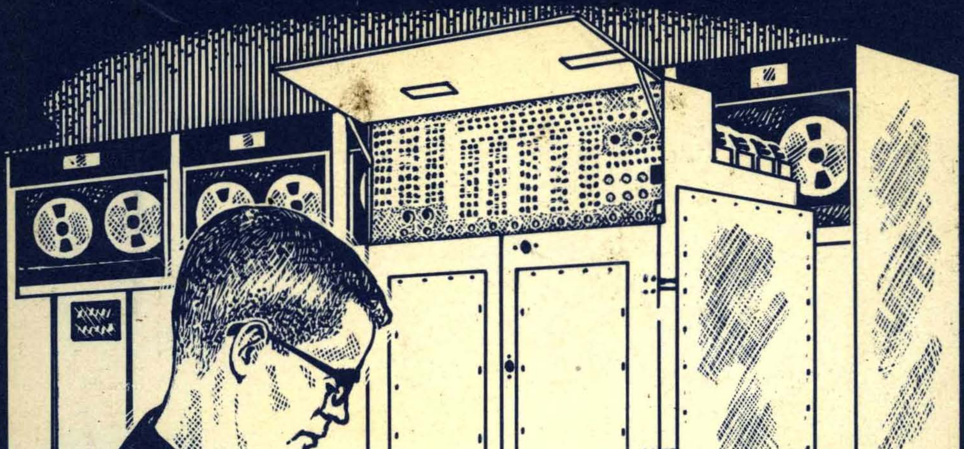


DATA SYSTEMS TECHNICIAN 3 & 2

BUREAU OF NAVAL PERSONNEL

NAVY TRAINING COURSE NAVPERS 10201

DATA SYSTEMS TECHNICIAN 3 & 2

NAVPERS 10201

PREFACE

This training course is written for men of the U.S. Navy and Naval Reserve who are interested in qualifying for advancement to Data Systems Technician third class and Data System technician second class. Combined with the necessary practical experience, this training course will aid the DS in preparing for the advancement-in-rating examination.

The qualifications for the DS Rating are included in section 3 of the Manual of Qualifications for Advancement in Rating, NavPers 18068-A. This training course contains information on each examination subject; and insofar as it is practical in a training course, information is also included on each practical factor. Because examinations for advancement in rating are based on these qualification, interested personnel should refer to them for guidance. The latest qualifications for advancement in rating should always be consulted.

The DS 3&2 training course is prepared by the Navy Training Publications Service, Washington, D.C., which is a field activity of the Bureau of Naval Personnel. Technical assistance was provided by the U.S. Fleet Anti-Air Warfare Training Center, Dam Neck, Virginia Beach, Virginia; U.S. Naval Schools Command Mare Island Vallejo, California; the Bureau of Ships and the U.S. Fleet Anti-Air Warfare Training Center, San Diego, California.

UNITED STATES
GOVERNMENT PRINTING OFFICE
WASHINGTON: 1965

THE UNITED STATES NAVY

GUARDIAN OF OUR COUNTRY

The United States Navy is responsible for maintaining control of the sea and is a ready force on watch at home and overseas, capable of strong action to preserve the peace or of instant offensive action to win in war.

It is upon the maintenance of this control that our country's glorious future depends; the United States Navy exists to make it so.

WE SERVE WITH HONOR

Tradition, valor, and victory are the Navy's heritage from the past. To these may be added dedication, discipline, and vigilance as the watchwords of the present and the future.

At home or on distant stations we serve with pride, confident in the respect of our country, our shipmates, and our families.

Our responsibilities sober us; our adversities strengthen us.

Service to God and Country is our special privilege. We serve with honor.

THE FUTURE OF THE NAVY

The Navy will always employ new weapons, new techniques, and greater power to protect and defend the United States on the sea, under the sea, and in the air.

Now and in the future, control of the sea gives the United States her greatest advantage for the maintenance of peace and for victory in war.

Mobility, surprise, dispersal, and offensive power are the keynotes of the new Navy. The roots of the Navy lie in a strong belief in the future, in continued dedication to our tasks, and in reflection on our heritage from the past.

Never have our opportunities and our responsibilities been greater.

CONTENTS

READING LIST

NAVY TRAINING COURSES

Basic Electricity, NavPers 10086-A
Basic Electronics, NavPers 10087-A
Introduction to Electronics, NavPers 10084
Basic Handtools, NavPers 10085-A
Blueprint Reading and Sketching, NavPers 10077-B
Mathematics, Vol. 1, NavPers 10069-B
Mathematics, Vol. 2, NavPers 10070-B
Mathematics, Vol. 3, NavPers 10073
Standard First Aid, NavPers 10081-A

OTHER PUBLICATIONS

Bureau of Ships Technical Manual, Chapter 9670

USAFI TEXTS

United States Armed Forces Institute (USAFI) courses for additional reading and study are available through your Educational Services' Officer.* The following courses are recommended:

A788	Introduction to Electronics I
A789	Introduction to Electronics II
C166	Advanced Algebra
C176	Plane Geometry I
C177	Plane Geometry II
C188	Trigonometry

*"Members of the United States Armed Forces Reserve components, when on active duty, are eligible to enroll for USAFI courses, services, and materials if the orders calling them to active duty specify a period of 120 days or more, or if they have been on active duty for a period of 120 days or more, regardless of the time specified on the time specified on the active duty orders."

CREDITS

Illustrations indicated below are included in this edition of Data Systems Technician 3&2 through the courtesy of the designated companies. Excerpts from Reference Manual Control Data 1604-A Computer; Control Data 160-A Computer, Programming Manual; and Control Data 1604-A Computer, Principles of Operation—Volume 1 are also included. Permission to use these materials is gratefully acknowledged.

Source (illustrations)	Figure Nos.
International Business Machines	7-8
Control Data Corporation	2-1
	15-1
	15-2

CHAPTER 1

ADVANCEMENT

This training course has been prepared to help you meet the technical qualifications for advancement to Data Systems Technician 3 and Data Systems Technician 2. The DS quals which were used as a guide in the preparation of this training course were current through Change No. 1 of the Manual of Qualifications for Advancement In Rating, NavPers 18068-A.

Chapters 2 through 9 treat basic principles of computers and should be understood before attempting to study any of the other chapters of this course. Chapter 2 presents the history, purpose, and some of the applications of computers. Chapter 3 treats number systems, Boolean Algebra, and methods of converting from one number system to another. Chapters 4 through 7 present the basic circuits and concepts used in the control, arithmetic, memory, and input/output sections of the computer. Chapter 8 treats basic principles of programming by defining the parts of an instruction word, and by showing how these basic word sections are used to express instructions in computer language. Chapter 9 presents the basic concepts and circuits involved in performing analog-to-digital and digital-to-analog conversions.

Chapters 10 through 14 describe the operation of a representative digital computer. The CP-642A/USQ-20(V), a component part of the Naval Tactical Data System, is used as the vehicle in the discussion. Chapter 15 treats the characteristics and word construction of the 160A and 1604A Computers as other representative Navy computers. Chapters 16, 17, and 18 treat test equipments, maintenance information, and maintenance procedures, respectively.

THE ENLISTED RATING STRUCTURE

The present enlisted rating structure, established in 1957, includes three types of ratings—general ratings, service ratings, and emergency ratings.

GENERAL RATINGS identify broad occupational fields of related duties and functions. Some general ratings include service ratings; others do not. Both Regular Navy and Naval Reserve personnel may hold general ratings.

SERVICE RATINGS identify subdivisions or specialties within a general rating. Although service ratings can exist at any petty officer level, they are most common at the PO3 and PO2 levels. Both Regular Navy and Naval Reserve personnel may hold service ratings.

EMERGENCY RATINGS generally identify civilian occupational fields. Emergency ratings do not need to be identified as ratings in the peacetime Navy, but their identification is required in time of war.

THE DATA SYSTEMS RATING

The DS rating is one of the most challenging Navy ratings. Despite this, the number of strikes for this rating is constantly increasing.

Data Systems Technicians maintain electronic digital data systems and equipment. They inspect, test, calibrate, and repair computers, video processors, tape units, buffer equipment, digital display equipment, data link transmitting and receiving equipment, "input-output" devices, and related equipment. They also test and maintain test equipments. Data Systems Technicians are included in the personnel allowance on ships and in shore stations where electronic digital computers are in use.

The DS should have a good background in mathematics. If you are lacking in this area, a study of the mathematics training courses listed in the front of this course is recommended. Skill in the use of tools and test equipment will be acquired through performance of your daily duties. Gaining the necessary technical knowledge, and keeping abreast of the changes in your field, however, will require reading and studying in your spare time.

Also upon advancement to DS 3, you will be graded on your leadership and supervisory ability as well as your ability to perform your technical duties. Study the leadership principles and techniques discussed in Military Requirements for Petty Officers 3&2. Additional material concerning leadership for petty officers is available to you as a result of the current Navy leadership program. As you study the material concerning leadership traits, keep in mind that probably none of our most successful leaders possessed all of these traits to a maximum degree, but a weakness in some traits was more than compensated for by strength in others. Critical self evaluation will enable you to realize the traits which you must strive to improve. Although leadership principles can be taught, a good leader develops these qualities only through hard work and practice. Your success as a leader will be decided for the most part by the success with which you have inspired others to learn and perform through your personal example.

ADVANCEMENT IN RATING

Some of the rewards of advancement in rating are easy to see. You get more pay. Your job assignments become more interesting and more challenging. You are regarded with greater respect by officers and enlisted personnel. You enjoy the satisfaction of getting ahead in your chosen Navy career.

But the advantages of advancing in rating are not yours alone. The Navy also profits. Highly trained personnel are essential to the functioning of the Navy. By each advancement in rating, you increase your value to the Navy in two ways. First, you become more valuable as a technical specialist in your own rating. And second, you become more valuable as a person who can train others and thus make far-reaching contributions to the entire Navy.

HOW TO QUALIFY FOR ADVANCEMENT

What must you do to qualify for advancement in rating? The requirements may change from time to time, but usually you must:

1. Have a certain amount of time in your present grade.
2. Complete the required military and professional training courses.
3. Demonstrate your ability to perform all the PRACTICAL requirements for advancement by completing the Record of Practical Factors, NavPers 760.
4. Be recommended by your commanding officer, after the petty officers and officers supervising your work have indicated that they consider you capable of performing the duties of the next higher rate.
5. Demonstrate your KNOWLEDGE by passing a written examination on (a) military requirements and (b) professional qualifications.

Some of these general requirements may be modified in certain ways. Figure 1-1 gives a more detailed view of the requirements for advancement of active duty personnel; figure 1-2 gives this information for inactive duty personnel.

Remember that the requirements for advancement can change. Check with your division officer or training officer to be sure that you know the most recent requirements.

Advancement in rating is not automatic. After you have all the requirements, you are ELIGIBLE for advancement. You will actually be advanced in rating only if you meet all the requirements (including making a high enough score on the written examination) and if the quotas for your rating permit your advancement.

HOW TO PREPARE FOR ADVANCEMENT

What must you do to prepare for advancement in rating? You must study the qualifications for advancement, work on the practical factors, study the required Navy Training Courses, and study other material that is required for advancement in your rating. To prepare for advancement, you will need to be familiar with (1) the Quals Manual, (2) the Record of Practical Factors, NavPers 760, (3) a NavPers publication called Training Publications for Advancement in Rating, Navpers 10052, and (4) applicable Navy Training Courses. Figure 1-3 illustrates these materials; the following sections describe them and give you some practical suggestions on how to use them in preparing for advancement.

The Quals Manual

The Manual of Qualifications for Advancement in Rating, NavPers 18068-A (change 1), gives the minimum requirements for advancement to each rate within each rating. This manual is usually called the "Quals Manual," and the qualifications themselves are often called

ACTIVE DUTY ADVANCEMENT REQUIREMENTS

REQUIREMENTS *	E1 to E2	E2 to E3	E3 to E4	E4 to E5	E5 to E6	†E6 to E7	† E7 to E8	† E8 to E9
SERVICE	4 mos. service—or completion of recruit training.	6 mos. as E-2.	6 mos. as E-3.	12 mos. as E-4.	24 mos. as E-5.	36 mos. as E-6.	48 mos. as E-7. 8 of 11 years total service must be enlisted. Must be permanent appointment.	24 mos. as E-8. 10 of 13 years total service must be enlisted.
SCHOOL	Recruit Training.	▨	Class A for PR3, DT3, PT3, AME 3, HM 3	▨	▨	Class B for AGCA, MUCA, MNCA.		
PRACTICAL FACTORS	Locally prepared check-offs.	Records of Practical Factors, NavPers 760, must be completed for E-3 and all PO advancements.						
PERFORMANCE TEST	▨	▨	Specified ratings must complete applicable performance tests before taking examinations.				▨	▨
ENLISTED PERFORMANCE EVALUATION	As used by CO when approving advancement.		Counts toward performance factor credit in advancement multiple.					
EXAMINATIONS	Locally prepared tests.		Navy-wide examinations required for all PO advancements.				Navy-wide, selection board, and physical.	
NAVY TRAINING COURSE (INCLUDING MILITARY REQUIREMENTS)	▨	Required for E-3 and all PO advancements unless waived because of school completion, but need not be repeated if identical course has already been completed. See NavPers 10052 (current edition).					Correspondence courses and recommended reading. See NavPers 10052 (current edition).	
AUTHORIZATION	Commanding Officer		U.S. Naval Examining Center			Bureau of Naval Personnel		
	TARS attached to the air program are advanced to fill vacancies and must be approved by CNARESTRA.							

* All advancements require commanding officer's recommendation.
† 2 years obligated service required.

Figure 1-1.—Active duty advancement requirements.

INACTIVE DUTY ADVANCEMENT REQUIREMENTS

REQUIREMENTS *		E1 to E2	E2 to E3	E3 to E4	E4 to E5	E5 to E6	E6 to E7	E8	E9
	FOR THESE DRILLS PER YEAR								
TOTAL TIME IN GRADE	48 24 NON-DRILLING	6 mos. 9 mos. 12 mos.	6 mos. 9 mos. 24 mos.	15 mos. 15 mos. 24 mos.	18 mos. 18 mos. 36 mos.	24 mos. 24 mos. 48 mos.	36 mos. 36 mos. 48 mos.	48 mos. 48 mos.	24 mos. 24 mos.
DRILLS ATTENDED IN GRADE †	48 24	18 16	18 16	45 27	54 32	72 42	108 64	144 85	72 32
TOTAL TRAINING DUTY IN GRADE †	48 24 NON-DRILLING	14 days 14 days None	14 days 14 days None	14 days 14 days 14 days	14 days 14 days 14 days	28 days 28 days 28 days	42 days 42 days 28 days	56 days 56 days	28 days 28 days
PERFORMANCE TESTS		▨	▨	Specified ratings must complete applicable performance tests before taking examination.					
PRACTICAL FACTORS (INCLUDING MILITARY REQUIREMENTS)		Record of Practical Factors, NavPers 760, must be completed for all advancements.							
NAVY TRAINING COURSE (INCLUDING MILITARY REQUIREMENTS)		Completion of applicable course or courses must be entered in service record.							
EXAMINATION		Standard exams are used where available, otherwise locally prepared exams are used.						Standard EXAM, Selection Board, and Physical.	
AUTHORIZATION		District commandant or CNARESTRA					Bureau of Naval Personnel		

* Recommendation by commanding officer required for all advancements.
† Active duty periods may be substituted for drills and training duty.

Figure 1-2.—Inactive duty advancement requirements.

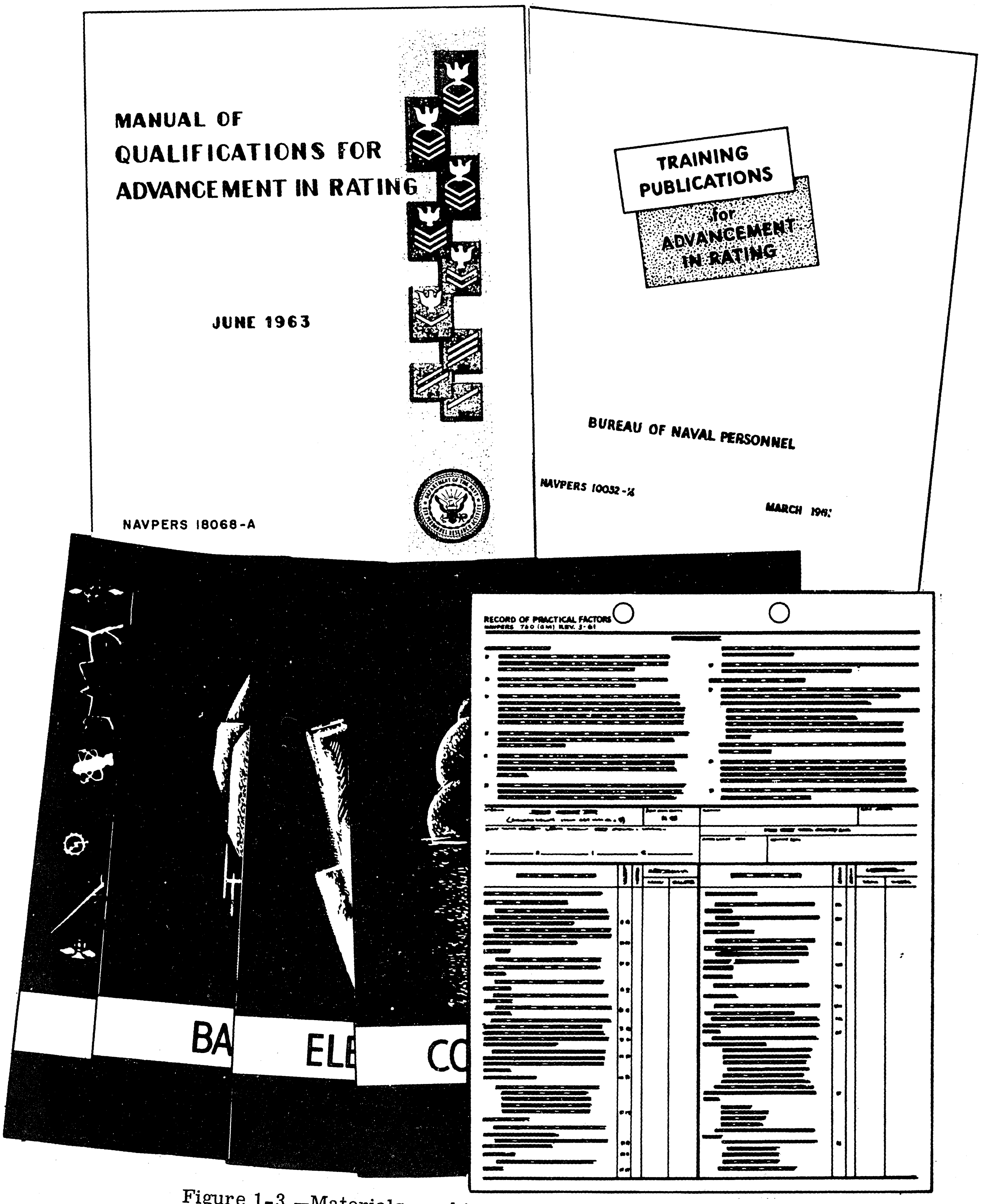


Figure 1-3.—Materials used in preparing for advancement.

