial letter of each priority option is a legal abbreviation (except for EXPRESS which is abbreviated X). The above command calls for PRIORITY processing and URGENT printing - printing as soon as a 1403 is available.

The print stage of the processing may be deleted altogether (note that it's senseless to do this in a LIST OFFLINE command) by giving a print priority of NOPRINT.

```
    COMMAND ? PUNCH B 333 O N
```

This would cause the punch job to be run overnight and the offline listing of the cards punched to be deleted.

If both the run and print priorities are to be the same, the user need only specify it once. This single priority will be taken to apply to both stages. If the NOPRINT priority is given as the only priority phrase, this will cause the run priority to be EXPRESS and the print priority to be NOPRINT.

The user may request WYLBUR to change the line numbers of the lines of text in his working data set by giving the NUMBER command

```
COMMAND ? NUMBER
```

This causes the renumbering of the data set, starting at 1.000 and using the current value of the DELTA parameter. The user may specify another number at which the numbering is to begin by giving it in the command

```
COMMAND ? NUMBER 2.065
```

which will cause WYLBUR to start at a value of 2.065 for the first new line number. In addition, the user may temporarily override the setting of the global parameter DELTA and ask WYLBUR to use another value

```
COMMAND ? NUMBER 2.065 BY .01
```

It should be emphasized that this command causes the renumbering of the total data set and that the user may not specify that only a range (either explicit or associative) be renumbered.

The user may delete lines in his data set by giving the DELETE command

```
COMMAND ? DELETE 5.03/6.08
```

(Associative ranges may also be specified in the DELETE command.) This will cause WYLBUR to remove from existence all the lines which lie in the specified range.

Opposite to the DELETE command is the INSERT command

```
COMMAND ? INSERT 13.02
 13.02  ?
```

The user has specified that he wishes to insert a line numbered 13.02 and WYLBUR responds with that number and expects the user to fill in the contents of the line as would be done in COLLECT mode. WYLBUR will not allow the user to insert a line which has the same number as an existing line in the data set. Note that only a single line may be inserted -- the user may not specify that an explicit range of lines be inserted. (Remember that this can be done by COLLECTing the lines into the data set.) If the user attempts to insert a line with the same number as an existing line, WYLBUR prompts with an error message telling the user that he should use the REPLACE command (to be described next).

Frequently, the user will want to delete the contents of a line and then put another line into the data set with the same number but with new contents -- i.e. replace the contents of some line. This he may do in a single step by issuing the REPLACE command

```
COMMAND ? REPLACE 13.06
 13.06 ?
```

which causes WYLBUR to overwrite the old contents of the line specified with the new contents which the user types in in response to the line number prompt. The user may specify that a range of lines be replaced, in which case WYLBUR will prompt successive line numbers from the specified range until the range is exhausted. As an example, suppose the data set has lines numbered 1.02, 1.03, 1.031, 1.032, 1.033, 1.04, and 1.05. Then commanding

```
    COMMAND ? REPLACE 1.02/1.03
```

will result in WYLBUR requesting new contents for lines 1.02 and 1.03. On the otherhand, requesting

```
    COMMAND ? REPLACE 1.03/1.05
```

will cause WYLBUR to prompt for new contents of lines 1.03, 1.031, 1.032, 1.033, 1.04, and 1.05. The user is not allowed to replace the contents of a line which does not exist in the data set - i.e. use the REPLACE command to insert. Rather, he should use the INSERT command.

If the user types just ATTN as the new contents of the line, the old contents of the line are not replaced -- the REPLACE command is aborted.

There are short forms for the DELETE, INSERT and REPLACE commands. The first is by giving only the first three characters of the command word. The second form is the same for all commands and consists of giving the full line of text for the inserted or replaced line as a response to the command prompt from WYLBUR

```
    COMMAND ? 13.06 PUT IN THIS LINE
```

This would cause WYLBUR to put this new line image into the data set in its appropriate place which is determined by the line number. If 13.06 already exists in the data set, then the old line will be replaced; and if 13.06 doesn't already exist, the line will be inserted into the data set. The user should be warned of two possible sources of error in using this command.

1. Be sure to terminate the line number with a single blank. Anything before the first blank will be used as the line number and anything following it will become the contents of the new line.

2. Don't use tabs in typing in the line contents, since the typing of the line number has displaced the characters in the content string by as many spaces as are used for the line number. Tabs will still work, but the user will find the contents displaced to the left of where he desires them.

If the user follows the line number with a single blank and then a carriage return, the specified line will be deleted. (beware !!!)

The user may find it necessary to copy the contents of a range of lines into another section of his working data set and may ask WYLBUR to do this for him. Consider the data set as

```
1.01    THIS IS A DUMMY DATA SET WHICH WILL BE USED
1.02    TO SHOW THE EFFECT OF VARIOUS  COPY COMMANDS.
2.0     RAND
2.01    SAMPLE
2.02    RUNIN
3.0     SRFREPP
6.0     DATA SET
```

Listed below are various COPY commands followed by a listing of the resultant data set. It is assumed that each COPY command is given when the data set is in the form given above.

```
COMMAND ? COPY 2.0/2.02 TO 47
COMMAND ? LIST
     1.01    THIS IS A DUMMY DATA SET WHICH WILL BE USED
     1.02    TO SHOW THE EFFECT OF VARIOUS  COPY COMMANDS.
     2.0     RAND
     2.01    SAMPLE
     2.02    RUNIN
     3.0     SRFREPP
     6.0     DATA SET
    47.0     RAND
    48.0     SAMPLE
    49.0     RUNIN

COMMAND ? COPY 3/7 TO 1.03 BY .001
COMMAND ? LIST
     1.01    THIS IS A DUMMY DATA SET WHICH WILL BE USED
     1.02    TO SHOW THE EFFECT OF VARIOUS  COPY COMMANDS.
     1.03    SRFREPP
     1.031   DATA SET
     2.0     RAND
     2.01    SAMPLE
     2.02    RUNIN
     3.0     SRFREPP
     6.0     DATA SET

COMMAND ? COPY 2.02 TO 4.01
COMMAND ? LIST
     1.01    THIS IS A DUMMY DATA SET WHICH WILL BE USED
     1.02    TO SHOW THE EFFECT OF VARIOUS  COPY COMMANDS.
     2.0     RAND
     2.01    SAMPLE
     2.02    RUNIN
     3.0     SRFREPP
     4.01    RUNIN
     6.0     DATA SET

COMMAND ? COPY 'DATA SET' IN ALL TO 2.03 BY .001
COMMAND ? LIST
     1.01    THIS IS A DUMMY DATA SET WHICH WILL BE USED
     1.02    TO SHOW THE EFFECT OF VARIOUS  COPY COMMANDS.
     2.0     RAND
     2.01    SAMPLE
     2.02    RUNIN
     2.03    THIS IS A DUMMY DATA SET WHICH WILL BE USED
     2.031   DATA SET
     3.0     SRFREPP
     6.0     DATA SET
```

(This example is actually wrong. The reader should try it on WYLBUR and figure out why the correct result occurs. If he can't figure out the reason for the result, he should ask one of the SCC consultants.)

```
COMMAND ? COPY 'DATA SET' IN 1/5 TO 33 BY .001
COMMAND ? LIST
     1.01    THIS IS A DUMMY DATA SET WHICH WILL BE USED
     1.02    TO SHOW THE EFFECT OF VARIOUS  COPY COMMANDS.
     2.0     RAND
     2.01    SAMPLE
     2.02    RUNIN
     3.0     SRFREPP
     6.0     DATA SET
    33.0     THIS IS A DUMMY DATA SET WHICH WILL BE USED
```

The user may copy lines from an external data set by giving a FROM phrase which specifies the external data set

```
COMMAND ? COPY 1/56 TO 35 BY .1 FROM DUMMY
```

This would result in all lines which are numbered between 1.000 and 56.000 in the external data set named DUMMY to be copied into the working data set starting at line 35 and occupying succeeding lines whose line numbers are automatically determined with a DELTA of .1. As with the COLLECT mode, the user is prevented from overwriting or interleaving existing lines in the working data set. Another rule concerning the COPY command is that any external data set which is specified must be in EDIT format. WYLBUR will return an error message if it is not, and the user must then get the external data set into EDIT format in order to reissue the command. This may be accomplished by using the data set (after saving the current working data set), which will cause WYLBUR to convert the data set to EDIT format, and then saving this edited version of the external data set under a new name in

EDIT format.

If the user wishes to copy a range of lines and then delete the old copy of the lines, he may do this with a single command - MOVE. This functions exactly as the COPY command with the following exceptions.

1. The old lines which were copied are deleted.
2. It is not possible to apply the MOVE command to an external data set, since a basic axiom is that the user may take destructive action only upon the current working data set.

Following are some examples of the MOVE command as applied to the exemplary data set used in the explanation of the COPY command

```
COMMAND ? MOVE 3/7 TO 1.03 BY .001
COMMAND ? LIST
    1.01    THIS IS A DUMMY DATA SET WHICH WILL BE USED
    1.02    TO SHOW THE EFFECT OF VARIOUS  COPY COMMANDS.
    1.03    SRFREPP
    1.031   DATA SET
    2.0     RAND
    2.01    SAMPLE
    2.02    RUNIN
COMMAND ? MOVE 'DATA SET' IN 1/5 TO 33 BY .001
COMMAND ? LIST
    1.02    TO SHOW THE EFFECT OF VARIOUS  COPY COMMANDS.
    1.03    SRFREPP
    2.0     RAND
    2.01    SAMPLE
    2.02    RUNIN
   33.      THIS IS A DUMMY DATA SET WHICH WILL BE USED
   33.001   DATA SET
```

The TO phrase of both the COPY and MOVE commands may specify the special line number END. END is defined as DELTA higher than the current LAST of the working data set.

INTRA-LINE EDITING

This completes the discussion of the inter-line editing commands available in WYLBUR. This group is powerful, by itself, as a tool to help the user edit his data sets, but requires that the full contents of the corrected line be typed when making changes. Helping the user make changes such as this in a more convenient manner is the function of the intra-line editing commands.

The first means of modifying the contents of a line is by use of the MODIFY command

```
COMMAND ? MODIFY 23.54
  23.54   THIS LINE IS TOOO BE MODIFID
 ALTERS ?
```

which causes WYLBUR to print out an image of the line as it stands, and then prompt for alterations from the user. The first thing that the user wants to do to the above line is to remove the extra O characters. This is called deleting characters from the content and is accomplished by typing a D underneath the characters to be removed

```
COMMAND ? MODIFY 23.54
  23.54   THIS LINE IS TOOO BE MODIFID
 ALTERS ?                DD
  23.54   THIS LINE IS TO BE MODIFID
 ALTERS ?
```

Actually the user need type a D only under the first and last character of the string which is to be deleted -- everything between the two D's will also be deleted

```
COMMAND ? MODIFY 23.54
  23.54   THIS LINE IS TOOO BE MODIFID
 ALTERS ?                DD
  23.54   THIS LINE IS TO BE MODIFID
 ALTERS ?                       D  D
  23.54   THIS LINE IS TO BE MOD
 ALTERS ?
```

The user may also insert new characters into the line by first typing an I underneath the character before which the insertion is to be made and following this with the character string to be inserted

```
COMMAND ? MODIFY 23.54
  23.54   THIS LINE IS TOOO BE MODIFID
 ALTERS ?                DD
  23.54   THIS LINE IS TO BE MODIFID
 ALTERS ?                       D  D
  23.54   THIS LINE IS TO BE MOD
 ALTERS ?             I NOW
  23.54   THIS LINE IS NOW TO BE MOD
 ALTERS ?
```

Finally the user may replace a string in the line contents by typing an R under the first character to be replaced and following this with a string of characters which are the replacements. As many characters are replaced, starting with the one over the R, as there are characters in the string

```
COMMAND ? MODIFY 23.54
  23.54   THIS LINE IS TOOO BE MODIFID
 ALTERS ?                DD
  23.54   THIS LINE IS TO BE MODIFID
 ALTERS ?                       D  D
  23.54   THIS LINE IS TO BE MOD
 ALTERS ?             I NOW
  23.54   THIS LINE IS NOW TO BE MOD
 ALTERS ?                  RBEING MODIFIED
  23.54   THIS LINE IS NOW BEING MODIFIED
 ALTERS ?
```

Using the R indicator restricts the user to replacing a number of characters with the same number of characters. If the user wants to do a replacement with a different number of characters he may use a combination of the D and the I indicators

```
COMMAND ? MODIFY 23.54
  23.54   THIS LINE IS TOOO BE MODIFID
 ALTERS ?                DD
 23.54   THIS LINE IS TO BE MODIFID
ALTERS ?                      D  D
 23.54   THIS LINE IS TO BE MOD
ALTERS ?             I NOW
 23.54   THIS LINE IS NOW TO BE MOD
ALTERS ?                  RBEING MODIFIED
 23.54   THIS LINE IS NOW BEING MODIFIED
ALTERS ?              D DIIN THE PROCESS OF
 23.54   THIS LINE IS IN THE PROCESS OF BEING MODIFIED
ALTERS ?
```

The D D caused the word NOW to be deleted and I under the blank following the word NOW caused the string following the I to be inserted before the blank. The user need not type the second D to indicate the range of the deletion since the occurrence of the I indicator will automatically cause the deletion range to be terminated in the character position just before the I occurs. The user may also follow the D range with a replacement indicator and a string of replacement characters

```
COMMAND ? MODIFY 23.54
 23.54   THIS LINE IS TOOO BE MODIFID
ALTERS ?                DD
 23.54   THIS LINE IS TO BE MODIFID
ALTERS ?                      D  D
 23.54   THIS LINE IS TO BE MOD
ALTERS ?             I NOW
 23.54   THIS LINE IS NOW TO BE MOD
ALTERS ?                  RBEING MODIFIED
 23.54   THIS LINE IS NOW BEING MODIFIED
ALTERS ?              D DIIN THE PROCESS OF
 23.54   THIS LINE IS IN THE PROCESS OF BEING MODIFIED
ALTERS ?           D                 RHAS BEEN
 23.54   THIS LINE HAS BEEN MODIFIED
ALTERS ?
```

The user signals that he has made all alterations to the line by not typing anything into the ALTERS line but giving only a carriage return

```
ALTERS ? CR
COMMAND ?
```

The above example shows that WYLBUR always types out the modified image of the line after the user has specified the alteration. This may be suspended by typing an N before giving any alteration indicators

```
COMMAND ? MODIFY 23.54
  23.54    THIS LINE IS TOOO BE MODIFID
 ALTERS ?        N         DD
 ALTERS ?
```

The position of the N is of no concern to WYLBUR as long as it appears before any other characters on the alteration line. On the other hand, the user may get a copy of the modified line, as it stands after correction, by typing a few blanks into the alteration line before hitting RETURN (remember that just hitting RETURN causes the modification process to be terminated)

```
COMMAND ? MODIFY 23.54
  23.54    THIS LINE IS TOOO BE MODIFID
 ALTERS ?        N         DD
 ALTERS ?     CR
  23.54    THIS LINE IS TO BE MODIFID
 ALTERS ?
```

The ATTN key is also used during the modification process to signal directions to WYLBUR. First, if it is preceded by any other characters in the alteration line, WYLBUR does not pay any attention to that line (remember the discussion of the similar use of the ATTN key in COLLECT mode)

```
COMMAND ? MODIFY 23.54
  23.54    THIS LINE IS TOOO BE MODIFID
 ALTERS ?                    DDATTN***
 ALTERS ?                  DD
  23.54    THIS LINE IS TO BE MODIFID
 ALTERS ?
```

If on the otherhand, no characters are typed into the alteration line before the ATTN key is hit, WYLBUR will erase all modifications which have been made to the line and suspend the modification process

```
COMMAND ? MODIFY 0.09
     0.09    RENEGE ON MODIFICATIONS ALREADY MADE
  ALTERS ?                      D           D
     0.09    RENEGE ON MODS ALREADY MADE
  ALTERS ? ATTN***
COMMAND ? LIST 0.09
     0.09    RENEGE ON MODIFICATIONS ALREADY MADE
COMMAND ?
```

The user should note that he may make only one alteration per ALTERS prompt with the exception of the two examples previously given -- a deletion indicator followed by either an insertion or replacement indicator.

The user may also modify a range of lines (either associative or explicit) and WYLBUR will operate in the same manner as in the replacement of a range of lines -- it will prompt successive lines in the range until the range is exhausted.