"quals." The qualifications are of two general types: (1) military requirements, and (2) professional or technical qualifications.

MILITARY REQUIREMENTS apply to all ratings rather than to any one particular rating. Military requirements for advancement to third class and second class petty officer rates deal with military conduct, naval organization, military justice, security, watch standing, and other subjects which are required of petty officers in all ratings.

PROFESSIONAL QUALIFICATIONS are technical or professional requirements that are directly related to the work of each rating.

Both the military requirements and the professional qualifications are divided into subject matter groups; then, within each subject matter group, they are divided into PRACTICAL FACTORS and KNOWLEDGE FACTORS. Practical factors are things you must be able to DO. Knowledge factors are things you must KNOW in order to perform the duties of your rating.

The written examination you will take for advancement in rating will contain questions relating to the practical factors and the knowledge factors of both the military requirements and the professional qualifications. If you are working for advancement to second class remember that you may be examined on third class qualifications as well as on second class qualifications.

The Quals Manual is kept current by means of changes. The professional qualifications for your rating which are covered in this training course were current at the time the course was printed. By the time you are studying this course, however, the quals for your rating may have been changed. Never trust any set of quals until you have checked it against an UP-TO-DATE copy in the Quals Manual.

Record of Practical Factors

Before you can take the servicewide examination for advancement in rating, there must be an entry in your service record to show that you have qualified in the practical factors of both the military requirements and the professional qualifications. A special form known as the RECORD OF PRACTICAL FACTORS, NavPers 760, is used to keep a record of your practical factor qualifications. This form is available for each rating. The form lists all practical factors, both military and professional. As you demonstrate your ability to perform each practical factor, appropriate entries are made in the DATE and INITIALS columns.

Changes are made periodically to the Manual of Qualifications for Advancement in Rating, and revised forms of NavPers 760 are provided when necessary. Extra space is allowed on the Record of Practical Factors for entering additional practical factors as they are published in changes to the Quals Manual. The Record of Practical Factors also provides space for recording demonstrated proficiency in skills which are within the general scope of the rating but which are not identified as minimum qualifications for advancement.

If you are transferred before you qualify in all practical factors, the NavPers 760 form should be forwarded with your service record to your next duty station. You can save yourself a lot of trouble by making sure that this form is actually inserted in your service record before you are transferred. If the form is not in your service record, you may be required to start all over again and requalify in the practical factors which have already been checked off.

NavPers 10052

Training Publications for Advancement in Rating, NavPers 10052 (revised), is a very important publication for anyone preparing for advancement in rating. This bibliography lists required and recommended Navy Training Courses and other reference material to be used by personnel working for advancement in rating. NavPers 10052 is revised and issued once each year by the Bureau of Naval Personnel. Each revised edition is identified by a letter following the NavPers number. When using this publication, be SURE that you have the most recent edition.

If extensive changes in qualifications occur in any rating between the annual revisions of NavPers 10052, a supplementary list of study material may be issued in the form of a BuPers Notice. When you are preparing for advancement, check to see whether changes have been made in the qualifications for your rating. If changes have been made, see if a BuPers Notice has been issued to supplement NavPers 10052 for your rating.

The required and recommended references are listed by rate level in NavPers 10052. If you are working for advancement to third class, study the material that is listed for third class. If you are working for advancement to second class, study the material that is listed for second class; but remember that you are also responsible for the references listed at the third class level.

In using NavPers 10052, you will notice that some Navy Training Courses are marked with an asterisk (*). Any course marked in this way is MANDATORY — that is, it must be completed at the indicated rate level before you can be eligible to take the servicewide examination for advancement in rating. Each mandatory course may be completed by (1) passing the appropriate enlisted correspondence course that is based on the mandatory training course; (2) passing locally prepared tests based on the information given in the training course; or (3) in some cases, successfully completing an appropriate Class A school.

Do not overlook the section of NavPers 10052 which lists the required and recommended references relating to the military requirements for advancement. Personnel of ALL ratings must complete the mandatory military requirements training course for the appropriate rate level before they can be eligible to advance in rating.

The references in NavPers 10052 which are recommended but not mandatory should also be studied carefully. ALL references listed in NavPers 10052 may be used as source material for the written examinations, at the appropriate rate levels.

Navy Training Courses

There are two general types of Navy Training Courses. RATING COURSES (such as this one) are prepared for most enlisted ratings. A rating training course gives information that is directly related to the professional qualifications of ONE rating. SUBJECT MATTER COURSES or BASIC COURSES give information that applies to more than one rating.

Navy Training Courses are revised from time to time to keep them up to date technically. The revision of a Navy Training Course is identified by a letter following the NavPers number. You can tell whether any particular copy of a Navy Training Course is the latest edition by checking the NavPers number and the letter following this number in the most recent edition of List of Training Manuals and Correspondence Courses, NavPers 10061. (NavPers 10061 is actually a catalog that lists all current training courses and correspondence courses; you will find this catalog useful in planning your study program.)

Navy Training Courses are designed to help you prepare for advancement in rating. The following suggestions may help you to make the best use of this course and other Navy training publications when you are preparing for advancement in rating.

1. Study the military requirements and the professional qualifications for your rating before you study the training course, and refer to the quals frequently as you study. Remember, you are studying the training course primarily in order to meet these quals.

2. Set up a regular study plan. It will probably be easier for you to stick to a schedule if you can plan to study at the same time each day. If possible, schedule your studying for a time of day when you will not have too many interruptions or distractions.

3. Before you begin to study any part of the training course intensively, become familiar with the entire book. Read the preface and the table of contents. Check through the index. Look at the appendixes. Thumb through the book without any particular plan, looking at the illustrations and reading bits here and there as you see things that interest you.

4. Look at the training course in more detail, to see how it is organized. Look at the table of contents again. Then, chapter by chapter, read the introduction, the headings, and the subheadings. This will give you a pretty clear picture of the scope and content of the book. As you look through the book in this way, ask yourself some questions: What do I need to learn about this? What do I already know about this? How is this information related to information given in other chapters? How is this information related to the qualifications for advancement in rating?

5. When you have a general idea of what is in the training course and how it is organized, fill in the details by intensive study. In each study period, try to cover a complete unit — it may be a chapter, a section of a chapter, or a subsection. The amount of material that you can cover at one time will vary. If you know the subject well, or if the material is easy, you can cover quite a lot at one time. Difficult or unfamiliar material will require more study time.

6. In studying any one unit — chapter, section, or subsection — write down the questions that occur to you. Many people find it helpful to make a written outline of the unit as they study, or at least to write down the most important ideas.

7. As you study, relate the information in the training course to the knowledge you already have. When you read about a process, a skill, or a situation, try to see how this information ties in with your own past experience.

8. When you have finished studying a unit, take time out to see what you have learned. Look back over your notes and questions. Maybe some of your questions have been answered, but perhaps you still have some that are not answered. Without looking at the training course, write down the main ideas that you have gotten from studying this unit. Don't just quote the book. If you can't give these ideas in your own words, the chances are that you have not really mastered the information.

9. Use Enlisted Correspondence Courses whenever you can. The correspondence courses are based on Navy Training Courses or on other appropriate tests. As mentioned before, completion of a mandatory Navy Training Course can be accomplished by passing an Enlisted Correspondence Course based on the Navy Training Course. You will probably find it helpful to take other correspondence courses, as well as those based on mandatory training courses. Taking a correspondence course helps you to master the information given in the training course, and also helps you see how much you have learned.

10. Think of your future as you study Navy Training Courses. You are working for advancement to third class or second class right now, but someday you will be working toward higher rates. Anything extra that you can learn now will help you both now and later.

SOURCES OF INFORMATION

One of the most useful things you can learn about a subject is how to find out more about it. No single publication can give you all the information you need to perform the duties of your rating. You should learn where to look for accurate, authoritative, up-to-date information on all subjects related to the military requirements for advancement and the professional qualifications of your rating.

Some publications are subject to change or revision from time to time, some at regular intervals, others as the need arises. When using any publication that is subject to change or revision, be sure that you have the latest edition. When using any publication that is kept current by means of changes, be sure you have a copy in which all official changes have been made. Studying canceled or obsolete information will not help you to do your work or to advance in rating; it is likely to be a waste of time, and may even be seriously misleading.

Technical publications that will be helpful as references and in preparing for advancement are discussed in chapter 17. Other training courses that will be helpful to you are listed in the front of this course.

Training films serve as a valuable source of supplementary information. A selected list of training films appears in appendix I of this training course. Other films that may be helpful are listed in the U.S. Navy Film Catalog, NavPers 10000 (revised).

CHAPTER 2

INTRODUCTION TO COMPUTERS

The development of high speed data processing and computing devices has enabled the Navy to utilize new and highly sophisticated tactical and logistic systems. In these and other ways, the use of computers has greatly improved the efficiency of many naval operations. As a prospective Data Systems Technician, the operation and maintenance of computers will be your responsibility. Keep in mind that the measure of our Navy's success with computing devices depends upon the ability of its technicians to understand and apply the necessary fundamental operating and maintenance concepts.

DEFINITION

A computer is a device which is capable of receiving information in a given form at its input, performing certain prescribed internal operations on this information, and producing a result at its output. A representative electronic digital computer presently in use by some naval activities is the Control Data 1604-A computer (fig. 2-1). Some of the features of this computer are treated in chapter 15 of this text.

Not all computers are so complex. An example of a basic computer is the commonly used gear reduction train. Here, the input information, or data, is altered by the gears to produce modified data at the gear train output. A second example is the adding machine discussed in Basic Electronics, NavPers 10087 (revised).

HISTORY

In order to appreciate the high speed and accuracy of computers today, it is desirable to have some knowledge of earlier computers and their originators. The first calculator was built by Pascal in 1642. This machine could add, subtract, and was successfully applied to tax collection in France. In 1671, Leibniz built a machine which could add, subtract, multiply, and divide. Leibniz also recognized the advantages of the binary (or two digit) number system over the decimal (or ten digit) number system which will be pointed out in chapter 3 of this course.

In 1842, Babbage started a machine called a ''Difference Engine.'' This machine would have been able to construct tables of any function that could be described by the first five differences. Work on this machine ceased, however, when Babbage conceived the idea of an ''Analytic Engine'' which could tabulate any function. In 1910, his son completed building the latter machine and used it to calculate π to 20 decimal places. Babbage also suggested the use of punched cards (now used as input and output media for computers) and the possibility of a machine that could alter its operations according to the results of its calculations. Present electronic computers resemble his analytic engine in many respects.

Between 1937 and 1944, Aiken built the first general purpose automatic digital computer, called the ''Mark I.'' It is electromechanical, using relays as one of the major calculating devices.

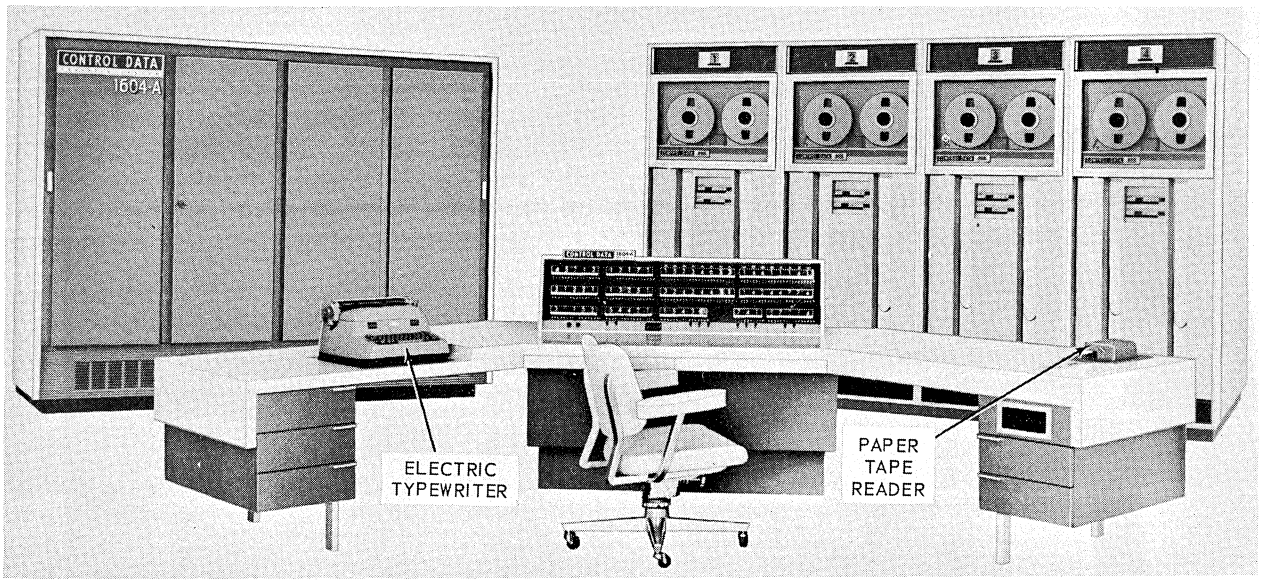
Between 1939 and 1945, Eckert and Mauchly built the ''Eniac,'' also a digital computer. It consisted of 18,000 electron tubes, weighed 30 tons, and dissipated 150 kilo watts. The time required for an add operation was only 0.21 μsec compared to 300 μsec for the Mark I.

In 1951 Eckert and Mauchly built the first ''Univac'' for the United States Census Bureau.

The ''Edvac,'' completed in 1952, was the first computer to use internally stored instructions (program).

CAPABILITIES

The two major characteristics of computers which make them so useful in military and commercial applications are SPEED and ACCURACY. The speed of computers is seen when we consider that problems which require days,



124.1X

Figure 2-1.—Representative digital computer.

weeks, or years to solve by man, with his slow pencil and paper tools, can be solved in seconds or minutes by a computer. This is conceivable when we consider that a single arithmetic operation can be solved and stored by a computer in a few microseconds whereas a mathematician needs a few seconds to do the same operation and record (or store) it on paper. Thus, the computer can solve the problems and produce an output record of its results, thousands or even millions of times faster than man.

The second characteristic of the computer is ACCURACY. Once a computer is provided with the correct instructions, the planned operations can be repeated millions of times without a single error. Computers make errors only when there is a breakdown in the computing system, or when there is human error in the prepared instructions. Once the breakdown or error is detected and corrected, the computer again operates at high speeds and without error.

Computers are used in many fields of research. In engineering, they are used in design. In business, they are used in bookkeeping and inventory; in government for the census; and by the military, in logistics and battle strategy problems. Any problem which can be reduced-to and solved-by a sequence of arithmetic steps can be done rapidly by a computer.

DIGITAL COMPUTERS

Computers are classified into two general types; digital and analog, although a variation of these types called a "hybrid computer" has both digital and analog characteristics. The hybrid computer is not treated in this text.

The digital computer, as implied in the name, produces its output by responding to changes in fixed increments, such as 0 to 1, or 1 to 2, etc. The digital change may be accomplished in a gear train, by changing a voltage from one level to another; by the "on" or "off" condition of a switch; by the energized or deenergized condition of a relay; or by the presence or absence of electrical pulses.

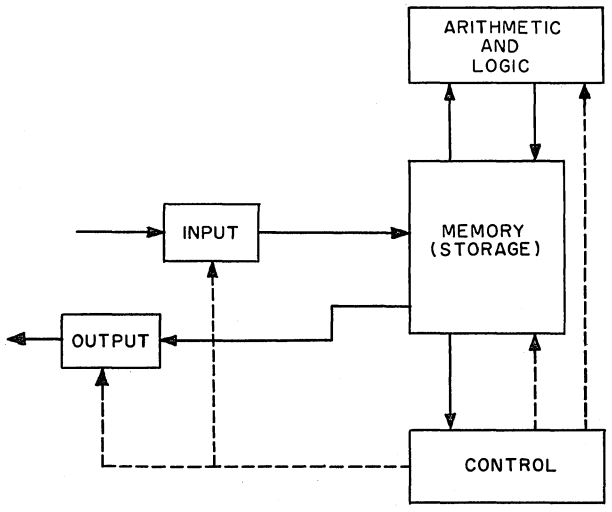
Examples of digital devices are adding machines, cash registers, the abacus, thought processes in human calculations, and the odometer used in conjunction with an automobile speedometer.

Electronic digital computers can be made accurate to any desirable degree. They are usually more expensive than analog computers but are also usually more versatile. A digital computer can be given a sequence of instructions in which it can execute later steps using the results of the earlier steps. It can also alter the sequence of instructions according to the results of previous steps.

ANALOG COMPUTERS

Analog computers use physical changes as input data and indicate the significance which such changes have on the device or unit as a whole. From this, it should be noted that analog computers are limited in their application to problems related to specific devices. They lend themselves ably to such problems as guided missile control or radar antenna training.

The measure of the output accuracy of the analog computer is "analogous" to the accuracy of the input data. In this respect, the analog computer can be compared to the slide rule. When using a short slide rule, it is difficult to feed information into the rule with accuracy. Likewise, it is difficult to read the rule accurately. The longer the rule (within limits), the easier it is to feed-in and read-out more accurate data. Obviously, there is a practical limit to the rule length, which thus imposes a limit on the accuracy of the input and output data. Because this principle is true of all analog computers, these computers are normally less accurate than digital computers which, as stated earlier, use fixed incremental inputs to yield fixed incremental outputs with nearly 100 percent accuracy. The treatment in this text is generally limited to DIGITAL COMPUTERS.



124.2
Figure 2-2.—Basic computer block diagram.

BASIC COMPUTER BLOCK DIAGRAM

The basic sections of an elementary digital computer are shown in figure 2-2.

The control section is comparable to a telephone exchange. It directs the operations of the computer under the direct influence of a sequence of instructions called the "program." These instructions are comparable to the phone numbers dialed into a phone exchange and cause certain switches and control lines to be energized.

The program may be stored in the internal circuits of the computer or it may be read instruction-by-instruction from external media. The internally stored program type of computer, generally referred to only as a "stored program" computer, is the most practical type to use when speed and fully automatic operation are desired.

In addition to the command which tells the computer what to do, the control unit also dictates how and when each specific operation is to be performed. It is also active in initiating circuits which locate any information stored in the computer and in moving this information to the point where the actual manipulation or modification is to be accomplished.

In the stored program computer, the control unit reads an instruction from the memory section (as instructed by the program). The information read into the control unit from memory is in the form of voltage levels that make up a "binary word" and represents a specific operation that is to be performed. The location of the data to be operated on is generally a part of the instruction and energizes circuitry which causes the specified operation (such as add, subtract, compare, etc.) to be executed. Subsequently, the control unit reads the next instruction, or jumps as directed, (explained later) to find the next instruction to execute.

The arithmetic unit of the computer is the section in which arithmetic and logic operations are performed on the input or stored data. The arithmetic operations performed in this unit include adding, subtracting, multiplying, dividing, counting, shifting, complementing, and comparing.