Often the user will wish to modify every occurrence of some string in some area of his data set or perhaps in his entire data set (consider as an example the changing of an implicit integer variable name to an implicit floating point variable name in a Fortran program). He may use the MODIFY command to do this by specifying an associative range and then making the appropriate alterations when each line is brought out for inspection by WYLBUR. Thus if the string to be changed is MAX and it is to be changed to

FMAX, the user could direct

```
    COMMAND ? MODIFY 'MAX' IN ALL
```

and make the changes to each line found and prompted by WYLBUR. He may do the same function in a much easier manner by issuing a CHANGE command to WYLBUR

```
    COMMAND ? CHANGE 'MAX' TO 'FMAX' IN ALL
```

which will cause WYLBUR to search the specified range (in this case, the entire data set) and change every occurrence of MAX in every line in the range to FMAX. WYLBUR will also print the new image of every line which it changes during the process

```
    COMMAND ? CHANGE 'MAX' TO 'FMAX' IN ALL
        1.36    FMAX = 0
       10.01    FMAX = FMAX(A(1),B(I),FMAX)
      999.      FMAX = FMAX/NSAMP
    COMMAND ?
```

If the user does not wish to have the new images printed out for the changed lines, he may append a NOLIST option to the CHANGE command

```
    COMMAND ? CHANGE 'MAX' TO 'FMAX' IN ALL NOLIST
    COMMAND ?
```

The user may wish to restrict the changing of the occurrences to only a section of the data set which he may do by giving a range specification rather than the phrase ALL

```
    COMMAND ? CHANGE 'MAX' TO 'FMAX' IN 999/9999
      999.      FMAX = FMAX/NSAMP
    COMMAND ?
```

The user may further wish to change only a specific occurrence of the string in each line. This is done by modifying the string to be changed with a number giving the ordinal position of the occurrence which is to be changed. Consider the result derived in the next to last example and the case that the user wishes to change the function name back to MAX.

```
COMMAND ? CHANGE 'FMAX' (2) TO 'MAX' IN 10.01
   10.01    FMAX = MAX(A(1),B(1),FMAX)
COMMAND ?
```

The (2) specifies that only the second occurrence of FMAX is to be changed.

The user may be confused by an ambiguity between ordinal positions of character strings to be changed and the ordinal position of the subset of lines in an associative range. Consider the following possible command

```
COMMAND ? CHANGE 'MAX' (2) TO 'FMAX' IN 'MAX' (2/6)
```

What does it mean? First of all WYLBUR is told to direct its attention to the second through sixth lines in the data set which contain an instance of the string MAX. Within this subset of lines, WYLBUR is told to change the second occurrence of the string MAX in each line to FMAX.

Frequently, a user may desire to change the characters in a certain group of character positions within a range of lines. This may be done by specifying the numbers of the first and last character positions in the group to be affected

```
COMMAND ? CH 2/4 TO 'MX' IN 10/999
   10.01 ? FMX = FMAX(A(1),B(1),FMAX)
  999.     ? FMX = FMAX/NSAMP
COMMAND ?
```

If 2/2 had been specified as the group of character positions, only column 2 would have been changed. Further, if only 2 had been specified, the string MX would have been inserted <u>before</u> column 2 in the old line

```
COMMAND ? CH 2 TO 'MX' IN 10/999
  10.01 ? FMXMAX = FMAX(A(1),B(I),FMAX)
 999.   ? FMXMAX = FMAX/NSAMP
COMMAND ?
```

The user may combine the string and column position notations to restrict the change to a string which is contained in a specified group of columns. Using the original example data set given above, consider the following exemplary commands

```
COMMAND ? CH 'MAX' 7/10 TO 'FMAX' IN ALL
   1.36   FMAX = 0
  10.01   FMAX = MAX(A(1),B(I),MAX)
 999.     FMAX = MAX/NSAMP
COMMAND ? CH 'MAX' 11/72 (2) TO 'FMAX' IN 10.01
  10.01   FMAX = MAX(A(1),B(I),FMAX)
COMMAND ?
```

In the first change, every occurrence of MAX in columns 7 through 10 (note that the string must be wholly contained within the specified group of character positions) is changed. In the second, only the second occurrence of MAX which occurs in columns 11 through 72 is changed. The first change could be effected by commanding

```
COMMAND ? CH 'MAX' 7 TO 'FMAX' IN ALL
```

since, if only one column position is specified, the string 'MAX' is restricted to begin in that column position.

APPLY COMMANDS

These commands, as the name implies, are used to request extra processing on the user's working data set. This is directly analogous to the situation in which the user has extra processing done on a card deck by submitting it to the computer through dispatch or the card readers.

The most important such command is the RUN command which directs WYLBUR to put the user's data set into the input stream of the HASP monitor

```
COMMAND ? RUN
```

This is exactly the same as submitting a program through dispatch or the card readers. The data set must be a logically complete unit containing all of the necessary JCL cards just as would be necessary in submitting a run. WYLBUR merely puts the data set into the input stream and informs the user of the job number which has been associated with the data set so that the user may monitor the job's progress.

```
COMMAND ? RUN
256 IS THE JOB NUMBER FOR YOUR RUN.
COMMAND ?
```

When the RUN command is given, WYLBUR automatically makes a copy (in CARD format) of the working data set so that the data set is acceptable to HASP. (The user should be careful not to run a data set which contains lower case characters since they are not converted to upper case.) In addition, the line numbers are automatically put into character positions 73 through 80, overwriting whatever was in these positions. Thus the output from the run will contain the line numbers to facilitate further editing. If the user does not wish the line numbers to overwrite character positions 73 through 80, he must append an UNNUMBERED phrase to the command

```
COMMAND ? RUN UNNUMBERED
```

Run and print priorities may be attached to the RUN command in the same way that they are attached to the PUNCH and LIST OFFLINE commands

```
COMMAND ? RUN URGENT NOPRINT
```

After the job has been submitted through WYLBUR, the user may wish to find out what (if any) processing has been done on the job. Thus he may ask WYLBUR

```
COMMAND ? LOCATE 256
```

and WYLBUR will respond with the current status of the job -- it is waiting, it is finished execution, it is waiting for the printer, it is on the printed, etc. If WYLBUR can not find the job requested, it will tell the user that the job has most likely finished all its associated processing.

In the LOCATE command, the user may also search for a job by its job name rather than its job number

```
COMMAND ? LOCATE RUN1
 JOB 118
 JOB 214
 JOB 226 IS AWAITING EXEC
```

This causes the batch stream to be searched for all jobs which have the name specified by the user. The job number of each is printed and the status of the last one found is determined and printed for the user.

Before running a job, the user may wish to send a message to the operators concerning tape mounts, etc. This is done with a varient of the SHOW command

```
COMMAND ? SHOW OPERATOR 'JOB MANUAL WILL NEED TAPE 231'
```

In general, this message sending capability should be used to communicate the special instructions which the user would put on the job request card of a batch job.

The user may wish to determine what load exists in the system. This is done by asking WYLBUR to show the status of the system

```
COMMAND ? SHOW STATUS
JOB 159 IS BEING PRINTED
JOB 181 IS BEING EXECUTED
JOB   1 IS BEING EXECUTED
112 JOB(S) AWAITING EXEC,    4 HRS 29 MIN ESTIMATED
  1 JOB(S) AWAITING PRINT,     25747 LINES
 40 PERCENT SPOOL DISK(S) UTILIZATION
COMMAND ?
```

This command, coupled with the LOCATE command, gives the user a means of finding his job and determining a reasonable estimate of when it will be executed.

The user may determine the queues waiting for execution and printing, broken down into priority classes, by issuing the appropriate SHOW commands

```
COMMAND ? SHOW RUN
 URGENT 3
 PRIORITY 4
 EXPRESS 20
 STANDARD 35
 IDLE 20
 OVERNITE  25
COMMAND ? SHOW PRINT
 URGENT 3
 PRIORITY 6
 STANDARD 20
 IDLE 35
 OVERNITE  20
COMMAND ?
```

This information will help the user in deciding what priorities to attach to his program.

The second apply mode command is ALIGN. This command causes WYLBUR to process an explicit range in the working data set so that at most LENGTH characters are on each line. Splits in a line are always made at a blank - a word is never split between two lines. In addition, any line which has a special MARKER character in its first column (this MARKER character must be specified in the ALIGN command if this option is to be used) is not aligned and will cause the alignment process to be restarted at that line. In addition to the special MARKER character specified by the user, a blank in column one (as in indented material) causes the alignment process to be restarted.

```
COMMAND ? SET LENGTH = 80
COMMAND ? CLEAR
COMMAND ? COL 11
   11.    ? P   THIS IS A SAMPLE DATA SET USED TO
   12.    ? SHOW THE EFFECTS OF
   13.    ? POSING THE ALIGN
   14.    ? COMMAND TO WYLBUR.
   15.    ? P   THE P MARKERS IN
   16.    ? COLUMN ONE CAUSE THE RESTARTING
   17.    ? OF THE
   18.    ? ALIGNMENT PROCESS. BUT
   19.    ? NOTE THAT ONE OF THE LINES BEGINS WITH A WORD
   20.    ? WHICH BEGINS WITH P SO THAT
   21.    ? THE MARKER MUST BE CHANGED.
   22.    ?   INDENTED MATERIAL
   23.    ?   WON'T BE ALIGNED
   24.    ? ATTN***
COMMAND ? CH 'P ' 1 TO '$ ' IN ALL NOLIST
COMMAND ? SET LENGTH = 30
COMMAND ? ALIGN 19/L MARKER = $
COMMAND ? LIST 8/L
    8.       ALIGNMENT PROCESS. BUT
    9.       NOTE THAT ONE OF THE LINES
   10.       BEGINS WITH A WORD WHICH
   11.       BEGINS WITH P SO THAT THE
   12.       MARKER MUST BE CHANGED.
   13.   ?     INDENTED MATERIAL
   14.   ?     WON'T BE ALIGNED
COMMAND ? ALIGN ALL MARKER = $
COMMAND ? LIST
    1.       $    THIS IS A SAMPLE DATA SET
    2.       USED TO SHOW THE EFFECTS OF
    3.       POSING THE ALIGN COMMAND TO
    4.       WYLBUR.
    5.       $    THE P MARKERS IN COLUMN
    6.       ONE CAUSE THE RESTARTING OF
    7.       THE ALIGNMENT PROCESS. BUT
    8.       NOTE THAT ONE OF THE LINES
    9.       BEGINS WITH A WORD WHICH
   10.       BEGINS WITH P SO THAT THE
   11.       MARKER MUST BE CHANGED.
   12.   ?     INDENTED MATERIAL
   13.   ?     WON'T BE ALIGNED
COMMAND ?
```

Note that the align process causes a complete renumbering of the entire data set starting at 1.000 and using the current DELTA.

A temporary LENGTH, for use only during an execution of the ALIGN command, may be specified in the command

```
COMMAND ? ALIGN ALL LENGTH = 40 MARKER = $
```

If WYLBUR attempts to ALIGN a line which has no blanks and is longer than the current LENGTH, an image of that line will be printed with the appended error message TOO LONG and execution of the align command will be aborted. The line that was too long will be lost from the data set and no renumbering will be done except for those lines which have already been aligned before the error occurred (and these numbers will not follow any prescribed method of determination).

MISCELLANEOUS COMMANDS

The user may destroy any of his external data sets by commanding WYLBUR to scratch them

```
COMMAND ? SCRATCH DUMMY
```

WYLBUR will remove the data set specified from existence. WYLBUR will not permit the user to scratch any data set except those which belong to him.

The user may ask WYLBUR to show the names of all the external data sets which belong to the user

```
COMMAND ? SHOW DSNAMES
     USER DATA SET DIRECTORY
SYS07
  DUMMY    11/13/67
  DIRECT   08/05/42
SYS04
  MANUAL   02/30/68
COMMAND ?
```

and as indicated, WYLBUR will return the DSNAME and VOLUME attributes of all external data sets which belong to the user.

The user may specify that only the DSNAMES of his data sets residing on a particular volume are to be found.

```
COMMAND ? SHOW DSNAMES ON SYS04
     USER DATA SET DIRECTORY
SYS04
  MANUAL   02/30/68
COMMAND ?
```

The current time of day, in hours, minutes, and seconds, may be obtained by issuing a SHOW TIME command

```
COMMAND ? SHOW TIME
19:43:15
```

WHEN FINISHED WITH EDITING A DATA SET

The user will most likely want to save a copy of the data set which he has been using WYLBUR to construct and correct. This is accomplished by directing WYLBUR to save the current version of the working data set with some specified DSNAME

```
COMMAND ? SAVE DUMMY
 FORMAT ?
```

WYLBUR always requests what format the data set is to be saved in, and the user may preempt this standard request by giving the format in the command

```
COMMAND ? SAVE DUMMY CARD
```

If the data set already exists on an external data set storage device, the user will be told and asked if he wishes WYLBUR to scratch the old version of the data set and replace it with the new version currently being saved.

```
COMMAND ? SAVE DUMMY EDIT
'T000.DUMMY' ALREADY EXISTS.  DO YOU WANT TO SCRATCH IT?
 REPLY  ? YES
```

A YES answer causes WYLBUR to scratch the old and save the new version of the data set. A NO answer causes WYLBUR to terminate the processing of the SAVE command and request a new command from the user. The user may preempt the standard request for permission to scratch an existing data set by appending the phrase SCRATCH to the end of the SAVE command.

A data set may be saved as a private file by specifying

PRIVATE.

```
    COMMAND ? SAVE DUMMY ON SYS04 PRIVATE
```

Any data set saved in this manner will be available for access only by a user signed on with the correct account number - no other user may access the data set, even by specifying its full DSNAME. The full DSNAME will be XXXX.*.DUMMY, where XXXX is the user's account number.

The user may designate that the data set be saved on a particular storage device by giving its name in the SAVE command

```
    COMMAND ? SAVE DUMMY ON SYS07 CARD
```

If the user does not specify the desired VOLUME attribute, WYLBUR will put the data set onto any device which has room, and tell the user where the data set was stored.

Normally, line numbers will not be retained when a data set is saved in CARD format. Instead, the data set will be renumbered (from 1.000 by DELTA) when the data set is used again. The user may, however, specify that he wants the line numbers retained

```
    COMMAND ? SAVE DUMMY ON SYS06 CARD NUMBERED
      or
    COMMAND ? SAVE DUMMY ON SYS06 LRECL=80 NUMBERED
```

Note that this option is only valid when 80 byte records are being saved. The line numbers will overwrite the contents of character positions 73-80. If the user has saved a data set with line numbers retained and in 80 byte records, he must explicitly tell WYLBUR this fact when he next uses the data set

```
COMMAND ? USE DUMMY ON SYS06 CARD NUMBERED
```

If this isn't done, the old, retained line numbers will appear in column positions 73-80 and the data set will be renumbered from 1.000 by DELTA. When NUMBERED is specified, the line numbers for the data set are taken as the contents of column positions 73-80 and these columns are replaced with blank characters. If the contents of character positions 73-80 are not numeric line numbers in ascending order, WYLBUR will do its own line numbering. The numbers must be in ascending order and if two lines appear with the same line number, then only the second will be retained in the working data set.

If the data set is being saved in CARD, PRINT, or LRECL format, a blocking factor may be specified, indicating how many lines of length LRECL should be placed into each logical record. The product of LRECL and the blocking factor must be less than or equal to 3520. The blocking factor is specified as an integer enclosed in parentheses.

```
COMMAND ? SAVE DUMMY ON SYS17 LRECL=80 (44) SCRATCH
COMMAND ? SAVE DUMMY1 (10) PRINT ON SYS05
COMMAND ?
```

In the first case above, each block will contain 44 80-byte records. In the second, each block will contain 10 133-byte records.

After the user has saved a copy of the working data set in the desired format, or if he does not wish to keep a copy of the working data set, he may clear the working data set (i.e. return the working data set to its initial empty state) by commanding WYLBUR to clear

```
COMMAND ? CLEAR
```

The user is cautioned against indiscriminate use of this command, since its effect is irreversible (unless, of course, a copy of the data set has been saved). CLEAR only affects the working data set, and the settings of the global parameters are unaffected.

END OF SESSION - LOGOFF PROCEDURE

All the user need do to end the session is issue a LOGOFF command

```
COMMAND ? LOGOFF
```

WYLBUR will reply with various statistics on the session

```
COMMAND ? LOGOFF
ELAPSED TIME = 00:04:32
END OF SESSION
#
```

The elapsed time is the time which the user has been sitting at the terminal. In addition, WYLBUR keeps track of activity at a terminal and if nothing has been typed in at a logged on terminal for a period of five minutes WYLBUR will ask the user to take some action to show that he is still there

```
COMMAND ? ARE YOU STILL THERE...
```

If the user doesn't type something in reply (carriage return is sufficient) then WYLBUR will give him another five minutes and then ask again

```
COMMAND ? ARE YOU STILL THERE...
```

COMMAND ? TYPE SOMETHING OR YOU WILL BE LOGGED OFF.

If no reply is made, WYLBUR will logoff the terminal.

The user is warned that the logoff command includes an implicit clear command -- the current working data set is not saved. The user must save the working data set, if he wishes to, before logging off.