All arithmetic operations can be reduced to any one of four arithmetic processes; addition, subtraction, multiplication, or division. In most computers, multiplication involves a series of additions; and division, a series of subtractions.

The arithmetic unit contains several registers; units which can store one "word" of computer data. This group of registers generally include X and Q registers (so named for identification purposes only), and a unit called an "accumulator" (A register). During an arithmetic process, the X and Q registers temporarily hold or store the numbers being used in the operation, called "operands". The accumulator stores the result of the operation. The control unit instructs the arithmetic unit to perform the specified arithmetic operation (as requested in the instruction); transfers the necessary information into the X and Q registers from memory (discussed later); and controls the storage of the results in the accumulator or in some specific location in memory.

The arithmetic unit also makes comparisons and produces "yes" or "no" or "go-no-go" outputs as a result. The computer may be programmed so that a "yes" or "go" result advances the computer units to perform the next step in the program, whereas a "no" or "no-go" instruction may cause the computer to jump several programmed steps. A computer may be programmed so that a "no" result at a certain point in the program will cause the computer to stop and await instructions from a keyboard or other input device.

Generally information delivered to the control unit represents instructions, whereas information routed to the arithmetic unit represents data. Frequently it is necessary to modify an instruction. This instruction may have been used in one form in one step of the program but must be altered for a subsequent step. In such cases, the instruction is delivered to the arithmetic unit where it is altered by addition-to or subtraction-from another number in the accumulator. The resultant modified instruction is again stored in the memory unit for use later in the program.

In most digital computers the storage or memory section is constructed of small magnetic cores, each capable of representing an "ON" ("1") or "OFF" ("0") condition. A system of these cores arranged in a matrix can store any computer word which is represented in binary form.

All computers must contain facilities to store computer words or instructions (which are intelligible to the computer) until these instructions or words are needed in the performance of the computer calculations. Before the stored program type computer can begin to operate on its input data, it is first necessary to store, in memory, a sequence of instructions and all figures, numbers, and any other data which are to be used in the calculations. The process by which these instructions and data are read into the computer is called "loading."

Actually the first step in loading instructions and data into a computer is to manually place enough instructions into memory by using the console or keyboard so that these instructions can be used to bring in more instructions as desired. In this manner a few instructions are used to "bootstrap" more instructions. Some computers make use of an auxiliary (wired) memory which permanently stores the "bootstrap program", thereby making manual loading unnecessary. This process is explained in more detail in subsequent chapters.

The memory (or storage) section of a computer is essentially an electronically operated file cabinet. It is actually a large number (generally between 1 and 40 thousand) of storage locations; each referred to as a storage address or register. Every computer word which is read into the computer during the loading process is stored or filed in a specific storage address and is almost instantly accessible.

Input and output devices are similar in operation but perform opposite functions. It is through the use of these devices that the computer is able to communicate with the outside world.

Input data may be in any one of three forms: It may be fed in manually from a keyboard or console; from instruments or sensors; or from a source on which data has previously been stored in a form intelligible to the computer.

Computers can process hundreds of thousands of computer words per second. Thus, a study of the first method (manual input) reflects the incompatibility of human-operated keyboards or keypunches to supply data at a speed which matches the electronic speed of digital computers. A high average speed for keyboard operation is 2 or 3 characters per second, which when coded to form computer words may have more than 15 to 20 binary digits. The computer is capable of reading several thousand times this amount of information per second. It is clear, therefore, that manual inputs should be minimized to make more efficient use of computer time.

Instruments are used as input sensors, and are capable of supplying several thousand samples regarding pressure, temperature, speed, etc., per second. This is equivalent to 10 or 20 thousand bits or binary digits per second. Digital computers which use these devices must be equipped with analog-digital converters to convert physical change to specific increments.

Finally, input information may be supplied from cards, paper tapes, or magnetic tapes; which have been stored previously outside of the computer but in a form understood by the computer program. This is the fastest method of supplying input data to the computer.

Some commonly used input devices are paper tape readers, teletypewriters, and magnetic type units.

Output information is also made available in three types: human information, such as codes or symbols presented on a cathode-ray screen which are used by the operator to answer questions or make decisions; information which operates a control device such as a lever, aileron, or actuator; or information which is stored in a machine language or human language, on tapes, or printed media.

Devices which store or read-out output information include magnetic tape, punched cards, punched paper tapes, cathode-ray oscilloscopes, electric typewriters, line-at-a-time printers, and surface-at-a-time printers.

One of the main features of computers is their ability to process large amounts of data quickly. In most cases, the processing speed far exceeds the ability of input devices to supply information. One common limitation of most input devices is that each involves some mechanical operation, that is, the movement of a tape drive or card feeder. Because a mechanical movement of some part of these devices cannot take place fast enough to match electronic speeds within the computer, these input devices limit the speed of operation of the associated computer particularly in cases where successive operations are dependent upon the reception of new data from the input medium.

Several methods of speeding up mechanical operations have been devised, all of which are designed to move a smaller mass a shorter distance and with greater driving force. Many of these designs have been directed toward increasing the drive speed of magnetic tapes. Present day tape drives can pass up to 112.5 inches of tape per second over a tape reading head. Card readers (discussed in a later chapter) can read between 100 and 1000 cards per minute, depending on the particular reader.

With an understanding of the function of the various computer sections, let us now consider a basic computer instruction and how this instruction is executed. Let the instruction be as follows:

> ''Add the contents of the A register to the contents of memory address location 123 and store the results in address 456 in memory.''

We will assume that the computer used is the stored program type and that all instructions, data, numbers, and symbols have been previously loaded or stored in memory at known addresses. The stored input may have been read from a magnetic tape (similar to that used with commercial tape recorders), from paper tape (similar to that used with teletype), or from punched cards.

If the instruction to be executed is the first programmed operation, energizing the start button will cause the control unit to issue an order ''Read instruction.'' The instruction will be read into a register in the control unit where it will remain throughout the execution cycle.

Note that the mathematical operation requested in the instruction is ADD. The instruction word thus contains a code which is interpreted by the control unit—ADD.

After reading the instruction, the control unit will automatically energize circuits which will (1) read-out the contents of memory address 123 and (2) transfer this information to a register (say the X register) in the arithmetic unit. The contents of register A (accumulator) is read into say the Q register in the arithmetic unit.

The ADD process is then accomplished, being constantly monitored by the control unit to ensure that no further actions are initiated before the ADD operation is completed. The results of the ADD operation are stored in the accumulator from which, by control request, it is transferred to address 456 in memory. This ends the instruction. The control unit will read and execute the next instruction.

If the result is to be displayed at the output immediately or at a later time (as stipulated in the programmed instructions) the control unit upon receipt of the instruction will issue an order to read-out the contents of memory address 456. Because read-out (which sometimes involve printing by some electromechanical apparatus) is extremely slow as compared to computer speed, most computers use a secondary storage device called a buffer onto which data is read directly from the primary (main) storage at computer speeds. When read-out is desired, the control unit enables the buffer storage to read-out all or any part of the buffer storage. The buffer read-out is therefore independent of the main computer operation, and in some computers only one instruction is required to start and stop the read-out process.

COMPONENTS USED IN COMPUTERS

Unlike the mechanical computers, such as adding machines and odometers which are based on the decimal (ten) digit system, modern electronic computers use components which will represent only 2 conditions. These conditions are sometimes referred to as the 1 (energized) or 0 (deenergized) states. Early computers used relays and electron tubes; now transistors and silicon or germanium diodes are used because of the higher speeds at which they can react, and too, because of their lower power consumption. The physics and operation of these devices is explained in Basic Electronics, NavPers 10087 (revised) training course.

Electronic circuits used in computers are basically simple. To a large extent these circuits are of four types: the OR circuit which produces an output when one or more of its inputs is high, that is, in the one state; the AND circuit which yields an output only when all inputs are high; the flip-flop circuit which is a bistable multivibrator; and the inverter circuit which yields a high output with a low input or a low output with a high input.

The reader should familiarize himself with basic electronic circuits by studying Basic Electronics, NavPers 10087 (revised). Only those circuits which are peculiar to computers will be treated in this text.

COMPUTER WAVEFORMS

To a great extent the speed and accuracy of digital computers are directly related to the type of pulse signals used. Most computer circuits use square or rectangular-shaped waveforms to trigger circuits designed to have 2 stable states. Pulses may be negative going, positive going, or alternating, but must have a sharp rise time to ensure positive triggering action.

In modern computers a series of pulses referred to as clock pulses is produced by a generator system which has a fixed rate. The term "clock pulses" is indicative of the action of these pulses to time all operations within the computer.

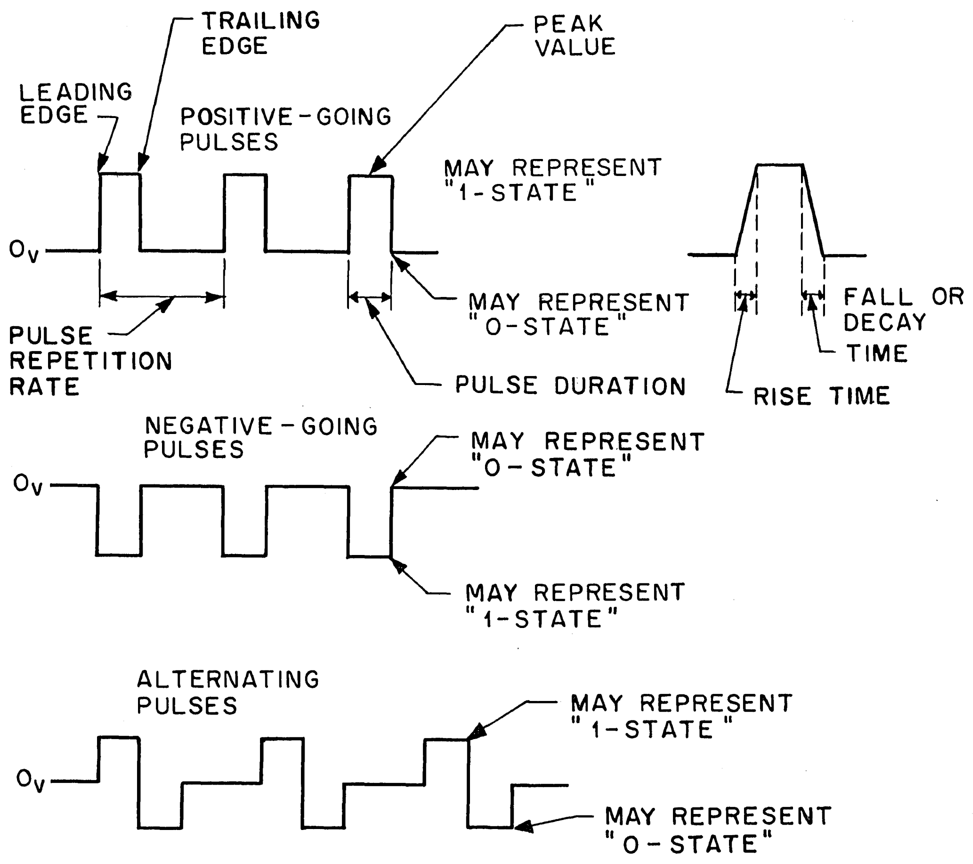
In many cases the source of the clock pulses is a synchronized multivibrator. The waveforms shown in figure 2-3 are representative of those used as triggering or gating pulses for computer circuits.

Each action within a computer must begin and end at a specified time. The clock pulses compensate for slight inaccuracies (either advances or delays) in the timing of binary pulses which initiate various actions throughout the computer. Take for example the AND circuit (fig. 2-4) which requires 2 input pulses (positive going) in coincidence and produces an output to initiate the execution of an operation. If the enable input is either advanced or delayed in time as it arrives at one of the AND circuit inputs, the clock pulse will coincide with the binary input only for a short period, and the condition of this coincidence will ensure that the output will appear at the proper time (only at the time of the clock pulse).

ENVIRONMENTAL CONDITIONS

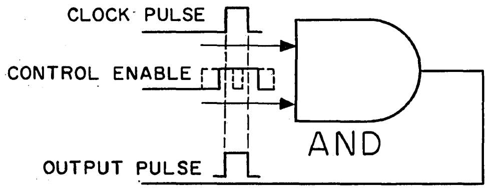
Each of the myriad contacts and components in a computer is a potential source of breakdown. Thousands of resistors, capacitors, crystal diodes, and transistors, are used in computers, each of which exhibits some change in its electrical value or conducting resistance when subjected to temperature or humidity changes. Some computer components such as photoelectric card readers are sensitive to light and can introduce errors in the computing system if exposed to room lighting. Gas-filled voltage regulating tubes or components which operate in pressurized (gas) containers are affected by changes in gas pressure.

Any significant change in the value of computer components can cause errors in the computations. If the error is introduced early in the computation, the output may be grossly inaccurate. However, a malfunction at any point in the calculation, whether caused by a transient voltage, intermittent component, environmental change, or a mistake in the program will produce incorrect results at the output.



124.3

Figure 2-3.—Computer waveforms.



124.4

Figure 2-4.—Basic example of the use of clock pulses.

As a DATA SYSTEMS Technician one of your responsibilities will be to control the environmental conditions surrounding the computing equipments within specified limits. The temperature and humidity must be controlled in order to prevent expansion of mechanical linkages, changes in resistance values and rust and corrosion of components which are not adequately protected.

Other duties involving maintenance responsibilities are treated in chapters 18 and 19 of this course.

CHAPTER 3

NUMBER SYSTEMS

How does a digital computer represent data internally in order to perform operations on it? The answer is found in the nature of the principal electronic components employed. These include diode rectifiers, transistor flip-flops, and magnetic cores, all of which have two possible states (bi-stable). They are either ON or OFF, conducting or nonconducting, energized or deenergized. The presence of electrical signals in certain components and the absence of signals in others represent the internal data.

BINARY ARITHMETIC

Computers function in the binary number system using digits 0 and 1. The components that represent data can only represent two possible stable states, as previously stated, that of conducting or nonconducting. For example, we may represent decimal numbers (data) by using light bulbs which are assigned the decimal values of 8, 4, 2, and 1 reading from left-to-right in figure 3-1. A lighted lamp in the 8's column represents a 1 condition in that column. The numerical value of this 1 condition is 8. A 1 condition at the 4 level represents 4, etc. The decimal value 0 is represented when all lights are off. The decimal value of 15 is represented when all lights are "on". The decimal value of 5 is represented by having the 4 light on and the 1 light on. The decimal value of 12 is represented by having the 8 light on and the 4 light on. Thus any decimal number from 0 to 15 may be represented by the four light bulbs.

The topic of number systems is discussed in detail in the training course Mathematics, Vol. 3. In this chapter we will only review the main points of number systems as discussed in that course.

BASIC OPERATIONS

Computation in the binary number system is relatively simple since the number system uses only the digits 1 and 0.

Addition

For addition three rules apply:
1. Zero plus zero equals zero.
2. Zero plus one equals one.
3. One plus one equals zero with a carry of one to the next position on the left.

Example:

Binary Representation	Decimal Equivalent
01111	15_{10}
+00111	$+\ 7_{10}$
10110	22_{10}

Subtraction

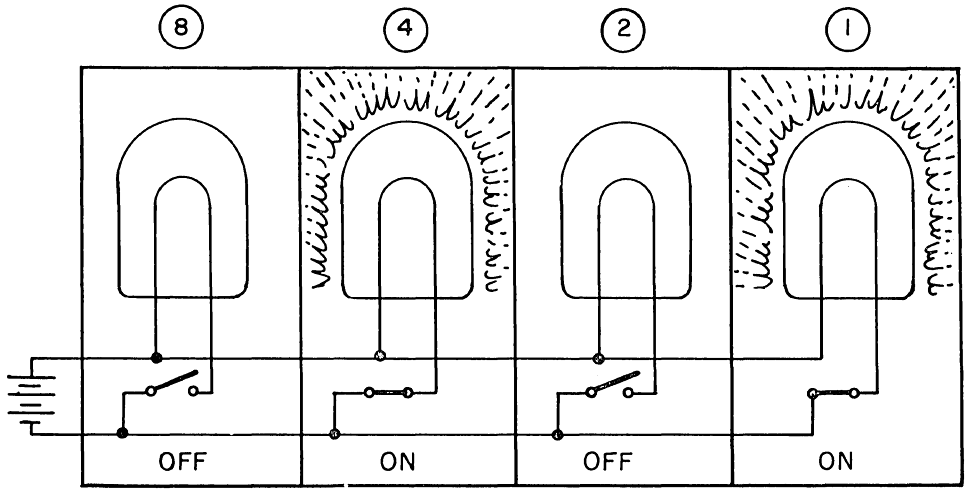
For subtraction four rules apply:
1. Zero minus zero equals zero.
2. One minus one equals zero.
3. One minus zero equals one.
4. Zero minus one equals one, with one borrowed from the left.

Example:

	Binary Representation	Decimal Equivalent
Borrows	$\left\{\begin{matrix}11\\11\end{matrix}\right\}$	
	$0\cancel{1}\cancel{1}01$	13_{10}
	-00110	$-\ 6_{10}$
	00111	7_{10}

Multiplication

For multiplication three rules apply:
1. Zero times zero equals zero.



124.5

Figure 3-1.—Light bulbs assigned decimal numbers.

2. Zero times one equals zero; no carries are considered.

3. One times one equals one.

Example:

Binary Representation	Decimal Equivalent
01111	15_{10}
x 100	$\times 4_{10}$
00000	60_{10}
00000	
01111	
0111100	

Division

For division similar concepts are applied:

Example:

Binary Representation	Decimal Equivalent
101	5_{10}
100/10100	$4_{10}/20_{10}$
100	20
100	0
100	

CONVERTING BETWEEN NUMBER SYSTEMS

Although the computer operates internally using the binary system, the programmer or operator may express input-output data in the decimal form; and by means of a stored program the computer can convert from one system to another.

It can be seen that binary numbers require about three times as many positions as decimal numbers to express the equivalent number. Although this is not much of a problem to the computer itself, in talking and writing, these binary numbers are quite bulky. A long string of zeros and ones is difficult to interpret. A more convenient method is desirable and the octal number system fills this need. Numbers can be converted from the octal to binary and from binary to octal by inspection. There is a simple relationship between the octal and binary system. Three binary positions are equivalent to one octal position. The following table is used constantly when working on or about the computer.

BINARY	OCTAL
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

At this point a carry to the next higher position of the number is required, since all eight symbols have been used.