GETTING HELP FROM WYLBUR

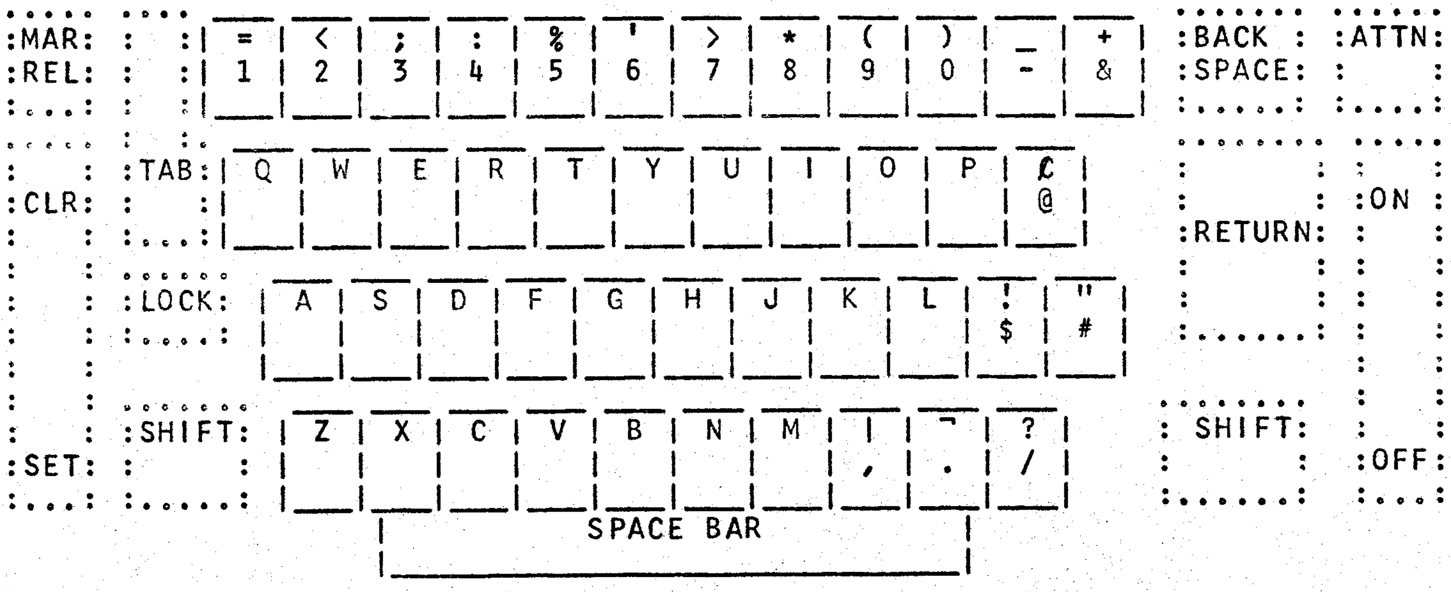
Frequently, the user may want some information or a review of some command or aspect of WYLBUR. For this purpose, the data set T000.WYLHELP may be called. This data set contains short explanations of the commands plus explanations of the use of ATTN, carriage return, line numbers, etc. The following procedure should be used to obtain help.

```
COMMAND ? SAVE ......
COMMAND ? CLEAR
COMMAND ? USE &T000.WYLHELP EDIT
COMMAND ? LIST 'NAME'
          This will be a listing of the first line which
     pertains to information on NAME, which is a command or
     some other aspect of WYLBUR.
COMMAND ? LIST N/LAST
          This listing will be an outline of the information
     which the user desires.  N is the line number of the line
     listed in response to the LIST 'NAME' command.  The user
     can abort the listing when he has information desired.
COMMAND ?
```

The file T000.NEWS contains information on changes and additions that have been made to WYLBUR subsequent to the printing of this manual. As in the WYLHELP data set, a directory to the information is present at the beginning of the data set.

APPENDIX A

2741 TERMINAL PICTURE



APPENDIX B

JCL Card Decks For Various Tasks

```
C  USE THIS JCL TO PUT COMPILER OUTPUT AND GO-STEP OUTPUT
C  ON THE SAME DATA SET.  SUBSTITUTE YOUR OWN ACCOUNT
C  NUMBER, DSNAME AND SPACE PARAMETER IN CARD 5. (1000 IS THE NUMBER OF
C  OF LINES (133 BYTES EACH) EXPECTED WITH  15*(100) MORE ALLOWED).
//STEP1  EXEC  FORTHCLG
//FORT.SYSPRINT  DD  SYSOUT=,DISP=(NEW,PASS),                          X
//               DCB=(RECFM=F,LRECL=133,BLKSIZE=133),                  X
//               VOLUME=SER=SYS04,UNIT=2314,                           X
//               SPACE=(133,(1000,100),RLSE),DSNAME=ACCT.DSNAMEXX
//FORT.SYSIN  DD  *
          YOUR SOURCE DECK
/*
//GO.FT06F001 DD SYSOUT=,DSNAME=*.STEP1.FORT.SYSPRINT,DISP=(MOD,KEEP)
//GO.SYSIN  DD  *
          YOUR DATA
/*
C  ---------------------------------------------------------
C  USE THIS JCL TO PUT COMPILER OUTPUT ON A SEPARATE DATA SET.
C  MODIFY CARD 4 WITH YOUR OWN ACCOUNT NUMBER, DSNAME AND
C  EXPECTED NUMBER OF LINES OF OUTPUT.
//FORT.SYSPRINT  DD  SYSOUT=,DISP=(NEW,KEEP),                          X
//               DCB=(RECFM=F,LRECL=133,BLKSIZE=133),                  X
//               VOLUME=SER=SYS04,UNIT=2314,                           X
//               SPACE=(133,(1000,100),RLSE),DSNAME=ACCT.DSNAMEXX
C
```

```
C  USE THIS JCL TO PUT LINK-EDIT OUTPUT ON A SEPARATE DATA SET.
C  LINK-EDIT OUTPUT MAY NOT BE PUT ON THE SAME DATA SET AS THE
C  COMPILER OUTPUT OR THE GO-STEP OUTPUT.
C  (MODIFY CARD 3 ACCORDINGLY).
//LKED.SYSPRINT  DD  SYSOUT=,DISP=(NEW,KEEP),                        X
//             VOLUME=SER=SYS04,UNIT=2314,                          X
//             SPACE=(133,(1000,100),RLSE),DSNAME=ACCT.DSNAMEXX
C
C  USE THIS JCL TO PUT GO-STEP OUTPUT ON A SEPARATE DATA SET.
C  THIS JCL MAY ALSO BE USED TO PUT BOTH COMPILER AND EXECUTION
C OUTPUT FROM WATFOR ONTO A DATA SET.  THE COMPILER AND EXECUTION
C LISTINGS MAY NOT BE SEPARATED UNDER WATFOR.
C (MODIFY THE LAST OF THE 4 CARDS ACCORDING TO YOUR SPACE AND
C DSNAME REQUIREMENTS)
//GO.FT06F001 DD SYSOUT=,DISP=(NEW,KEEP),                            X
//             DCB=(RECFM=FB,LRECL=133,BLKSIZE=3458),                 X
//             VOLUME=SER=SYS04,UNIT=2314,                           X
//             SPACE=(133,(1000,100),RLSE),DSNAME=ACCT.DSNAMEXX
C -----------------------------------------------------------------
C
C EXAMPLE:
C    AFTER YOU HAVE TYPED YOUR JOBCARD (AS LINE 1), AND YOU WISH TO USE
C    5/10 OF THE FOREGOING, YOU MAY TYPE:
    COMMAND ? COPY 5/10 TO 2 FROM &T000.JCL
    COMMAND ? MOD 6
     6.     //       SPACE=(133,(1000,100),RLSE),DSNAME=ACCT.DSNAMEXX
    ALTERS  ?
C  NOW MODIFY THE ACCT.DSNAMEXX PARM (AND THE SPACE PARM IF NECESSARY).
C
```

```
C  --------------------------------------------------------------
C USE THIS JCL TO PUT CARD DECKS INTO A DATA SET, TO BE
C REFERENCED LATER FROM A WYLBUR TERMINAL.
//STEP1  EXEC  PGM=IEBGENER
//SYSPRINT  DD  SYSOUT=A
//SYSUT2  DD  DISP=(NEW,KEEP),VOLUME=SER=SYS04,UNIT=2314,              X
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3520),                   X
//             SPACE=(TRK,(NN,2),RLSE),DSNAME=ACCT.DSNAMEXX
//SYSIN  DD  DUMMY
//SYSUT1  DD  DATA
         YOUR CARDS (DO NOT INCLUDE ANY /* CARDS)
/*
C
C PUT YOUR ACCOUNT NUMBER AND DSNAME IN THE ACCT.DSNAMEXX FIELD.
C THIS BLOCKING FACTOR ALLOWS 88 CARDS PER TRACK.  SET NN, THE NUMBER
C OF TRACKS TO BE ALLOCATED, ACCORDINGLY.
C  -------------------------------------------------------------------
C  USE THIS JCL TO PUT LISTINGS FROM THE LEVEL F ASSEMBLER ON A DATA
C  SET TO BE REFERENCED LATER FROM TERMINALS.  THE BLOCKING FACTOR ALLOWS
C  APPROXIMATELY 70 LINES PER TRACK ON THE 2314; SET NN, THE NUMBER OF
C  TRACKS TO BE ALLOCATED, ACCORDINGLY.
//STEPNAME  EXEC  ASMFCLG
//ASM.SYSPRINT DD SYSOUT=,VOLUME=SER=SYS14,UNIT=2314,                  X
//             DCB=(RECFM=FB,BLKSIZE=3509,LRECL=121),DISP=(NEW,KEEP),  X
//             SPACE=(TRK,(NN,2),RLSE),DSNAME=ACCT.DSNAMEXX
//ASM.SYSIN  DD  *
         YOUR ASSEMBLER LANGUAGE PROGRAM
/*
```

APPENDIX C

Index and Outline of Commands

This appendix gives a succinct review of all WYLBUR commands as well as references to information in the text of this manual on the commands and other key features of WYLBUR.