BINARY	OCTAL
1000	10
1001	11
1010	12
1011	13
	and so forth

DIRECT CONVERSION OF RADIX r_1 TO r_2

In order to convert an octal number to a binary number or a binary number to an octal number, use the following rule:

Rule: Express the number in binary groups of three and arrange the groups according to the following example:

OCTAL TO BINARY			BINARY TO OCTAL		
2	2	5	010	010	101
010	010	101	2	2	5

In the decimal system a number is represented or expressed by a sum of terms. An individual term consists of a product of a power of ten and some integer from 0 to 9. For example, the number 125 means 100 plus 20 plus 5. This can also be expressed as:

$$(1\text{x}10^2) + (2\text{x}10^1) + (5\text{x}10^0)$$
$$100 + 20 + 5 = 125_{10}$$

In a like manner the binary number 1111101 can be expressed as follows:

$$(1\text{x}2^6) + (1\text{x}2^5) + (1\text{x}2^4) + (1\text{x}2^3) + (1\text{x}2^2) + (0\text{x}2^1) + (1\text{x}2^0) =$$
$$64 + 32 + 16 + 8 + 4 + 0 + 1 = 125_{10}$$

The octal number 175 can be expressed as follows:

$$(1\text{x}8^2) + (7\text{x}8^1) + (5\text{x}8^0)$$
$$64 + 56 + 5 = 125_{10}$$

To convert a DECIMAL integral number to an OCTAL integral number, divide the decimal number by 8, and the remainder equals the unit's position of the octal number. Divide the quotient (obtained from the first division) by 8 and this remainder equals the next (eight's) position of the octal number. Repeat the above process until the quotient zero is reached.

Example: Convert 149_{10} to an octal number.

```
   18           2              0
8/149        8/18           8/2
  8            16             0
  69            2 remainder   2 remainder
  64
   5 remainder
```

The last remainder is the most significant digit in the octal number.

$$149_{10} = 225_8$$

To convert a DECIMAL fractional number to an OCTAL fractional number, multiply by 8 and develop the octal number as follows:

Example: Convert 0.149_{10} to an octal number.

```
1       0.149
        x   8
        1.192

1       0.192
        x   8
        1.536

4       0.536
        x   8
        4.288

2       0.288
        x   8
        2.304
```

$$0.1142_8 = 0.149_{10}$$

To convert an OCTAL integral number to a DECIMAL number multiply the highest-order digit of the octal number by 8 and add the next lower order digit of the octal number to the product. Multiply this sum by 8 and add the next lower order digit to the result. When the lowest order digit has been added to the answer, the process ends.

Example: Convert 225_8 to a decimal number.

```
 225
 x8
 16
+ 2
 18
 x8
144
+ 5
149₁₀ = 225₈
```

To convert an OCTAL fractional number to a DECIMAL number, express the number in power of 8, add and divide as follows:

Example: Convert 0.1142_8 to a decimal number.

$$\begin{aligned}0.1142 &= (1\text{x}8^{-1}) + (1\text{x}8^{-2}) + (4\text{x}8^{-3}) + (2\text{x}8^{-4})\\ &= 1/8 \quad + 1/64 \quad + 4/512 \quad + 2/4096\\ &= 610/4096\\ &= 0.1489^{+} \text{ or } 0.149\end{aligned}$$

Negative Numbers

A convention for representing negative numbers is the use of the minus sign (-). For instance, negative 23 is written as -23. Negative numbers comprise an important part of our number system. The question arises, how does a digital computer represent negative numbers?

SIGN-BIT NOTATION

In the binary number system, the digits 0 and 1 are used to represent any number of our familiar decimal number system. This is also true for negative numbers. The 0 is assigned the plus (+) sign and 1 is assigned the minus (-) sign, when either the 0 or the 1 is found in a specified column. This column is called the "sign bit" column. Using the far left column as the sign bit column, -12 and +22 are written as:

sign
bit

$$\overset{\frown}{1}\ 01100 = -12$$
$$0\ 10110 = +22$$

The confusion which arises when performing mathematic operations and not being able to differentiate between the sign bit and the true magnitude of the number is solved by using the "complemented form" of the negative number. The use of complements is discussed in detail in Mathematics Vol. 3.

All data that is input-output data to a computer does not always represent numerical data. Thus, it has become necessary to develop codes to represent data consisting of both numerical and alphabetical symbols. In the computer, the code relates binary indications to data which may consist of either numerical or alphabetical symbols.

If one understands the principles of coding, memorization of the codes is not necessary. In fact, it is sometimes impossible. There are so many different computer codes that no one individual knows all of them. A small portion of such codes is shown below. They are named after their four lowest integral weights.

5,3,1,1	6,3,1,1
4,3,2,1	5,3,2,1
4,2,2,1	2,4,2,1
8,4,2,1	6,4,2,1

Another fact of which one should be made aware is that two different computers seldom use exactly the same code. If one begins to work with a specific computer, one will become so familiar with it and with its coding that memorization will be automatic.

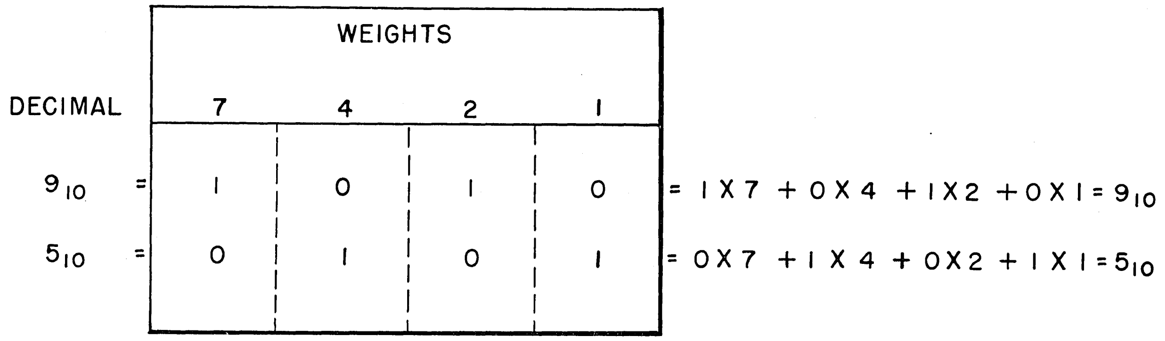
Computer codes can be divided into three categories:

1. Regularly weighted codes
2. Arbitrarily weighted codes
3. Nonweighted codes

REGULARLY WEIGHTED CODES: In these codes, each number position has a weighting value just as the decimal number system and its weighting values are regular. For example 9_{10} and 5_{10} are formed in the 7,4,2,1 code as shown in figure 3-2. These codes are formed according to a specific rule. A regularly weighted code is 7,4,2,1 code. The weighting values form a simple arithmetic series which begins with 1 and increases regularly, plus one unit, plus two units, plus three units, etc.

The important point to remember about regularly weighted codes is that the weights are determined by some rule. If one knows the rule, he can find the weighting value for any desired position.

ARBITRARILY WEIGHTED CODES: These codes also use weighting values, but have no rules for forming them. Consider the 4,2,2,1 code in which the first four weighting values


WEIGHTS
DECIMAL
7
4
2
1
9_{10} =
1
0
1
0
$= 1 X 7 + 0 X 4 + 1 X 2 + 0 X 1 = 9_{10}$
5_{10} =
0
1
0
1
$= 0 X 7 + 1 X 4 + 0 X 2 + 1 X 1 = 5_{10}$

124.6

Figure 3-2.—Formation of 9_{10} and 5_{10} in the 7,4,2,1, code.

from left to right are 4,2,2,1. There is no rule, or formula, for generating these values, and one cannot know what the weighting value of the fifth digit from the right might be. However, one will find that the 4,2,2,1 code, as well as many other arbitrarily weighted codes, can be a very practical one, and that it makes no difference what the fifth digit weight might be, because it is never used.

NONWEIGHTED CODES: These codes have no weighting values at all. Each coded group is defined so as to represent some quantity. The Roman numeral system is such a nonweighted code.

PURE BINARY CODING: Although the decimal code is quite appropriate for use in the small mechanical office calculator, it is impractical for use in the electronic digital computer. One important reason for this limitation, which has been mentioned before, is that electronic digital computers use binary components such as magnetic cores, flip-flops, and diodes, that function best with pure binary numbers. Pure binary calculation is of the utmost simplicity, as shown by the following comparison between binary and decimal addition:

	BINARY	DECIMAL
ADDEND:	100	4
AUGEND:	001	1
SUM:	101	5

(The augend, from Mathematics, Vol. 3, is a number which has another number, the addend, added to it.)

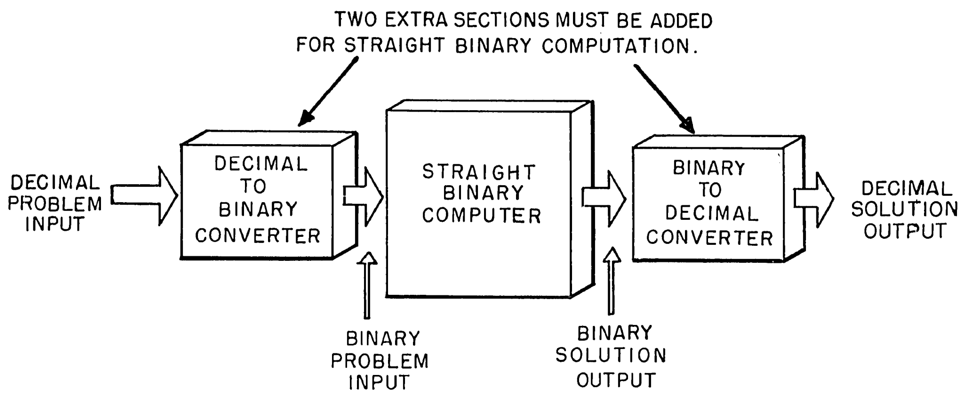
The simplicity of binary calculation means fewer and simpler instructions are required to program the computer. This, in turn, means less hardware and lower cost. For these reasons straight binary coding, as it is called, provides a definite design advantage.

Every design advantage is usually accompanied by some limitations. The major disadvantage is that one must first convert the problem into binary code, and then he must convert the binary solution back into decimal numbers. Obviously, while binary calculations is convenient for the machine's internal operations, the machine's input and output must be able to "talk" man's language.

In order to do this, two translation units are required. These include a decimal-to-binary converter immediately after the input, and a binary-to-decimal converter just before the output. This arrangement is shown in figure 3-3. These converters must be able to convert any number from one system into the other. Because they must handle an infinite number of possibilities, these converters are complex, expensive, and difficult to maintain. Despite these disadvantages, however, the balance remains in favor of the pure binary code, and many of the early computers were built in accordance with this system.

BINARY-CODED DECIMALS—(A COMPROMISE)

Our decimal number language is different from the one used by a large electronic computer. Although we understand decimal numbers best, all digital computers, (except the simplest mechanical calculators), function best by using the pure binary number system. In order to bridge the two systems, it was necessary to sacrifice some of the computer's simplicity by adding complex encoding and decoding circuits.


TWO EXTRA SECTIONS MUST BE ADDED
FOR STRAIGHT BINARY COMPUTATION.
DECIMAL PROBLEM INPUT
DECIMAL TO BINARY CONVERTER
STRAIGHT BINARY COMPUTER
BINARY TO DECIMAL CONVERTER
DECIMAL SOLUTION OUTPUT
BINARY PROBLEM INPUT
BINARY SOLUTION OUTPUT

124.7

Figure 3-3.—Relationship of converters to the straight binary computer.

At the present time no complete solution has been developed for this problem. Even the best modern computer employs a system that is a compromise. Thus, one of the better modern computers uses a combination of both the binary and decimal codes, and the outstanding features of its operation will be described in the paragraphs that follow.

Recall that each decimal digit can be represented by a binary number, as shown in figure 3-4. Each of the binary numbers that is shown in the figure is called a ''binary-coded decimal digit'' or ''coded digit.'' One can see that without using any other binary numbers, it is possible to represent any desired quantity by means of these coded digits. For example, in order to represent 736, we simply write the coded digit for each decimal, then we place the digit in the same order as the decimals thus: 011100110110.

Binary coded decimal digits need not be spaced because we know that each one is composed of four binary digits. For easier reading, however, a space usually will be placed between each coded digit, thus: 0111 0011 0110. Note that this number is not pure binary; it is, instead, a binary-coded decimal number, as shown in figure 3-5.

This notation possesses important advantages for computer applications. Any decimal number can be expressed in the binary-coded decimal system, but only ten different conversions are necessary—one for each decimal. Therefore, the machine is designed to compute with each coded decimal as its basic unit, not with each binary digit. Thus, although the computer never operated with anything but binary numbers, it remains, from one point of view, a decimal calculator.

DECIMAL DIGIT	BINARY-CODED DECIMAL DIGIT
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

124.8

Figure 3-4.—Binary-coded decimal digits.

$$\begin{tikzcd}
7 \arrow[d] & 3 \arrow[d] & 6 \arrow[d] \\
0111 & 0011 & 0110
\end{tikzcd}$$

124.9
Figure 3-5.—Binary-coded decimal digits can represent any decimal number.

Binary-coded decimals are not a complete solution, because they are longer than pure binary numbers. For example, the number 0111 0011 0110 (12 bits) can be written in pure binary code a 1011100000 (10 bits). Thus, more components are needed to represent the extra digits of binary-coded decimals. Calculating circuits must also be made more complex, in order to process each coded-digit as a unit.

There is a net gain, however. The increase in calculating circuit complexity is more than balanced by the vastly simplified encoding and decoding circuits.

THE BIQUINARY CODE

This is a binary-coded decimal system that uses seven-bit words instead of four-bit words. (See figure 3-6.) The code derives its name from the fact that we can conveniently divide it into two parts: the ''bi''-part, which consists of the first two bits at the left, and the ''quinary''-part, which makes up the other five digits. With this code, the decimal number 306 is written as:

0101000 0100001 1000010

The major disadvantage of this code is obvious. It requires more bits per word, and is therefore more expensive. On the other hand with this code, it is easy to detect an error. Note that every single biquinary digit representation has exactly two 1's in it; no more, no less.

Let us assume, for example, that a computer contains the biquinary number 2, using a positive pulse to represent 1, and no pulse to represent 0, as shown in figure 3-7. Let us assume further that during the process of passing this coded pulse train from one circuit to another, a pulse is dropped. (See figure 3-6.)

We can easily determine that an error has occurred, even if we do not know the exact value of each biquinary number. The error will be detected when we fail to find two 1's in the pulse train. This method of looking for pairs is called a parity check.

DECIMAL DIGIT	BIQUINARY CODE		8421 CODE
	"BI" 50	"QUINARY" 43210	8 4 2 1
0	01	00001	0000
1	01	00010	0001
2	01	00100	0010
3	01	01000	0011
4	01	10000	0100
5	10	00001	0101
6	10	00010	0110
7	10	00100	0111
8	10	01000	1000
9	10	10000	1001

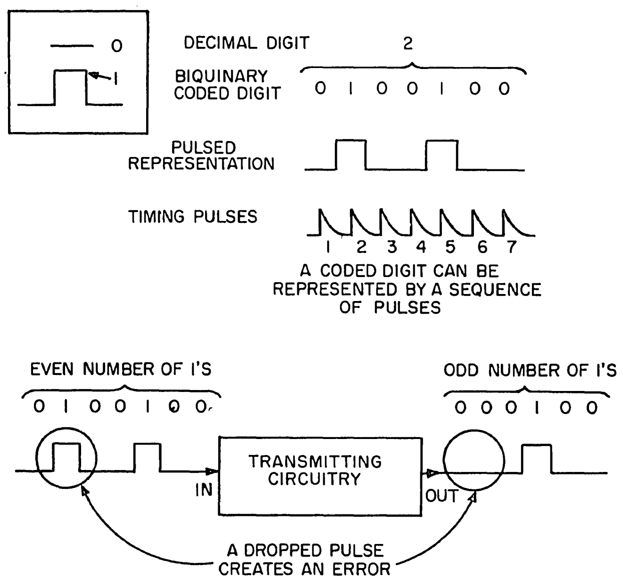
BIQUINARY-CODED DIGITS CAN REPRESENT ANY DECIMAL NUMBER

$$\begin{tikzcd}
3 \arrow[d] & 0 \arrow[d] & 6 \arrow[d] \\
010\,1000 & 0100001 & 1000010
\end{tikzcd}$$

124.10
Figure 3-6.—Comparing the biquinary with the 8,4,2,1 code.

ERROR DETECTING CODES

The biquinary code is only one of a class of codes called two-out-of-seven codes, all of which have the same error-detecting property. The quibinary code, for example, uses a different



124.11
Figure 3-7.—Biquinary code.

notation to produce the same effect. Its seven weighting values are

8,6,4,2,0,1,0

and its decimal equivalents are given in figure 3-8.

Another group of codes, called two-out-of-five codes, possess error-detecting qualities, but are less expensive. As their name implies, two-out-of-five codes require only five bits instead of seven bits to represent a decimal digit.

A two-out-of-five code can be generated very simply from the 7,4,2,1 code. Remember that this code never has more than two 1's in any single word. Now add a fifth bit to each four-bit word in such a manner that the total number of 1's per word is always an even number. In other words, if there is one 1 in a coded digit, 0010 for example, then another 1 is added to it, making it 00101. On the other hand, if there are two 1's in the word, such as 0110, then a zero is added to produce 01100, thus retaining the quality of evenness. The 7,4,2,1 code, converted into a two-out-of-five code, is shown in figure 3-9.

Note that each word has exactly two 1's in it, except for the zero, which has none. Although this complete lack of the 1 bit does no harm to the code's error-detecting properties, it can cause trouble in certain instances. If 1 is represented by a pulse, and 0 is represented by no pulse, there is no method of telling whether 00000 means zero or whether it means a complete absence of information (fig. 3-10). Although this dilemma can be resolved by means of the computer's timer (a count of five time-units with no pulses is defined as representing zero), it is safer to have 1's in every word. Besides, it is not difficult to find a 7,4,2,1 code word that has not been used. For example, the decimal number eleven (7 + 4) in the 7, 4, 2, 1 code is the word 1100. Since we do not need this "spare," we can use it to represent zero. If we add a 0, to its right, we have 11000, which satisfies the two-out-of-five requirements perfectly (see fig. 3-11).

We are obviously willing to pay for reliability with an extra bit per word, although these extra bits possess no code values. If these extra bits represent no number, how does the computer calculate with them? One answer is that if it is provided with enough circuitry, the computer can calculate with

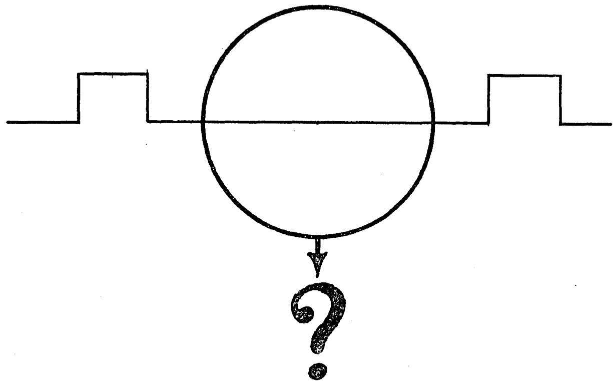
DECIMAL DIGIT	QUI WEIGHTS	BINARY WEIGHTS
	8 6 4 2 0	1 0
0	0 0 0 0 1	0 1
1	0 0 0 0 1	1 0
2	0 0 0 1 0	0 1
3	0 0 0 1 0	1 0
4	0 0 1 0 0	0 1
5	0 0 1 0 0	1 0
6	0 1 0 0 0	0 1
7	0 1 0 0 0	1 0
8	1 0 0 0 0	0 1
9	1 0 0 0 0	1 0

124.12
Figure 3-8.—The quibinary code.

DECIMAL DIGIT	WEIGHTS 7 4 2 1 0
0	00000
1	00011
2	00101
3	00110
4	01001
5	01010
6	01100
7	10001
8	10010
9	10100

PARITY CHECK COLUMN

124.13
Figure 3-9.—Two-out-of-five code.



124.14
Figure 3-10.—Is it 00000 ..... or simply a pause between pulses?

almost any number system. However, this is not a very economical method of calculation. The method more frequently used is to remove the redundant bit (or parity bit, as it is sometimes called) just before the coded number enters a calculating circuit, and to replace the redundant bit as soon as the coded number leaves the circuit. This method requires fewer extra circuits.

For further economy, the parity bit can be removed before a word is stored, and it can be added on as soon as the word leaves storage. One important fact about the error-detecting codes so far discussed is that they have been misnamed. The term "error-detecting" is not completely accurate. Consider, for example, a situation in which the word 01001 enters a faulty component, and leaves it with two errors, such as 00101. The comparator will allow the erratic 7,4,2,1-coded word to pass, for it has an even number of 1's. Obviously, the "error-detecting" codes should be called "single-error detecting" codes for that is the full limit of their capability.

DECIMAL DIGIT	WEIGHTS 7 4 2 1 0
0	11000
1	00011
2	00101
3	00110
4	01001
5	01010
6	01100
7	10001
8	10010
9	10100

AN IMPROVEMENT

124.15
Figure 3-11.—Revised two-out-of-five code.

ERROR-PREVENTING CODES

We have already considered error-detecting codes, and one should understand that these codes are useful only after the error has been committed. Sometimes, however, it is important to detect an error, or at least to minimize the error, before it occurs. This is where error-preventing codes can be used.

Gray Code

An analog-to-digital converter frequently uses error-preventing codes. Basically, an analog-to-digital converter is a device that changes a number from analog to digital form. If the analog value is a mechanical movement, the quantity can be expressed in terms of a specific angle of shaft rotation or by the longitudinal movement of a rack and pinion. An error-preventing code frequently used with an analog-to-digital converter is the Gray code. This Gray code is a cyclic code which changes only by one bit when going from one number to the one immediately following. Operational errors are reduced if only one digit at a time changes as the numerical value increases.

Let us examine the Gray code in figure 3-12. If we cover the leftmost digit in the first and last rows under the reflected binary code, we will observe that the bits that represent 0 are identical with the bits that represent 15. Thus, all but the leftmost digit of 0000 and 1000 "reflect" each other. Similarly, the number that represents 1 is reflected by the number that represents 14, etc.

A simple method of constructing the Gray code is to employ the following steps:

1. Write the number 00, and the number 01 below it.
```
00
01
```
2. Now "reflect" the least significant digits of these two numbers, thus
```
0
1
——    reflection line
1
0
```
3. Place 0's to the left of the top half of the numbers and 1's to the left of the bottom half.
```
00
01
11
10
```
4. Again, "reflect" these numbers, thus
```
00
01
11
10
——    reflection line
10
11
01
00
```
5. Once more, place 0's to the left of the top half of the numbers and 1's to the left of the bottom half.
```
000
001
011
010
110
111
101
100
```

DECIMAL DIGIT	REFLECTED BINARY CODE
0	0 0 0 0
1	0 0 0 1
2	0 0 1 1
3	0 0 1 0
4	0 1 1 0
5	0 1 1 1
6	0 1 0 1
7	0 1 0 0
8	1 1 0 0
9	1 1 0 1
10	1 1 1 1
11	1 1 1 0
12	1 0 1 0
13	1 0 1 1
14	1 0 0 1
15	1 0 0 0

124.16

Figure 3-12.—The Gray code.

6. Reflect, and again add 0's to the left of the top half of the numbers and 1's to the left of the bottom half. Remember to add these extra 0's to the left.

```
0000
0001
0011
0010
0110
0111
0101
0100
____    reflection line
1100
1101
1111
1110
1010
1011
1001
1000
```

Of course, we could continue this process of adding and reflecting, thus making this code as large as we desire.

The reflected binary code that has been constructed, is called the Gray code. It is only one of a number of reflected binary codes. These codes, in turn, belong to a large group of cyclic codes.

FUNDAMENTALS OF BOOLEAN ALGEBRA

As stated previously, a digital computer uses bi-stable elements as electrical components in its circuitry. These bi-stable elements act as electronic switches. These electronic switches are put in a definite order to perform certain prescribed tasks such as adding and comparing.

When designing switching circuits for the first digital computers, it became evident that a simple mathematical way of representing switching circuits was needed so that these circuits could be simplified. Upon investigation and research, it was discovered that Boolean algebra could be applied to switching circuits.

Boolean algebra is based on George Boole's book, An Investigation of the Laws of Thought, On Which are Founded the Mathematical Theories of Logic and Probabilities, published in 1854. His book led to the development of a "logical algebra" used to design logical circuitry in digital computers. This algebra will be discussed in the following paragraphs. A more detailed discussion may be found in Mathematics, Vol. 3.

BASIC CIRCUITS AND SYMBOLS

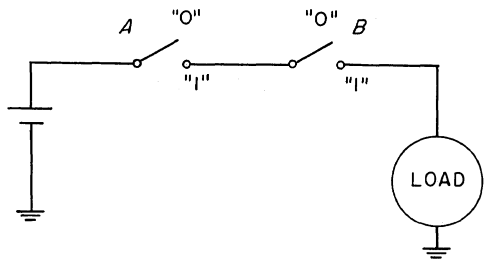
Boolean algebra uses the mathematical symbols of "·", "+", and "-" to represent a logical description. The "·" represents AND, the "+" represents OR, and the "-" represents NOT. With these three symbols, one can represent the basic logic switching circuits of a digital computer.

AND Circuit

The AND circuit of two switches (variables) is shown in figure 3-13A. Switch A or B may be either open or closed, 0 or 1 position. The series circuit will transmit a signal only if both A AND B are closed, i.e., equal to 1. If either switch A or switch B is open, i.e., equal to 0, then the circuit will not transmit a signal.

The preceding AND circuit is represented in Boolean algebra as A·B, which means A AND B. Note that the variables A and B can only take on the values of either 0 or 1.

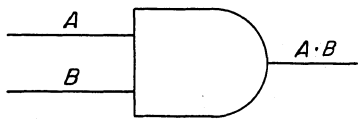
(A)



(B)

A	B	$f(A,B) = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

(C)

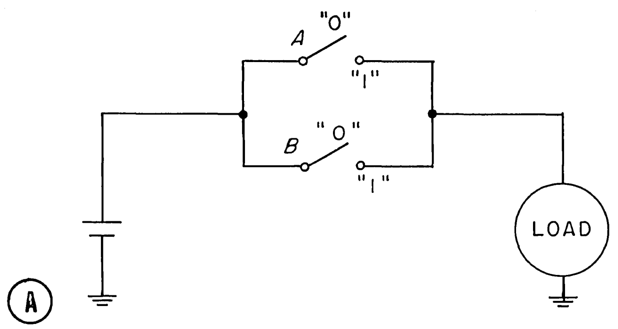


12.133

Figure 3-13.—The AND circuit.

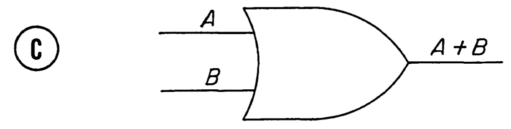
In Boolean algebra any group of variables, which represents an expression of logic, is called a function and is symbolized "f". For any Boolean function there is a corresponding truth table which shows, in tabular form, the true condition of the function (0 or 1) for each way in which conditions can be assigned to its variables. The associated AND circuit truth table (fig. 3-13B), shows the various condition of 0 and 1 assigned to variables A and B, and the AND circuit outputs f(A,B) to the load.

In any digital computer, there will be many AND circuits. In order to analyze circuit operation, it is necessary to refer frequently to these circuits without looking at their detailed electronic switching arrangements. This simplification is accomplished by substituting logic diagram mechanizations for the actual circuits. The logic symbol for the AND circuit is illustrated in figure 3-13C. Two coincident inputs, A and B at the left will produce the output function A·B in Boolean form at the right.



B

A	B	f(A,B) = A+B
0	0	0
0	1	1
1	0	1
1	1	1



12.134

Figure 3-14.—The OR circuit.

OR Circuit

The OR circuit is shown in figure 3-14A with its corresponding truth table (fig. 3-14B), and logic symbol (fig. 3-14C). Notice that the circuit switches are placed in parallel. Inspection of the arrangement shows that the circuit will transmit a signal to the loads if either A OR B is in the closed position, i.e., equal to 1. If, and only if, both A and B are open, i.e., equal to 0, the circuit will not transmit.

The logic symbol for the OR circuit is illustrated in figure 3-14. Either of two inputs A or B at the left will produce the output function A+B in Boolean form at the right.

NOT Function

Figure 3-15A shows the NOT function and its corresponding truth table (fig. 3-15B) and logic diagram mechanization (fig. 3-15C). The requirements of a NOT condition are that a signal applied to the input produce the complement (reverse or opposite) of the signal at the output, and furthermore, that the complement of the signal at the input produce the signal at the output. Thus, in figure 3-15A when switch A is closed, i.e., equal to 1, the relay opens the circuit to the load, and the output is 0. When switch A is open, i.e., equal to 0, the relay completes a closed circuit to the load, and the output is 1.

The logic diagram (fig. 3-15C) for the NOT function indicates one input, say A, produces the output function in Boolean form of $\bar{A}$.

NOR Function

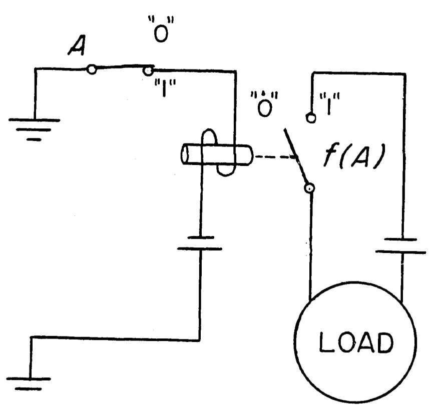
The NOR function is produced by negating the output of an OR circuit (fig. 3-16A). Thus, the logic diagram mechanization is as illustrated in (fig. 3-16B). The truth table of the NOR function is illustrated in figure 3-16C.

NAND Function

The NAND function is produced by negating the output of an AND circuit (fig. 3-17A). Thus, the logic diagram mechanization is as illustrated in (fig. 3-17B). The truth table of the NAND function is given in figure 3-17C.

The Boolean algebraic symbols for the AND and OR circuits may be used to describe various

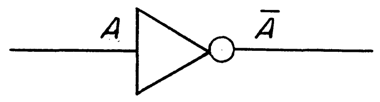
(A)



(B)

A	$f(A) = \bar{A}$
1	0
0	1

(C)

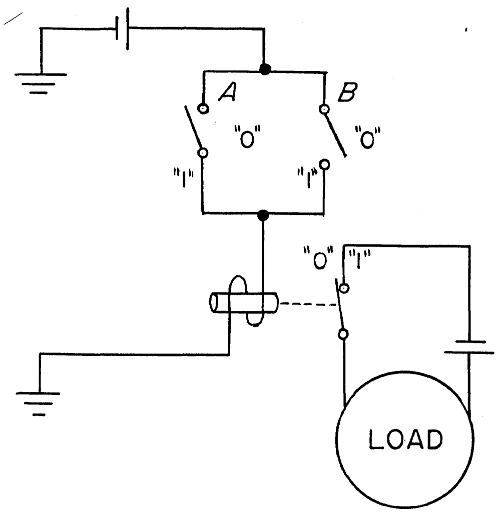


12.135
Figure 3-15.—The NOT function.

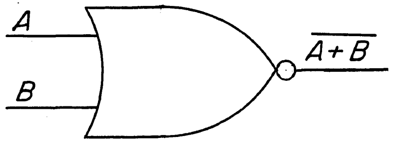
combinations of these circuits as illustrated in figure 3-18. Between points 1 and 2 switches A, B, and C are in parallel and may be represented in Boolean form as A + B + C. Between points 2 and 3, switches D and E are in series and may be represented as DE. Thus the parallel series combination between points 1 and 3 is equal to (A + B + C)DE. Between points 4 and 5 the parallel combination of F and G is in parallel with the series combination of H and I. Thus the Boolean expression for the circuit between points 4 and 5 is F + G + HI.

Between points 3 and 6 the series arrangement of switch J, K, and L may be represented as JKL. This combination is effectively in parallel with the circuit between points 4 and 5 and the circuit between points 3 and 6 may be represented in Boolean form as F + G + HI + JKL. The circuit between points 3 and 6 is in series with the circuit between points 1 and 3; thus the entire circuit from point 1 to point 6 may be expressed as DE (A + B + C)(F + G + HI + JKL).

(A)



(B)



(C)

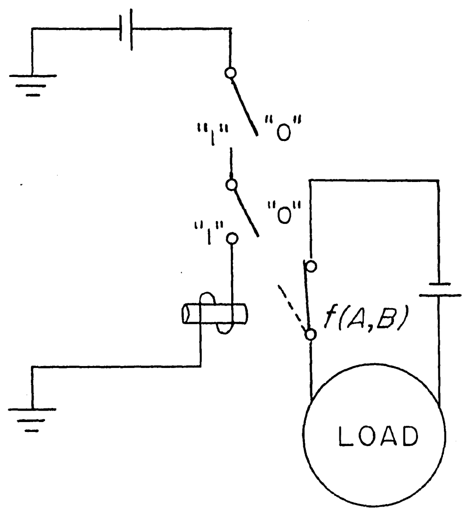
A	B	$A+B$	$f(A,B) = \overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

67.89
Figure 3-16.—The NOR function.

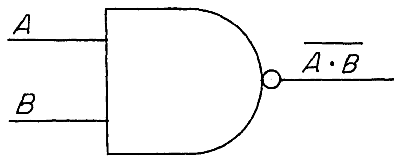
LOGIC DIAGRAM MECHANIZATION

Figure 3-19 illustrates the switching circuit diagram for the Boolean expressions (A + B + C) ABA(B + C + AD) and [AB (A + B + C) + CDE] EB.

A



B



C

A	B	AB	$f(A,B) = \overline{A \cdot B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

67.90

Figure 3-17.—The NAND function.

We now have a method of describing switching circuits mathematically with Boolean algebra. This algebra, like conventional algebra, follows certain laws and axiomatic expressions. These laws and axiomatic expressions are used to simplify Boolean expressions. These methods of simplification are discussed in detail in Mathematics, Vol. 3. It is advised that the reader study the applicable chapters in order to grasp more fully number systems and basic logic.

VEITCH DIAGRAMS

The Veitch diagram, a very quick and easy way for finding the simplest logical equation needed to express a given function, is discussed in the following paragraphs.

Veitch diagrams for two, three, four, five, or more variables are readily constructed. Any number of variables may be plotted on a Veitch diagram, though the diagrams are difficult to use when more than four variables are involved. The Veitch diagrams for two, three, and four variables are illustrated in figure 3-20.

Since each variable has two possible states (0 and 1), the number of squares needed is the number of possible states (two) raised to a power equal to the number of variables. Thus, for four variables the Veitch diagram must contain 2^4 or 16 squares. Five variables require 2^5 or 32 squares. An eight-variable Veitch diagram needs 2^8 or 256 squares—a rather unwieldy diagram. If it becomes necessary to simplify logical equations containing more than six variables, other methods of simplification are available and are discussed in Mathematics, Vol. 3.

An exploded view of a four-variable Veitch diagram is shown in figure 3-21. Notice the division marks which divide the diagram into labeled columns and rows. The location of each square represents the combination of the variables labeling each row and column.

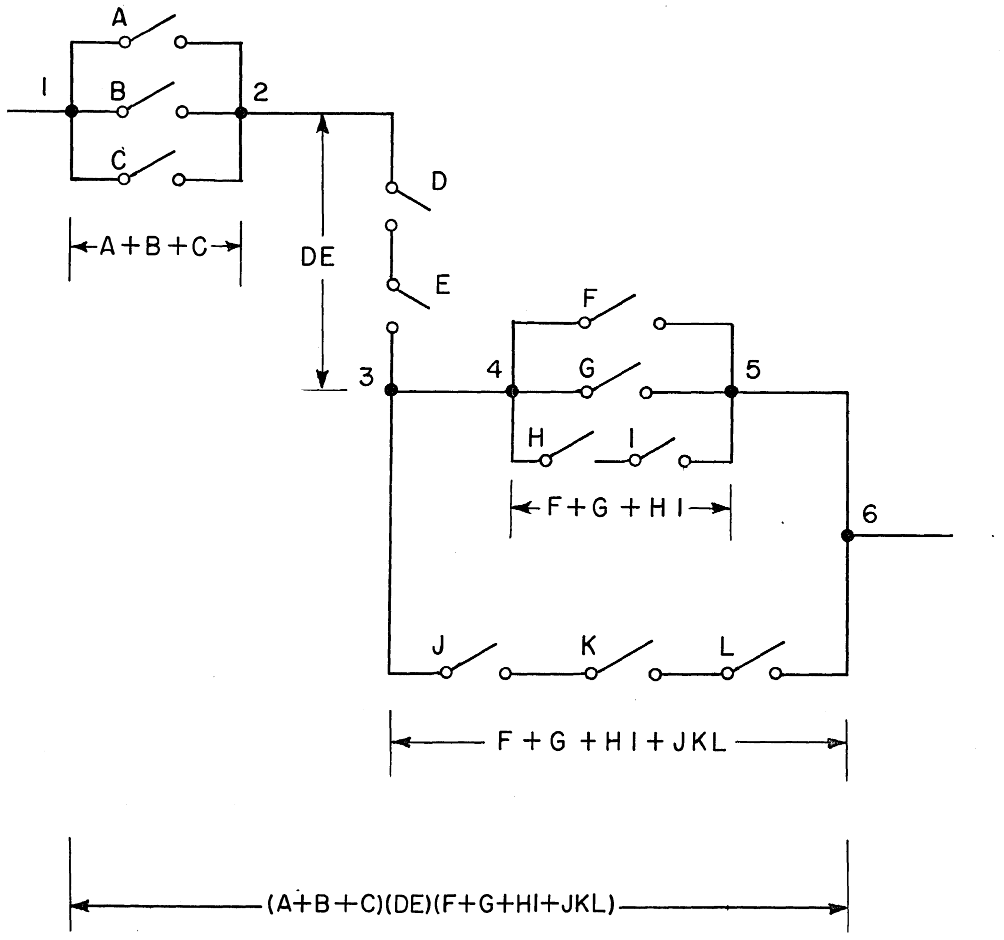
To illustrate the plotting of the Veitch diagram, the Boolean equation $f\ (A,B,C,D) = ABC + AB\overline{D} + A\overline{C} + \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}C$, will be used. Keep in mind the purpose of the plot is to simplify the given equation. First plot the equation by placing 1's in appropriate blocks corresponding to the variables ABC and D of the various terms (fig. 3-22).