For each command, the format is given in outline form under the following rules

> Syntactic types are denoted by enclosure in <...> brackets. Each syntactic type is discussed under its own heading.
>
> Optional phrases are denoted by enclosure in [...] brackets.
>
> Alternative choices are separated by a | character and are enclosed in (...) brackets.

VALID ABBREVIATIONS

BY	none	ON	none
SKIP	none	EDIT	none
CARD	none	PRINT	none
LRECL	none	URGENT	U
PRIORITY	P	STANDARD	S
IDLE	I	OVERNITE	O
NOPRINT	N	EXPRESS	X
OVERFLOW	OVER	COLLECT	COL, C
USE	none	SAVE	none
SCRATCH	none	CLEAR	none
RUN	none	LOCATE	none
LOGOFF	none	LIST	none
NUMBER	none	INSERT	INS
REPLACE	REP	COPY	none
MOVE	none	SET	none
SHOW	none	MODIFY	MOD
CHANGE	CH	PUNCH	none
ALIGN	none	FIRST	F
LAST	L	END	none
LENGTH	LEN	UNNUMBERED	UNN

NOLIST	none	NOTEXT	none
OFFLINE	OFF	BIN	B
ACCOUNT	ACCT,A	MARKER	none
TO	none	FROM	none
DELTA	none	TABS	none
UPLOW	none	UPPER	none
OFF	none	CASE	none
OPERATOR	none	TIME	none
STATUS	none	PRINT	none
IN	none	DELETE	DEL
DSNAMES	none	NUMBERED	none
MC	none	PRIVATE	none

ALIGN COMMAND

ALIGN <EXPLICIT RANGE> [LENGTH=XXX] MARKER=C

This causes the contents of the lines in the specified <EXPLICIT RANGE> to be changed so that as many full words as possible are in a line but such that no line contains more than LENGTH characters. Splits in a line are made only at blanks. After the lines in the specified <EXPLICIT RANGE> are aligned, WYLBUR renumbers the entire data set. One blank is inserted after each word unless it ends with a . or ? or !, which force two blanks. (See page E-59 ff.)

If LENGTH is not specified in the command, the current value of the global LENGTH is used. (See page E-60, E-29.)

The MARKER option must be specified in the command if this feature is desired. When the MARKER character or a blank occurs in column one of any line in the range being aligned, the alignment process is restarted at that line. (See page E-59.)

ASSOCIATIVE RANGE

A group of lines designated by giving a string, an occurrence of which in any line in the data set will signify that that line containing the string is to be included in the range. The form of an associative range is

```
'STRING' [N[/M] ][(MA[,MB ... ,MN])]  IN <EXPLICIT RANGE>
```

(See page E-11 ff.)

Either single quote (') or double quote (") marks may be used to delimit the string, but consistency must be maintained. If the same quote mark appears in the string as is used to delimit the string, the internal occurrence must be repeated twice. Blanks are significant in the string, including leading and trailing blanks. (See page E-11, E-12.)

The IN phrase gives an explicit range in which lines containing an instance of the string are to be searched for. (See page E-13.)

The first set of suffix modifiers indicates column positions within which the string must be contained in any given line. If only the first one of the two column positions is specified, the string is constrained to start in that column. (See page E-13, E-14.)

The second set of suffix modifiers can be of two forms

M   M/N

and serve to to limit the range to a specific subset of lines, namely those whose ordinal positions in the set of all lines containing the string are given by the modifiers. Thus (3) and (3/7) restrict the subset to the third and third through seventh lines, respecively, which have an occurrence of the string in the specified explicit range. In other words, if a line is found in the specified explicit range which contains an instance of the string, then that line is included in the associative range only if it is the Ith line containing the instance and the I modifier is given. If no modifiers are given, all lines containing the string and in the specified explicit range are included in the associative range. (See page E-13.)

The correct associative range to retrieve blank lines is the null string - '' or "". Associative ranges consisting of only blanks (e.g. '   ' or "   ") will retrieve lines which have the specified number of blanks <u>and</u> also have at least one non-blank character - blank lines will not be retrieved. (See page E-14.)

<u>ATTN KEY</u>

Typed as other than the first character in a line, this signals that WYLBUR should not pay any attention to the line. (See page E-22, E-51.)

Typed as an answer to any prompt, this signals that WYLBUR is to abort execution of the command. (See page E-23, E-43, E-51, E-52.)

Typed while a listing is being done, this key suspends the rest of the listing.

Typed while WYLBUR is putting out a message at the user's terminal, this suspends the printing of the rest of the message.

Given as an answer to a COMMAND ? prompt, this key puts the user into collect mode. (See page E-21.)

BACK-SPACE KEY

Back-spacing over any character typed by the user erases it from the line. This applies to command as well as text lines. (See page E-21 ff.)

CARRIAGE RETURN

This must be the final character of every line. It signals to WYLBUR that it should look at the line and take appropriate action.

Given as the only alteration to be made to a line (in the MODIFY command), it signals that all modifications have been made to the line. (See page E-50.)

CHANGE COMMAND

This command allows the user to change a specified character string in all lines which are in some specified <RANGE> in his working data set.

```
CHANGE 'STRINGA' [P[/Q]][(N)] TO 'STRINGB' IN <RANGE>
```

The specified <RANGE> is searched for lines containing an instance of STRINGA. In each of these lines, the specified occurrence of STRINGA is changed to STRINGB. (See page E-52 ff.)

The P and Q modifiers restrict the instance of STRINGA to be wholly contained in columns P through Q of the line.

If only P is specified, STRINGA is restricted to start in column P of the line. (See page E-55.)

If the N modifier is not given, all occurrences of STRINGA in the line are changed. N=3 would specify the third occurrence in the line. (See page E-53.)

```
CHANGE X[/Y] TO 'STRINGB' IN <RANGE>
```

In this alternative form, changes are made to columns X through Y of each line in the <RANGE>. If X and Y are both specified then the characters in positions X through Y are replaced by STRINGB. If only X is specified, STRINGB is inserted before character position X. (See page E-54 ff.)

CLEAR COMMAND

This command scratches the current <WORKING DATA SET>. The settings of global parameters are unaffected.

```
CLEAR
```

(See page E-64.)

COLLECT MODE

Mode in which a series of lines are entered into the data set with WYLBUR calculating and prompting successive line numbers.

```
COLLECT [XXXX.XXX] [BY YYYY.YYY]
```

(See page E-20 ff.)

Lines are collected into the data set beginning with line XXXX.XXX, and deriving successive line numbers by using a DELTA of YYYY.YYY. If YYYY.YYY isn't specified, the current value of the global DELTA is used. If XXXX.XXX isn't specified the starting number is DELTA higher than the LAST (number of the last line) of the data set. (See page E-24.)

The COLLECT command will not allow overwriting or interleaving existing lines in the data set. (See page E-25.)

If the OVERFLOW global parameter is set ON, lines containing more than LENGTH characters will be truncated at the last blank occurring before LENGTH characters and the remaining characters will be prompted, with the line number, as the start of the contents of the next line to be COLLECTed. (See page E-25 ff.)

COPY COMMAND

This command allows the user to copy a <RANGE> to another spot in the <WORKING DATA SET>.

```
    COPY <RANGE> TO (XXXX.XXX|END) [BY YYYY.YYY]
                         [FROM <DSNAME> [ON <VOLUME>]]
```

(See page E-44 ff.)

A copy of the specified <RANGE> is placed in the <WORKING DATA SET>, beginning at a line numbered XXXX.XXX and deriving successive line numbers by using YYYY.YYY

If YYYY.YYY is not specified, the default value is the current working DELTA. If the value of DELTA (specified or default) would cause the copied lines to interleave existing lines in the data set, WYLBUR automatically calculates a new value of DELTA so that interleaving does not take place. (See page E-46.)

The special line number END is defined as DELTA higher than the current LAST. (See page E-47.)

Existing lines in the data set may not be replaced by using the COPY command.

The <RANGE> is taken from the <WORKING DATA SET> unless the FROM option is present, in which case, the <RANGE> will be taken from the named <EXTERNAL DATA SET>, which must be in EDIT format. (See page E-46.)

There is an anomaly when a COPY is done from an <EXTERNAL DATA SET>. This occurs only when the <RANGE> is specified as FIRST, FIRST/FIRST, LAST, or LAST/LAST -- the first or last line of the <EXTERNAL DATA SET> will <u>not</u> be retrieved correctly.

<u>DELETE COMMAND</u>

This allows the user to delete a <RANGE> in the <WORKING DATA SET>.

```
DELETE <RANGE>
```

(See page E-41.)

An alternate form may be used. This consists of giving an <LINE NUMBER> followed immediately by a carriage return as the command. (See page E-43.)

<u>DSNAME</u>

This is the name of a data set. Any identifier is legal as long as it has no more than eight characters and begins with an alphabetic character. WYLBUR automatically appends the user's account number to the beginning of the DSNAME - e.g. DUMMY becomes T000.DUMMY for a user with account number T000. If a user wants to obtain a copy of another user's data set, he must give the full name preceded by & - e.g. &T000.DUMMY. (See page E-14.)

EXPLICIT RANGE

A group of lines designated by giving the upper and lower limits on the line number.

XXXX.XXX/YYYY.YYY

XXXX.XXX must be less than or equal to YYYY.YYY. If XXXX.XXX = YYYY.YYY, then the range may be specified by giving only the single number. (See page E-10 ff.)

The first line of the <WORKING DATA SET> may be referred to by the implicit reference FIRST. Similarly, LAST will reference the last line. The special explicit range ALL is equivalent to FIRST/LAST. (See page E-10.)

An alternative form of an explicit range is just a list of (ten or fewer) line numbers, e.g.

10.01,20,12,125,199.015

The numbers in the list must be in ascending order. (See page E-11.)

EXTERNAL DATA SET

Any data set external to the current scope of attention of a user. (See page E-16.)

FORMAT

The storage configuration of a data set. EDIT format has line numbers stored with the text in a special form. CARD format has lines stored as 80 byte records (i.e. card images) without line numbers unless the user directly tells WYLBUR to store the line numbers also. PRINT format has lines stored as 133 byte records without line numbers. LRECL=XXX has lines stored as XXX byte records - XXX must lie in the range 1 to 133 - without line numbers unless XXX=80 and the user specifies that line numbers are to be stored. (See page E-15 ff.)

INSERT COMMAND

This command allows the user to insert a line of text into the working data set. Only one line may be inserted -- insertion of more than one line must be done by using the COLLECT command.

```
    INSERT <LINE NUMBER>
```

This command causes WYLBUR to prompt with the specified <LINE NUMBER>, after which the user should type in the contents of the new line. (See page E-42.)

The specified <LINE NUMBER> may not already exist in the <WORKING DATA SET>.

Alternatively, insertion may be accomplished by giving a command consisting of the <LINE NUMBER>, followed by a single blank character, and the contents of the new line of text. In this alternative form, if the specified line already exists in the <WORKING DATA SET>, then the old contents will be overwritten -- a replacement is done rather than an insertion. (See page E-43.)

LINE NUMBER

Any number of the form XXXX.XXX between .001 and 9999.999. Leading and terminal blanks may be dropped. The decimal point may be dropped in the case that the number is an integer.

```
    0065.000  =  65.000  =  65.  =  65
```

(See page E-7 ff.)

LIST COMMAND

This command allows the user to obtain a listing of part or all of the <WORKING DATA SET>.

```
    LIST [<RANGE>] [UNNUMBERED] [NOTEXT] [MARKER=C]
      [OFFLINE BIN YYY [(N)] [ACCOUNT XXXX] ['TITLE']
        [<RUN PRIORITY>] [<PRINT PRIORITY>] [MC]]
```

(See page E-35 ff.)

If no options are specified, the entire data set is printed. (See page E-35.)

The <RANGE> phrase limits the listing to that of some specified set of lines. (See page E-35.)

The UNNUMBERED option produces a listing containing only the text of the lines in the <RANGE>. (See page E-35, E-37.)

The NOTEXT option suppresses printing of the text and the listing will consist of only the line numbers of the lines in the specified <RANGE>. (See page E-36.)

The effect of the MARKER option is to cause the listing to be suspended whenever a line is reached which contains the MARKER character, C, in column one. The suspended listing is restarted by giving a CR command, and an ATTN command while the listing is suspended will cause the rest of the listing to be aborted. Note that the MARKER option has no effect on an OFFLINE listing or when UNNUMBERED is not specified. (See page E-37.)

The OFFLINE modifier signals that the user wishes a printing of his data set on the 1403 highspeed printers. The user must tell WYLBUR his bin number, YYY. In addition, he may have the charges incurred put onto Account XXXX and if this isn't specified, the charges will be put onto the Account with which the user logged on. (See page E-38.)

The (N) phrase causes the insertion of N blanks at the left on the 1403 output, thus allowing the user to center his output on the paper. If N=0, the WYLBUR listing routine does not put any carriage control characters in print position one and it is assumed that the data set contains the carriage control characters (be careful that the UNNUMBERED option is specified when N=0 so that line numbers are not picked up as carriage control characters). (See page E-38.)

If the TITLE is specified in an OFFLINE list, it will be printed as the heading for the first page of the output.

The TITLE may not have more than 60 characters. (See page E-38.)

The <RUN PRIORITY> and <PRINT PRIORITY> specify the priority that the user wishes to have attached to these two stages of the processing of his job. (See page E-39 ff.)

The MC option is only valid when an UNNUMBERED OFFLINE (0) list is being done with a data set created by one of the IBM assemblers. (See page E-39.)

LOCATE COMMAND

This command allows the user to inquire about the status of any job which he has put into the job queue through WYLBUR. This may be a job created by use of the RUN command or the LIST OFFLINE command.

```
LOCATE XXX
LOCATE JOBNAME
```

XXX is the job number which has been assigned to the user's job. (See page E-57.)

In the second form of the command, the batch stream is searched for all jobs having the specified JOBNAME. The job number of each job found will be printed out and the status of the last one found will be determined and printed for the user. (See page E-57.)

LOGOFF COMMAND

This command allows the user to terminate the session. The <WORKING DATA SET> is not automatically saved -- LOGOFF includes an implicit CLEAR.

LOGOFF

(See page E-65.)

MODIFY COMMAND

The MODIFY command allows the user to alter the contents of each line in a specified <RANGE>. Successive lines are prompted for alteration until the <RANGE> is exhausted.

MODIFY <RANGE>

(See page E-48 ff.)

For each line in the specified <RANGE> the following process is followed. First the image of the line as it now stands is printed out for the user. Then the user is prompted to specify an alteration to be made to the line. Alterations are one of the following

Insert - I

The letter I is typed immediately below the character in the line before which characters are to be inserted. This indicator is followed by the string of characters which are to be inserted. The string of characters to be inserted is terminated by a carriage return. (See page E-49.)

Replace - R

This indicator is typed immediately below the first character to be replaced in the line. The string of replacement characters is typed immediately following the indicator. As many characters are replaced as there are characters in the string of replacement characters. (See page E-49.)

Delete - D

A string of characters to be deleted from the line is delineated by typing a D under each successive character to be removed. The user need only type a D underneath the first and last character in the string to be deleted -- all characters between the two D's are also deleted. (See page E-48.)

The deletion range may be terminated by typing either an I or R indicator. If terminated by an R indicator, the characters following the deleted string are replaced by the replacement string that the user specifies. If the deletion range is terminated by an I indicator, the effect is replacement of the deleted characters with the string to be inserted. (See page E-50.)

After the alteration is specified, the user should type a return. WYLBUR will make the indicated alterations in the line and type the new image of the line out for the user to inspect. Then WYLBUR will prompt for more alterations. The user signals that all alterations have been made by typing only a carriage return in answer to the alteration prompt. This will cause WYLBUR to put the new image of the line into the data set and prompt for alterations to the next line in the <RANGE>. (See page E-50.)

The printing of the altered image of the line may be suspended by typing an N indictor into the alteration line before any other alteration indicators. (See page E-51.)

The user may have WYLBUR type out the image, as it currently stands, of the line being modified by typing a few blanks into the alteration line and issuing a carriage return. (See page E-51.)

The user may renege on all alterations made to a line by typing an ATTN as the first character in the alters line. WYLBUR will forget about all alterations which the user may have made to the line presently being altered and the image of the line will remain what it was before alterations were made. (See page E-52.)

MOVE COMMAND

This command moves a specified <RANGE>. It is essentially a COPY command followed by a DELETE command, where the old instances of the copied lines are deleted.

```
MOVE <RANGE> TO (XXXX.XXX|END) [BY YYYY.YYY]
```

(See page E-47 ff.)

If YYYY.YYY is not specified, the current value of the global parameter DELTA is used in its place. (See page E-31.)

The special line number END is defined as DELTA higher than the current LAST. (See page E-47.)

The user may not MOVE lines from an <EXTERNAL DATA SET> into the <WORKING DATA SET>. (See page E-47.)

NUMBER COMMAND

This permits the user to have his entire data set renumbered.

```
NUMBER [XXXX.XXX] [BY YYYY.YYY]
```

(See page E-41.)

XXX.XXX is the new number for the first line in the data set. If it is not specified, 1.000 is used. (See page E-41.)