Then simplify the plot using certain rules (fig. 3-23) to obtain the simplified form (fig. 3-24) of the given equation.

The plot is begun by placing 1's for the first term in the indicated squares as follows:

f(A,B,C,D) = ABC squares 14 and 15.

Consider the extensions AB and C (fig. 3-21) to be cut out and placed over the original diagram (center portion) so the numbered squares on the extensions coincide with the corresponding numbers on the original. Only squares 14 and 15 are common to all three variables A, B, and C.



67.92

Figure 3-18.—A switching circuit.

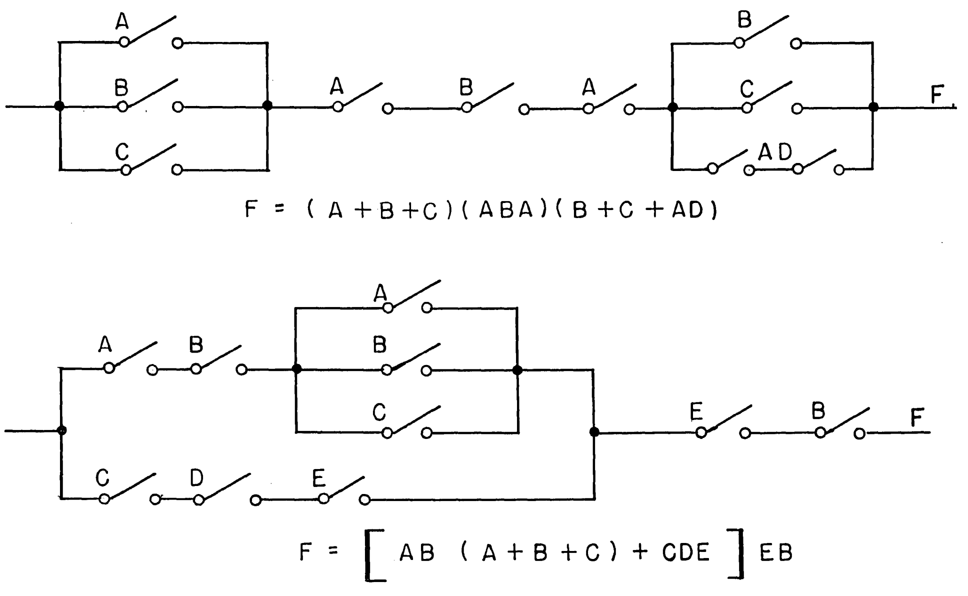
In a similar manner the second term ABD of the given equation is plotted by placing 1's in squares 12 and 14. The entire process of plotting this equation is summarized as follows:

$f(A,B,C,D) = ABC$ . . . . squares 14 and 15
$= +AB\overline{D}$ . . . squares 12 and 14
$= +A\overline{C}$ . . . . squares 12, 13, 9, and 8
$= +\overline{A}\overline{B}\overline{C}\overline{D}$ . . square 0
$= +\overline{A}C$ . . . . squares 6, 7, 3, and 2

The completed Veitch diagram is illustrated in figure 3-22. Notice that when representing a term which does not contain all the variables (A, B, C, and D) the plotting will represent both the complemented and the noncomplemented forms of the variables which were missing from the term. Refer to the plot of $A\overline{C}$. One's are placed in squares 12, 13, 9 and 8. The missing variables in this term are B and D. Their complements are $\overline{B}$ and $\overline{D}$. Note that squares 12, 13, 9, and 8 represent the plot for $A\overline{C}$ and also for the missing terms B and D and their complements $\overline{B}$ and $\overline{D}$.

TRUTH TABLES

The decimal numbers in the upper lefthand corner of each square in figures 3-21 and 3-22 correspond to the decimal numbers in the left-hand column of table 3-1. These numbers identify the various combinations of the variables ABC and D in the Veitch diagram. The equation $f = ABC + AB\overline{D} + A\overline{C} + \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}C$ satisfies



67.92

Figure 3-19.—Switching circuit examples.

f = 1 for certain conditions illustrated in the truth table and they are the ones plotted on the Veitch diagram (fig. 3-22).

Figure 3-22 shows the Veitch diagram of $f = ABC + AB\bar{D} + A\bar{C} + \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}C$. We will also use this expression in discussing the use of the Veitch diagram in simplification.

To obtain the simplified logical equation from a Veitch diagram of four variables, observe the following rules: (fig. 3-23).

a. If 1's are located in adjacent squares or at opposite ends of any row or column, the variable appearing in its complemented and noncomplemented form of the variables may be dropped.
b. If any row or column of squares, any block of four squares, or the four end squares of any adjacent rows of columns, or the four corner squares are filled with 1's, two of the variables which are contained in the complemented and noncomplemented forms may be dropped.
c. If any two adjacent rows or columns, the top and bottom rows, or the right and left columns are completely filled with 1's, three variables may be dropped.

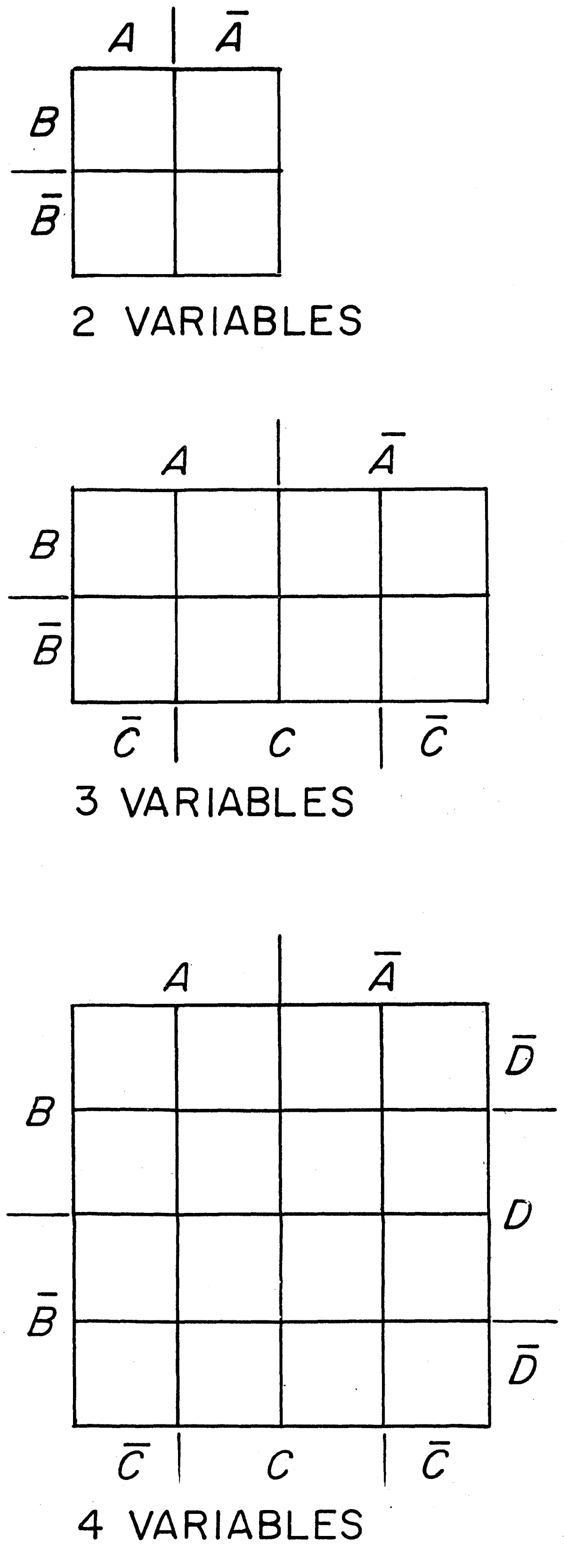
To reduce the original equation to its simplest form, sufficient simplification must be made until all 1's have been included in the final equation. One's may be used more than once, and the largest possible combination of 1's in groups of 8,4, or 2 should be used.

Table 3-1.—Truth Table of $f = ABC + AB\bar{D} + A\bar{C} + \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}C$.

	Decimal	A	B	C	D	f
Decimal equivalent of the binary represented variables	0	0	0	0	0	1
	1	0	0	0	1	0
	2	0	0	1	0	1
	3	0	0	1	1	1
	4	0	1	0	0	0
	5	0	1	0	1	0
	6	0	1	1	0	1
	7	0	1	1	1	1
	8	1	0	0	0	1
	9	1	0	0	1	1
	10	1	0	1	0	0
	11	1	0	1	1	0
	12	1	1	0	0	1
	13	1	1	0	1	1
	14	1	1	1	0	1
	15	1	1	1	1	1

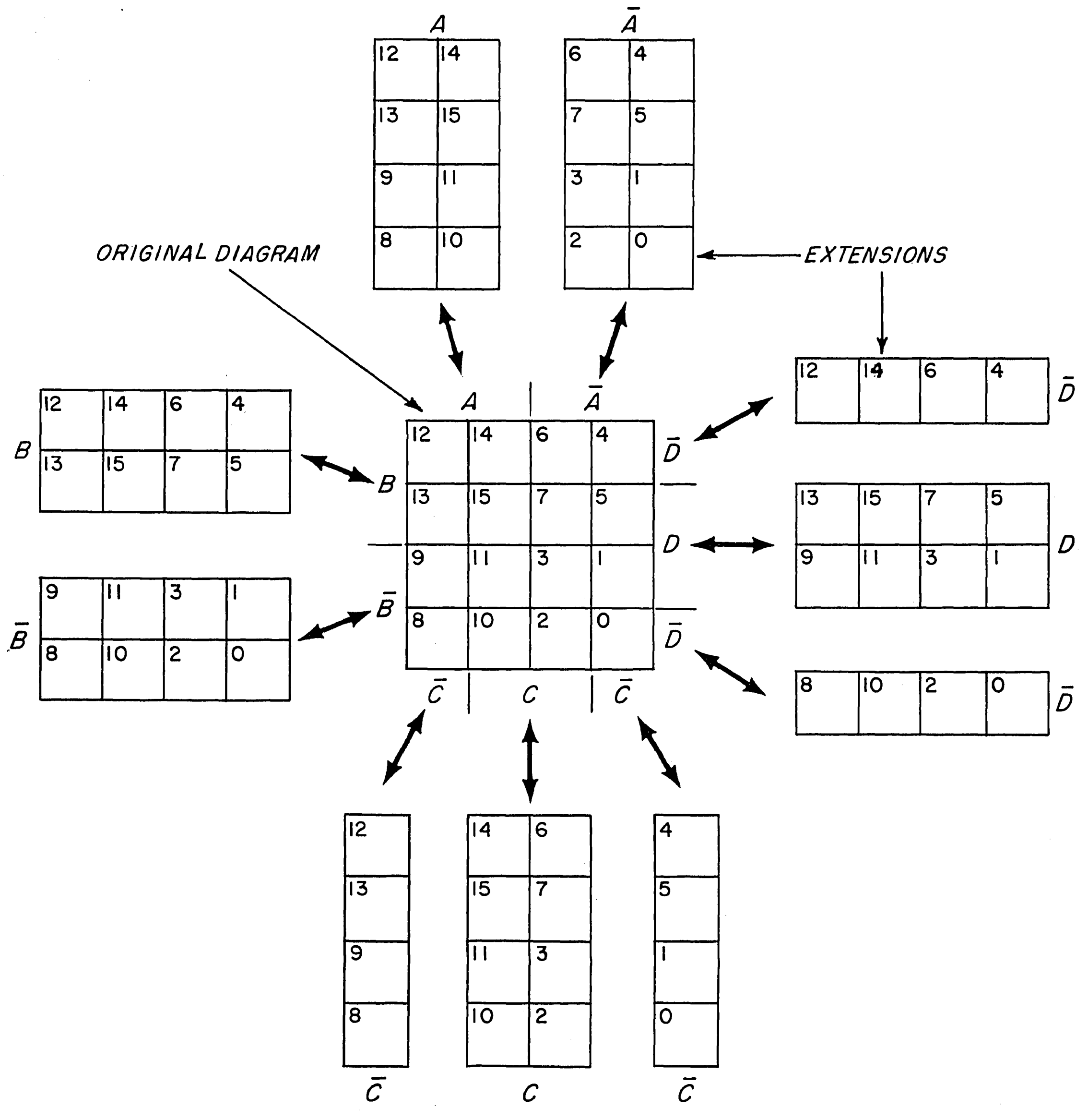
To proceed with the simplification of $f = ABC + AB\bar{D} + A\bar{C} + \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}C$:

1. Squares 12, 13, 9, and 8 may be combined using rule (b) to yield $A\bar{C}$.
2. Squares, 6, 7, 3, and 2 may be combined using rule (b) to yield $\bar{A}C$.



124.20

Figure 3-20.—Veitch diagram.



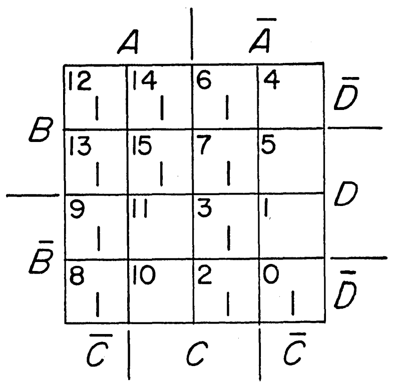
124.21

Figure 3-21.—Exploded Veitch diagram.

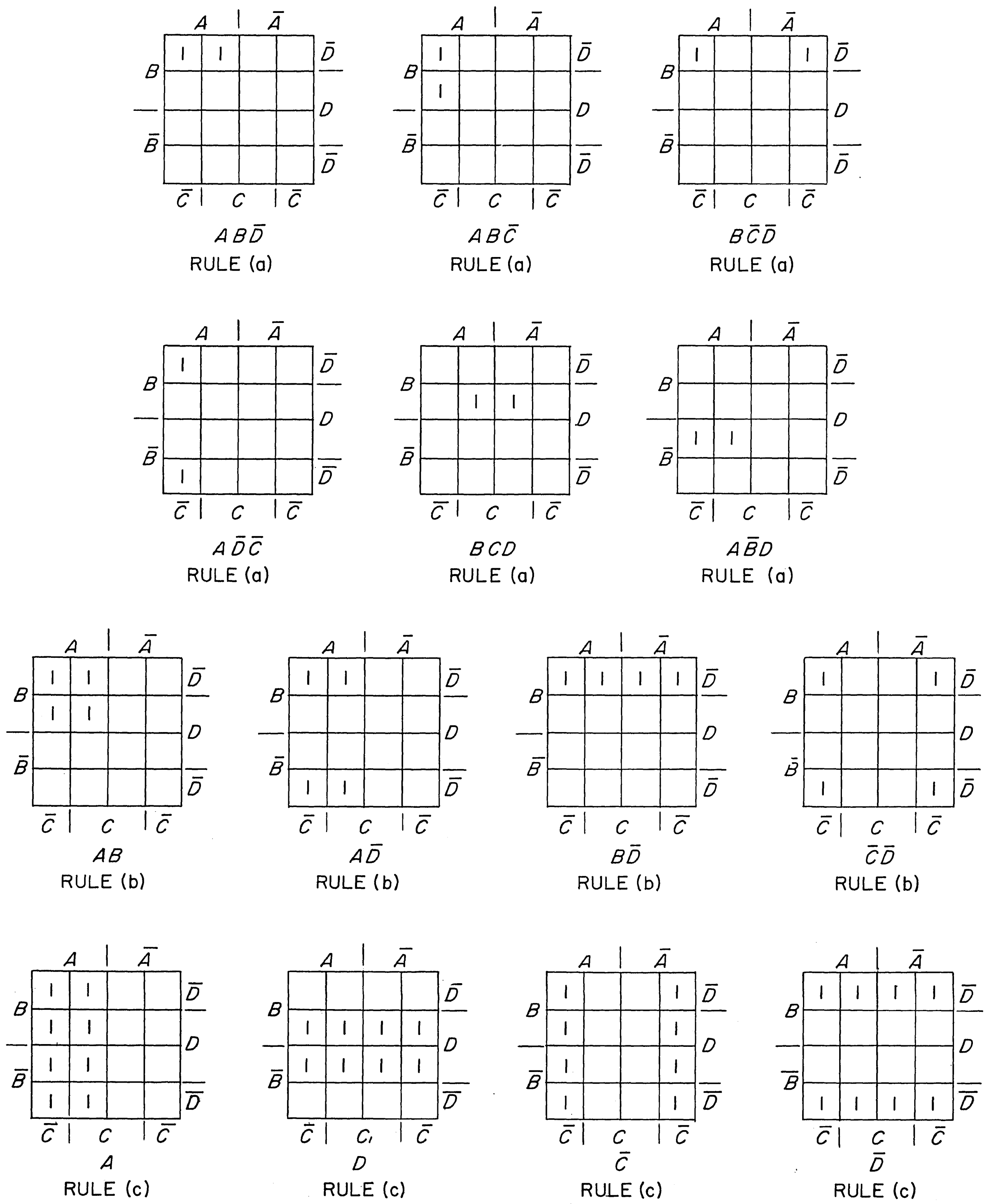
3. Squares 12, 14, 13, and 15 may be combined using rule (b) to yield AB.
4. Squares 0 and 2 may be combined using rule (a) to yield $\bar{A}\bar{B}\bar{D}$.

To keep track of the squares combined, draw loops around the combined squares. Doing this, the Veitch diagram is modified as in figure 3-24. All 1's have been used at least once, therefore, the Boolean expression can now be written in its simplest form as

$$f = AB + A\bar{C} + \bar{A}C + \bar{A}\bar{B}\bar{D}.$$



124.22
Figure 3-22.—Veitch diagram of f(A,B,C,D) = ABC + ABD + AC + ABCD + AC.



124.23

Figure 3-23.—Veitch combinations.

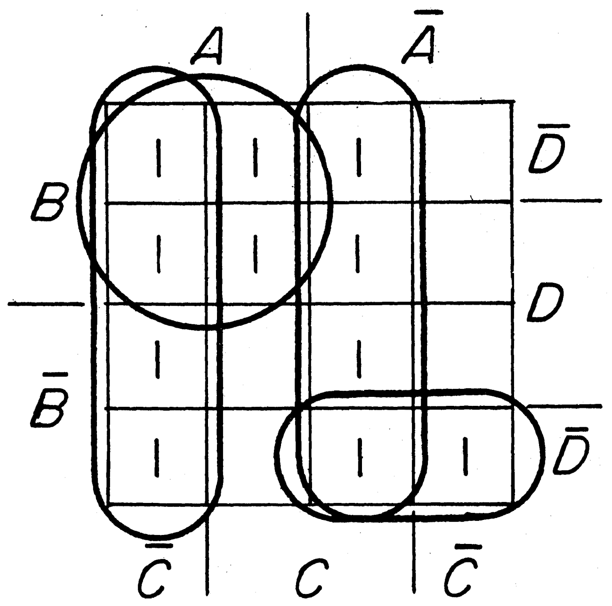


Figure 3-24.—Simplification of $f = ABC + AB\bar{D} + A\bar{C} + \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}C$ to $f = AB + A\bar{C} + \bar{A}C + \bar{A}\bar{B}\bar{D}$.