YYYY.YYY is the DELTA to be used in deriving successive line numbers. If it is not specified, the current value of the global parameter DELTA is used. (See page E-31.)

PRINT PRIORITY

The priority to be attached to the print phase of a job. Must be one of the following, which are in decreasing order

```
URGENT
PRIORITY
STANDARD
IDLE
OVERNITE
NOPRINT - don't do the print phase
```

STANDARD is the default priority. (See page E-39 ff.)

PUNCH COMMAND

This command allows the user to produce a punched card deck of a <RANGE> in the <WORKING DATA SET>.

```
PUNCH <RANGE> [UNNUMBERED] [ACCOUNT XXXX] [BIN YYY]
      [<RUN PRIORITY>] [<PRINT PRIORITY>]
```

(See page E-39.)

The ACCOUNT and BIN options are the same as in the LIST OFFLINE command. (See page E-38.)

If the UNNUMBERED option isn't specified, the line numbers will be punched into columns 73-80 of the cards. (See page E-38.)

The PUNCH command automatically produces an offline listing of all the cards that are punched unless a <PRINT PRIORITY> of NOPRINT is specified. (See page E-38.)

The <RUN PRIORITY> and <PRINT PRIORITY> specify the priority that the user wishes to have attached to these two stages of the processing of his job. (See page E-39 ff.)

RANGE

This is either an <EXPLICIT RANGE> or an <ASSOCIATIVE RANGE>.

REPLACE COMMAND

This allows the user to replace the content string of any specified <RANGE> in the <WORKING DATA SET>.

REPLACE <RANGE>

(See page E-42 ff.)

WYLBUR will prompt with successive line numbers in the specified <RANGE>, expecting the user to type in the new contents after each prompt. This is continued until the <RANGE> is exhausted. (See page E-43.)

Alternatively, the user may replace the contents of a single line by giving the <LINE NUMBER>, followed by a single blank, and then the new contents of the line. (See alternative form of the INSERT command.) (See page E-43.)

RUN COMMAND

This command allows the user to put the <WORKING DATA SET> into the HASP job input stream of the 360/67

RUN [UNNUMBERED] [<RUN PRIORITY>] [<PRINT PRIORITY>]

(See page E-56 ff.)

The data set will be converted to CARD format (remember that the <WORKING DATA SET> is always in edit format while being worked upon). (See page E-56.)

If the UNNUMBERED option is not given, the line numbers of the lines of text will be put into columns 73-80. (See page E-56.)

The <RUN PRIORITY> and <PRINT PRIORITY> specify the priority that the user wishes to have attached to these two stages of the processing of his job. (See page E-57.)

RUN PRIORITY

The priority to be attached to the execution phase of a job. Must be one of the following, which are in decreasing order

```
URGENT
PRIORITY
EXPRESS
STANDARD
IDLE
OVERNITE
```

STANDARD is the default priority for jobs entered via the card readers and EXPRESS is the default priority for jobs entered via WYLBUR. See the fuller explanation of priorities in the main text of the user's manual. (See page E-39 ff.)

SAVE COMMAND

The SAVE command tells WYLBUR to save the current <WORKING DATA SET>, having a name which the user specifies, on some external storage device.

```
SAVE <DSNAME> [ON <VOLUME>] [<FORMAT>] [(N)]
        [SCRATCH] [NUMBERED] [PRIVATE]
```

(See page E-61 ff.)

The data set will be saved with the expanded name

XXXX.<DSNAME>, where XXXX is the user's account number.

If the <FORMAT> is not specified, WYLBUR will ask the user what <FORMAT> he desires. (See page E-62.)

The (N) phrase specifies a blocking factor - the number of lines which should be placed into each record. (See page E-64.)

If an old copy of the data set already exists on the storage device, WYLBUR will ask the user whether he wishes the old copy scratched before storing the new copy, unless the SCRATCH option is specified. (See page E-62.)

If no <VOLUME> is given, WYLBUR will store the data set on the first available storage device which it finds. (See page E-63.)

The NUMBERED option is only valid when <FORMAT> is CARD or LRECL = 80 and causes the line numbers to be retained as the contents of character positions 73-80. (See page E-63.)

If the data set is saved with a PRIVATE option, then only a user signed on with the appropriate account number will be able to access the data set. The full DSNAME will be XXXX.*.DSNAME, where XXXX is the user's account number. (See page E-22.)

SCRATCH COMMAND

This command allows the user to scratch any <EXTERNAL DATA SET> which belongs to him.

```
SCRATCH <DSNAME> [ON <VOLUME>]
```

(See page E-61.)

SET COMMANDS

SET DELTA = XXXX.XXX

This sets the value of the global DELTA, and hence the default value to be used in all commands where a DELTA may be specified. (See page E-31.)

XXXX.XXX may be any value between .001 and 9999.999 (inclusive).

DELTA has a value of 1.000 unless the user specifies otherwise.

SET TABS [=NA[,NB ... ,NN]]

This allows the user to inform WYLBUR of what tab setting are in effect at the terminals. WYLBUR will prompt with directions telling the user what to do. If no tabs have been set (or if not enough tabs have been set - e.g. only three set and the tab key pressed four or more times) any use of the tab key will cause WYLBUR to forget about processing the line and issue an error message. (See page E-29 ff.)

SET LENGTH = N

ıhe LENGTH is the number of characters which may be in any line of text. If the user types in a line of text containing more than this number of characters, WYLBUR will accept the line but issue a warning message to tell the user that he has exceeded the limit which was specified. (See page E-29.)

N may be an integer between 1 and 133 (inclusive).

LENGTH has a value of 72 unless the user specifies otherwise.

```
SET UPLOW
SET UPPER
```

These commands govern the case of alphabetic characters. Normally, all alphabetic characters are recognized as upper case. If UPLOW is specified, full upper and lower case facilities are available to the user. If UPLOW is in effect, the user may revert to the normal case (all alphabetic characters in upper case) by giving the SET UPPER command. (See page E-32 ff.)

```
SET OVERFLOW = (ON|OFF)
```

This command turns on and off the OVERFLOW option for COLLECT mode. (See page E-32.)

SHOW COMMANDS

```
SHOW DELTA
SHOW LENGTH
SHOW TABS
SHOW CASE
SHOW OVERFLOW
```

These commands allow the user to find out the current value or any of the global parameters. (See page E-33 ff.)

```
SHOW LAST
```

LAST is the number of the last line in the <WORKING DATA SET>. (See page E-33.)

```
SHOW DSNAMES ON <VOLUME>
```

This will result in a listing of the <DSNAME>s of all <EXTERNAL DATA SET>s which belong to the user. If the <VOLUME> is specified, the DSNAMES of data sets belonging to the user and residing only on the specified volume will be retrieved. (See page E-61.)

SHOW OPERATOR 'STRING'

The STRING is sent as a message to the 360 operator. This may be used to communicate set up information. (See page E-57.)

SHOW TIME

The current time, in hours, minutes, and seconds, is printed for the user. (See page E-61.)

SHOW STATUS

The current load of work on the 360, in terms of the jobs being executed and the jobs awaiting processing, is printed out for the user. (See page E-58.)

SHOW RUN

SHOW PRINT

These commands produce a listing of the queues, broken down into the various priority classes, awaiting execution and printing, respectively. (See page E-58.)

SIGN-ON PROCEDURE

This is the dialogue carried on between the user and the Stanford Terminal Processor and is used to determine the validity of the user's account number and the right of the user to employ WYLBUR. (See page E-17 ff.)

USE COMMAND

The USE command tells WYLBUR that the user wants to work on an <EXTERNAL DATA SET> and that WYLBUR is to get a copy of it so that modifications may be made.

```
     USE NAME [ON <VOLUME>] [<FORMAT>] [SKIP XXXX] [NUMBERED]

        where NAME is just <DSNAME> for a data set belonging
                          to the user
                   is &XXXX.<DSNAME> for a data set belonging
                          to a user with Account number XXXX
```

(See page E-26 ff.)

If the user does not specify the <FORMAT>, WYLBUR will ask that it be specified in answer to a prompt. (See page E-28.)

The SKIP option allows the user to skip over XXXX records at the beginning of the data set being used. The first line of the <WORKING DATA SET> will become the (XXXX+1)th line of the <EXTERNAL DATA SET>. (See page E-29.)

The NUMBERED option is only valid when <FORMAT> is CARD or LRECL = 80 and causes the line numbers to be taken as the contents of character positions 73-80 and these character positions are blanked out. (See page E-63.)

VOLUME

These are names of the storage devices used to hold WYLBUR data sets - e.g. SYS07 or SYS04. See current issues of the User's Bulletin for the names of volumes which are available to the WYLBUR user. (See page E-14.)

WORKING DATA SET

The data set which constitutes the user's scope of attention and upon which the user is affecting changes. (See page E-16.)