CHAPTER 4

CONTROL UNIT

INTRODUCTION

The control unit is the coordinator or director of all operations within the computer. Its actions include: directing the reading of information from memory; controlling the inputs and outputs of the computer; directing the operations within the arithmetic unit; and transferring information back into memory. To perform these actions, it is necessary for the control unit to be aware of each operation that is to be performed, the location of the data involved in the operation, and of course, where to place the results.

Actually, the control section must be able to tie together any circuit or group of circuits necessary to perform any operation capable of being performed by the computer. One might expect, and rightfully so, that a maze of components, wires, and circuits must be contained in the computer. However, this is minimized somewhat by design and by using a single circuit to perform many functions.

This chapter treats the control section by (1) considering the method of representing computer instructions; (2) discussing the functional operation of some circuits contained in the control section; and (3) discussing a few of the ways by which the control section coordinates the computer operations.

SPECIAL PURPOSE AND GENERAL PURPOSE COMPUTERS

There are two types of digital computers, special purpose and general purpose. Special purpose computers, as implied by the name, are wired to perform specific types of operations in a planned sequence. Changing the operation and the sequence by which they are performed necessitates changes in the wiring of the computer. Of course, this limits the usefulness of this type of computer to special applications and to the solution of problems which are not dependent on certain prevailing conditions.

General purpose computers, again as the name implies, are much more versatile since their circuits are designed to perform many different types of operations depending on the programmed instructions. The general purpose computer can be made to alter its sequence of operations based on conditions which occur while solving the problem. Thus, the programmed sequence can be changed or jumped to any step in the program on ''condition'', i.e., if certain conditions prevail. The sequence may be altered ''unconditionally'', i.e., without regard to conditions.

This text discusses the operation of general purpose computers although many of the circuits and procedures are common to both types.

The general purpose computer performs its actions as determined by a repertoire of instructions or program which has been previously read into its storage or memory section in the form of computer words. For this reason, the general purpose computer is also referred to as a ''stored-program'' computer. Each computer instruction word is intelligible to the computer and is interpreted by the control unit and executed in sequence unless conditions derived during the calculations dictate otherwise.

INSTRUCTION WORD

It has been pointed out in an earlier chapter that computer circuits respond to ''on-off'' pulses. The presence of a pulse usually represents a 1 condition, whereas the absence of a pulse represents a zero condition. Each of the on-off pulses in an instruction word is referred to as a ''bit.'' In chapter 3 it is shown how 1's and 0's (bits) are combined to represent BCD and octal digits. Table 4-1 illustrates how digits in binary-coded decimal form are used to represent 4593.

Table 4-1.—Binary-Coded Decimal Representation of 4593.

BINARY-CODED DECIMAL REPRESENTATION	0100	0101	1001	0011
DECIMAL VALUE	4	5	9	3

In octal form the number 27563 is represented as shown in table 4-2.

Table 4-2.—Octal Representation of 27563.

BINARY VALUE OCTAL REPRESENTATION	010	111	101	110	011
OCTAL VALUE	2	7	5	6	3

Significantly, the number represented in table 4-2 may not be interpreted by the control section as twenty-seven thousand five hundred and sixty three, but rather as an instruction word. The two left-most digits may be a computer code for "READ-OUT." The remaining portion of the number (563) may be interpreted as an address in memory. Thus, the computer interprets the group of characters represented in table 4-2 as an instruction word, meaning to read-out the contents of memory address 563.

Computer words also represent data, and in fact, it is often not possible (by visual inspection) to distinguish between data and instructions. It is therefore necessary to exercise care when organizing words in storage and to keep close track of the location of instructions and data stored in memory.

OPERATION CODE SECTION

Each instruction word contains at least two sections (table 4-3). The first section (65 in this case), consisting of one or more digits, indicates the function or operation code, i.e., ADD, SUBTRACT, etc. The second part can be the operand, although it most frequently contains the address (or addresses) of the data (operand) involved. The operand may represent the number to be used in an arithmetic operation that is to be performed, it may be transferred, left shifted or right shifted, read-out, or any type of computer operation.

Table 4-3.—Basic Instruction Word.

	Operation Code		Address (or OPERAND)			
OCTAL REPRESENTATION	6	5	5	3	4	0
BINARY REPRESENTATION	110	101	101	011	100	000

The number of digits used to represent the operation code is sometimes determined by the number of different types of operations the computer can perform. If it is assumed that a given computer can only add, subtract, multiply, and divide, the particular operation code could be represented by the four possible combinations of two binary digits. Here, 00 could represent ADD; 01 could represent subtract; 10, multiply; and 11, divide. Thus, the address/operand section of the word will be preceded by one of these codes depending on the operation desired.

Generally, computers can perform many more than 4 operations as described above. In fact, some computers can perform more than 200 different types of operations. Where this is true, the operation code must contain a sufficient number of digits to represent any operation desired.

ADDRESS SECTION

The second part of the instruction word, namely the address section, generally represents the memory address (or addresses) of the data to be operated on (operand). It should be understood that in the stored program computer, the instruction word does not normally contain the operand but merely the address of the operand.

To further explain this point, assume that a certain computer has 32,000 memory address registers, each of which can store a computer word (either an instruction word or an operand). Assume further that the operation code 134 means "ADD to the contents of the accumulator." During the load routine, instructions may be loaded into addresses 00000 to 20000. The remaining memory registers are filled with information words (data).

Now assume that memory address 00748 contains the instruction 13430155. This is interpreted by the computer as "ADD to the contents of the accumulator the contents of memory address 30155." In this case, 134 is the operation code and 30155 is the address of the operand.

Some computers operate with word lengths which vary from word to word and are called variable word length computers. Others use fixed word lengths wherein each word contains a fixed number of digits.

The use of either of the two types of words has certain advantages and the choice of one over the other is a design feature. The fixed word length computer does not always lend itself to an economic conservation of memory registers since words which are normally shorter than the fixed limit must be filled with the spaces necessary to complete the word. Several short words could therefore waste a large number of memory spaces. On the other hand, fixed word lengths which are compatible with the number of bits that can be stored in a register, facilitate orderly storage and simplify the instructions necessary to store-in or extract-from the memory section.

In the variable word length computer, one block of data may be stored in one, two, or more registers (depending on the length of the word). Thus, variable word-length computers make possible far more compact storage of information in memory by filling up all spaces (except for word separators).

When this method is used, the stored information is not referenced by location alone, but by its position relative to other information in the program. Instruction words must therefore be made more complex in order to locate the correct information to be used in a given operation.

SINGLE AND MULTIPLE ADDRESS COMPUTERS

Although the number of instructions necessary to solve a given problem can be minimized by using the multiple address computer, instructions which reference a single address are much simpler. Generally a single-address computer is one in which each instruction word references a single address in memory. This type of computer begins its operation at the first or some specified instruction in the program and continues by taking instruction words one at a time from memory in sequence unless interrupted by a stop or jump instruction.

For example, ADD THE CONTENTS OF ADDRESS A TO THE ACCUMULATOR, is a single-address instruction because only one memory address is referenced (address A). A variation of the single-address instruction is a "replace ADD" instruction, which may read "ADD THE CONTENTS OF ADDRESS A TO THE ACCUMULATOR AND REPLACE IN A." Again only one address is referenced.

Another variation of the single-address method where the word length of the registers permits, is to place two instruction words in a single memory address. In this case, two instructions will be performed each time a word is read. This method is illustrated in table 4-4.

Table 4-4.—Two-Instruction Words Using Single-Address Operation.

First instruction word		Second instruction word	
OPERATION	ADDRESS	OPERATION	ADDRESS

The left instruction is usually executed first, followed by the execution of the right instruction. Where this method can be used, it increases the operating speed of the computer by making fewer references to memory and also provides a far more expedient use of memory spaces.

A two-address computer word is shown in table 4-5. As the name implies, the word consists of a single operation code and the two addresses (A and B) which are included in the operations. An instruction for this type of machine may read ADD THE CONTENTS OF ADDRESS A TO THE CONTENTS OF ADDRESS B. Here, two addresses are referenced by a single instruction. The "replace ADD" instruction may also be used such as, ADD THE CONTENTS OF ADDRESS A TO THE CONTENTS OF ADDRESS B AND REPLACE IN A.

Table 4-5.—Two-Address Computer Word.

OPERATION CODE	ADDRESS A	ADDRESS B

The two-address instruction word is sometimes used as illustrated in table 4-6. Because the instruction specifies only one address involved in the computer operations, this word is sometimes called a one and one-half address word to distinguish it from the conventional two-address word.

Table 4-6.—Variation of Two-Address Instruction Word.

OPERATION CODE	ADDRESS A	ADDRESS OF NEXT INSTRUCTION

The three-address instruction word (table 4-7) generally contains the operation code, two addresses (A and B) and an address C for storing the results of the operation performed on the information from the A and B addresses.

Table 4-7.—Three-Address Instruction Word.

OPERATION CODE	ADDRESS A	ADDRESS B	ADDRESS C (for storing the result of A modified by B).

The four-address word (table 4-8) contains the operation code, two data addresses (A and B), address C for storage, and the fourth address which is the address of the next instruction to be performed.

Table 4-8.—Four Address Word.

OPERATION CODE	ADDRESS A	ADDRESS B	ADDRESS C FOR STORAGE OF RESULTS	ADDRESS D NEXT INSTRUCTION

The number of addresses in each instruction word is a design feature. Words containing more than four addresses do not seem practicable at this time.

CONTROL CIRCUITS

To be able to gain a thorough understanding of control operations, it is first necessary to study and understand the functional operation of such circuits as OR, AND, NOR, NAND, counters, flip-flops, decoders, and inverters which are used extensively in the control unit as well as throughout the computer. Each of these circuits uses semiconductor or solid-state components such as the silicon diode or transistor. The operation of solid-state devices is treated in the Basic Electronics training course, NavPers 10087 (revised).

The OR, AND, NOT (inverter) NOR, and NAND circuits are introduced in the discussion of Boolean algebra in chapter 3 of this course and are not repeated here. An understanding of the material presented is important to an understanding of the circuits which follow.

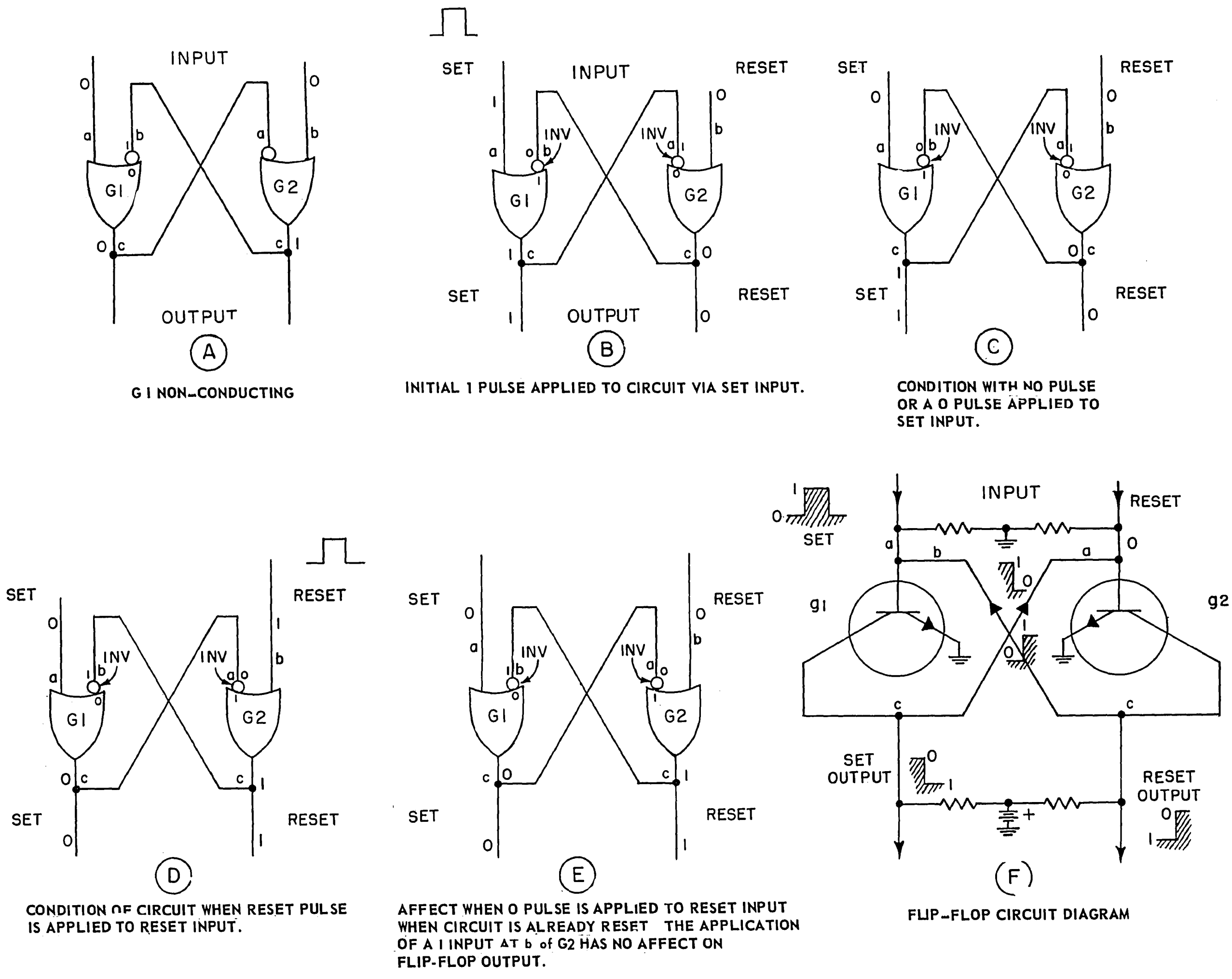
FLIP-FLOP

The basic flip-flop consists of OR circuits and inverters as shown in figure 4-1A thru F. The circuit has two inputs called the SET and RESET inputs and two outputs similarly referred to as the SET and RESET outputs. (The word "RESET" may be used interchangeably with "Clear".) The circuit is a form of multivibrator and thus adheres to the principle that one side is conducting while the other side is cut off.

For simplicity we will refer to the left OR circuit as G1 and the right OR circuit as G2. We will assume that G1 (fig. 4-1A) is nonconducting (and thus has both inputs in the "0" state) until the arrival of the pulse at the "a" input (fig. 4-1B). The 1 input at "a" of G1 produces a 1 output at "c" and at the set output. The 1 output at "c" is coupled to the inverter at the input of G2 where it is converted to 0 at "a." The 0 input at both OR inputs of G2 produce an 0 output at "c" and a 0 reset output. The 0 output is inverted and reproduced at "b" of G1 as a 1 input. This output sustains the flip-flop in the 1 state after the input pulse is removed.

Note that each side of the flip-flop is now producing a logical output: G1 produces a logical 1 and G2 a logical 0. The circuit will hold this position indefinitely, and is thus capable of storing binary digits.

The circuit at figure 41-C shows that no change takes place in the binary output when the 1 pulse is removed from the set input. Because the input at "b" of G1 is 1 the set output remains


INPUT
OUTPUT
G1
G2
A
G I NON-CONDUCTING
SET
RESET
INV
B
INITIAL 1 PULSE APPLIED TO CIRCUIT VIA SET INPUT.
C
CONDITION WITH NO PULSE OR A 0 PULSE APPLIED TO SET INPUT.
D
CONDITION OF CIRCUIT WHEN RESET PULSE IS APPLIED TO RESET INPUT.
E
AFFECT WHEN 0 PULSE IS APPLIED TO RESET INPUT WHEN CIRCUIT IS ALREADY RESET THE APPLICATION OF A 1 INPUT AT b of G2 HAS NO AFFECT ON FLIP-FLOP OUTPUT.
g1
g2
SET OUTPUT
RESET OUTPUT
F
FLIP-FLOP CIRCUIT DIAGRAM

124.25

Figure 4-1.—Basic flip-flop.

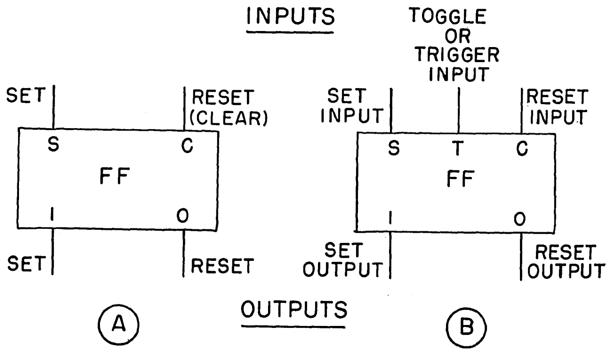
in the 1 state. Neither a 1 nor 0 input at "a" of G1 will affect the output.

Now assume that a 1 input is applied to the reset input as shown at "b" of G2 in figure 4-1D. This condition causes a 1 output at "c" and at the reset output. The 1 output at "c" is inverted and reproduced at the "b" input of G1 as a 0. This action changes the G1 output to the 0 state. Simultaneously, the 0 output at "c" of G1 is inverted to 1 and applied to the "a" input of G2. This action does not change the reset output since it is already in the 1 state.

If the reset input is removed the condition of the flip-flop is unchanged at the output of G2. This condition is illustrated in figure 4-1E. The condition of the flip-flop will be altered only when a 1 pulse is applied to the set input of G1. Figure 4-1F is a circuit diagram of a conventional flip-flop using NPN transistors.

One flip-flop symbol is a rectangle (fig. 4-2A) with terminals extending from opposite sides to indicate set and reset input terminals and set and reset output terminals. The OR circuits and inverter contained in the flip-flop are not shown in the symbol, although an understanding of the operation of these circuits in the flip-flop is essential in determining the effect the various types of input pulses will have on the output.

A second flip-flop symbol (fig. 4-2B) uses a toggle (T) or trigger input terminal. An input signal on this terminal is applied to both halves



124.26
Figure 4-2.—Flip-flop symbol.

of the flip-flop and causes both sides to change their conducting condition, i.e., from 0 to 1 or vice versa. Stated another way, a signal on the "T" input terminal produces the 1's complement of the binary number held in the flip-flop.

COUNT-BY-2 CIRCUIT

The circuit shown in figure 4-3 can be used to count every other pulse, or, count by 2. In the discussion which follows, a set output from the flip-flop indicates a count. The circuit is basically a flip-flop to which has been added two AND circuits, D1 and D2, and associated delay circuits. The delay circuits introduce a controlled time lapse between their input and output pulses to ensure positive triggering and eliminate interference. Delay circuits are explained in detail later.

Consider the circuit at A, with a 1 input at "b" of D1 and at the reset (R) output of the flip-flop prior to the application of the input pulse at t_2, (see waveform). The circuit will remain in this condition from t_0 to t_2. At t_2 the input pulse will simultaneously appear at "a" of D1 and "b" of D2 (fig 4-3B). The input pulse at "a" of D1 along with the 1 pulse state previously stored at "b" of D1 will produce a 1 pulse output at "c" of D1 which enters the delay line. The set input pulse to the flip-flop will be delayed for a period greater than the period t_2 to t_3. Thus, when an output pulse does appear at the set output of the flip-flop (and at the "a" input of D2) the trailing edge of the input pulse (at t_3) will have returned to 0 and the D2 circuit will not produce an output. This condition is illustrated in figure 4-3C. The circuit has now counted 1 pulse and will remain in the condition shown in figure 4-3C for the period from slightly after t_3 to t_5. Although the circuit is triggered into action at t_5, the delay lines prevent a set output return to zero until slightly after t_6 as explained below.

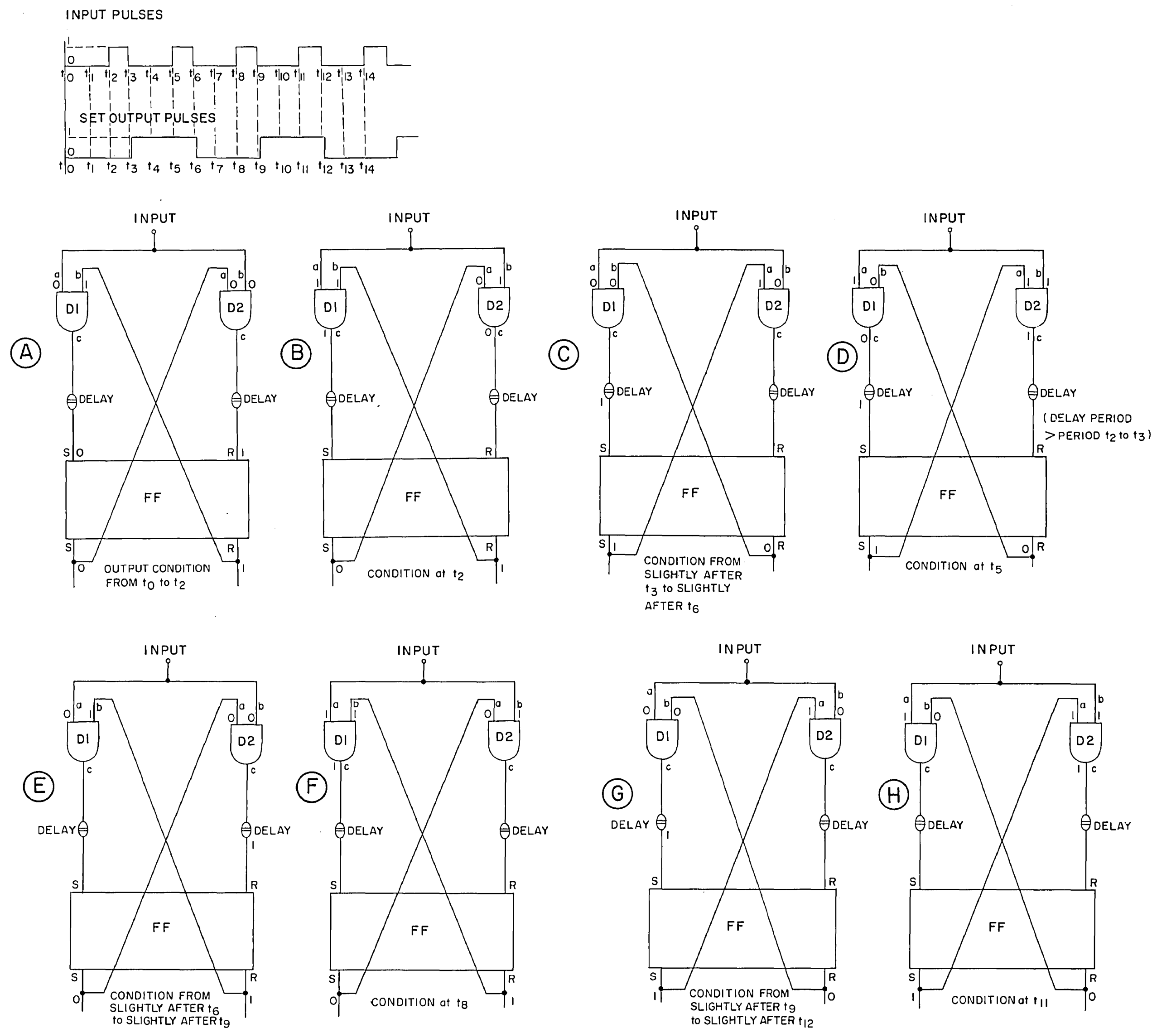
At t_5, a 1 input appears at "a" of D1 and "b" of D2 (fig. 4-3D). The simultaneous application of pulses at "a" and "b" of D2 produce an output at "c". This pulse is applied to the delay line. At the trailing edge of the pulse-period (t_6), the voltage at "a" of D1 and "b" of D2 returns to 0. The output from "c" of D2, which is fed to the delay line, appears, (after the delay period) as a reset input to the flip-flop. The reset input also causes the flip-flop to switch its condition so that the reset output goes to 1 and the set output goes to 0 (fig 4-3E). The reset output is applied to "b" of D1. No output appears at "c" of D1 since the "a" input is 0. The circuit remains in the state shown in figure 4-3E until the arrival of the leading edge of the pulse at t_8.

At t_8, the condition shown in figure 4-3F prevails. The inputs at "a" and "b" of D1 produce an output at "c", which after a delay greater than the period from t_8 to t_9, appears at the set output of the flip-flop (fig. 4-3G), and at the "a" input of D2. The set output remains in this condition from slightly after t_9 to slightly after t_{12}. The leading edge of the pulse at t_{11} (fig 4-3H) produces an output at "c" of D2. After a delay greater than the period from t_{11} to t_{12}, the input from "c" of D2 produces a reset output from the flip-flop (fig 4-3A). This action causes the reset output to change to the 1 condition and the set output to assume the 0 condition. The circuit, having now returned to the condition illustrated in figure 4-3A, repeats this action for every four pulses received at the input. The 1 output at the set terminal indicates a count for alternate pulse inputs and the circuit is appropriately called a "count-by-2" circuit. This circuit is used extensively in computer applications.

FLIP-FLOP COUNTER

Several count-by-2 circuits can be arranged in cascade so that they can count any number of pulses. Such an arrangement is shown in figure 4-4. The basic flip-flop symbol (discussed earlier) is altered to represent the count-by-2 circuit as indicated by the single input terminal.

Both the set and reset outputs are used in the count process. The set output indicates the


INPUT PULSES
SET OUTPUT PULSES
INPUT
D1
D2
DELAY
FF
OUTPUT CONDITION FROM t0 to t2
CONDITION at t2
CONDITION FROM SLIGHTLY AFTER t3 to SLIGHTLY AFTER t6
CONDITION at t5
(DELAY PERIOD > PERIOD t2 to t3)
CONDITION FROM SLIGHTLY AFTER t6 to SLIGHTLY AFTER t9
CONDITION at t8
CONDITION FROM SLIGHTLY AFTER t9 to SLIGHTLY AFTER t12
CONDITION at t11

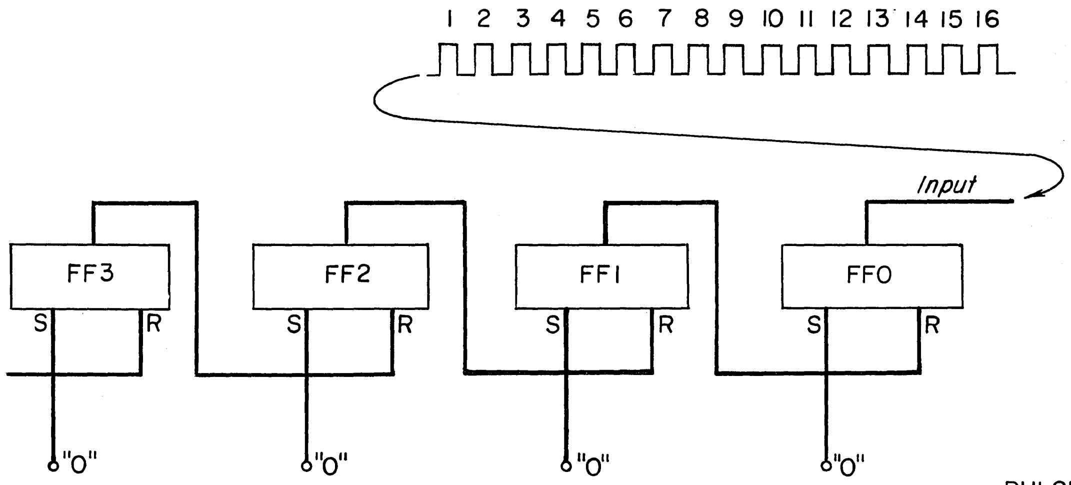
124.27

Figure 4-3.—Count-by-2 circuit.

count, i.e., 0 or 1 at a particular level (2^3, 2^2, 2^1, or 2^0). The weight of a digit increases in value from right to left so that if a count of 1111 exists, the binary 1 at the set output of FF3 represents the most significant digit. A 1 count at the output of FF2 will indicate the second most significant digit; at FF1 the next significant digit, and so on.

Now consider the actions in the count circuit. We will first assume all flip-flops are in the reset condition and the output is 0000 as shown at the set terminals.

You can make a mechanical training aid of figure 4-4 with some paper clips (short straight ones). Open the book to form a flat horizontal surface for the figure. Use four of the paper clips as switch blades of single pole double-throw switches pivoted at top center of the flip-flops. At the start all switch blades are in the reset (R) position (top center to lower right).



OUTPUT of FF3	OUTPUT of FF2	OUTPUT of FF1	OUTPUT of FF0	PULSE COUNT
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15
0	0	0	0	0

124.28

Figure 4-4.—Flip-flop counter.

Place a fifth paper clip at the top of the pulse count column to keep track of the decimal count.

Before starting the count remember a flip-flop will produce an output to the next flip-flop to the left, only when going from set (S) to reset (R). When the switch blade is on R, the R output is 1 and S output is 0. When the switch blade is on S the S output is 1 and the R output is 0.

Start the count by moving the pulse count clip to decimal 1. This indicates a pulse to flip-flop 0 (FF0), causing FF0 to flip from R to S. This action produces a set output of 1 as indicated in the first row of the table, under the FF0 column.

The reset output of FF0 is 0 and the initial pulse does not get through to FF1. The other three flip-flops remain in the 0 state and the count of decimal 1 in binary form is 0001. Move the pulse count to decimal 2. This pulse causes FF0 to change state from S to R. Move the FF0 switch blade from S to R. Now the R output is 1 and the S output is 0. This action causes FF1 to flip from R to S and the FF1 output is 1 (second row of column FF1). The R output of FF1 is 0 and the outputs of FF2 and

FF3 remain 0. The count of decimal 2 in binary form is 0010.

Move the pulse count to 3. FF0 flips from R to S and the count is 0011 (no output to FF1).

At the count of 4 FF0 flips from S to R. This action flips FF1 from S to R which causes FF2 to flip from R to S (no pulse to FF3). The count is now 0100 (fourth row of the table).

In this way you can go through 15 counts to produce 1111. On the next (16th count) all flip-flops revert to 0.

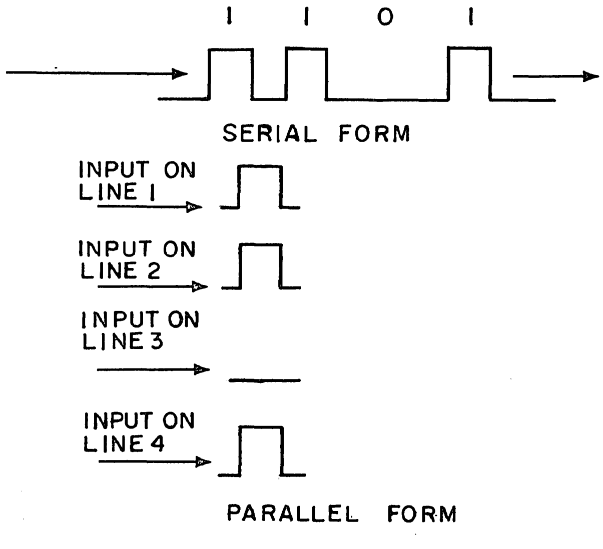
Any number of these circuits can be connected in cascade to produce a particular count. Neon lamps (or other indicating devices) can be connected in the set output circuits so that they light when a given flip-flop is in the 1 condition. When used, the lamps are displayed on the computer console to give the operator an indication of a particular internal count.

SERIAL AND PARALLEL OPERATION

Computers may process information pulses in either serial or parallel form. In serial form the pulses occur as a timed series, one pulse at a time (fig. 4-5). In parallel operations each pulse occurs simultaneously and is transferred within the computer on separate transmission lines. Serial and parallel data are explained in detail in a later chapter.

The counter circuit shown in figure 4-4 is versatile in that it may perform several computer operations. For example, if we consider a series of pulses at the input to the first counter, say 4 pulses, the counter will advance to a state where the set outputs will represent 0100, which may represent a computer word. Because the counter is a static device and will remain in this condition unless subsequent pulses are applied, the circuit remembers or stores this count. In this respect, a flip-flop counter may be used as a REGISTER, since it stores a computer word.

A second application of this circuit is seen when we assume that 5 additional pulses are applied to the counter input. The count now advances to 1001 (9) which is the sum of the two input pulse series. This process is called serial addition. This circuit arrangement thus acts as a basic accumulator.



124.29
Figure 4-5.—Serial and parallel coded pulses.

PARALLEL ADDER

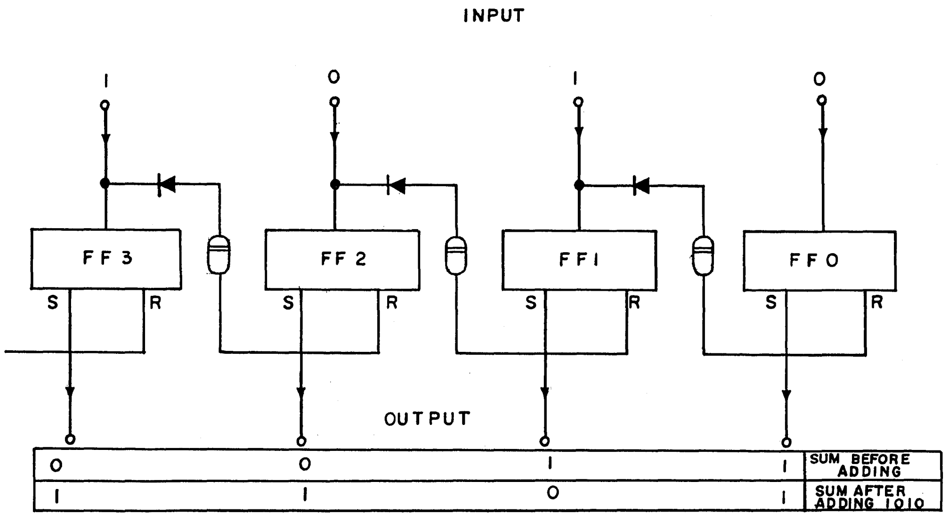
The circuit shown in figure 4-6 is used to illustrate the basic example of parallel addition. Note that 4 binary digits (bits) are received in parallel, one at the input of each flip-flop. If we assume that the number previously stored in the counter is 0011 (3, as shown in the first column below the set terminals) and the parallel input pulses are 1010 (10), the counter will add these pulses. In the following explanation of the addition process it is assumed that the set position of the flip-flop, as well as a significant pulse, is represented by a 1 condition. Conversely, a reset output, as well as a no significant pulse input, represents 0. Further, we will assume a flip-flop of the type shown in figure 4-3 to be the basic unit.

A delay line is connected between successive flip-flops. The output of each flip-flop is used to provide a significant pulse to the delay circuit on its left only when the set output of that flip-flop changes from 1 to 0 and the reset output changes from 0 to 1.

A 1 input, whether at the parallel input or from the associated delay line, causes a flip-flop to change its state. Thus the flip-flop will change whenever it receives a 1 pulse from any source.

Because the original condition of the counter is 0011, the following changes will occur when it receives 1010 at its parallel inputs:

1. FF2 and FF0 receive no significant pulses from the parallel input and remain unchanged for the present.
2. FF3 and FF1 do receive 1 pulse from the parallel input and both change states. FF3 now reads 1 and FF1 now reads 0.



124.30

Figure 4-6.—Parallel adder.

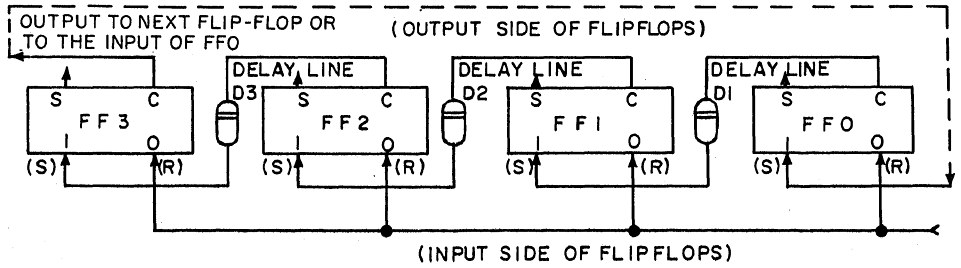
3. Of the four flip-flops only FF1 has changes from 1 to 0. Therefore, it is the only flip-flop to send a 1 pulse (carry) through its delay line to FF2.
4. This pulse changes FF2 from 0 to 1. The adder now reads 1101 (13), which is the correct sum of 0011 (3) and 1010 (10).

Note that the diodes prevent pulses from the parallel input (assumed to be positive-going in this case) from interfering with the adjoining flip-flops, while the delay lines prevent the pulses from being fed to the adder and any carry pulses from appearing at any flip-flop input simultaneously. In a sense, then, the delay units are used to perform the carry operation.

RING COUNTER

Several flip-flops can be connected to form a ring counter (fig. 4-7); the name is derived from the fact that the output of the last flip-flop (FF3) is sometimes connected back to the input of the first (FF0). This is not a requirement, however, as other means can be provided to initiate the action in the first circuit at the proper time.

In most applications of this circuit, only one of the flip-flops is in the "on" (set) condition at a given time. An advance input pulse, which is applied to all flip-flops simultaneously, causes the "on" flip-flop to change its conducting state, and, in turn, transmit an input pulse to the next


OUTPUT TO NEXT FLIP-FLOP OR TO THE INPUT OF FFO
(OUTPUT SIDE OF FLIPFLOPS)
DELAY LINE D3
DELAY LINE D2
DELAY LINE D1
FF 3
FF 2
FF 1
FF 0
S C
1 0
(S) (R)
(INPUT SIDE OF FLIPFLOPS)

124.31

Figure 4-7.—Ring counter.

flip-flop to the left. The pulse output from the affected flip-flop is delayed until after the trailing edge of the advance input pulse has dropped to zero to prevent double-triggering. However, after the delay period, the input pulse to the next flip-flop changes the conducting state in that circuit from "off" to "on".

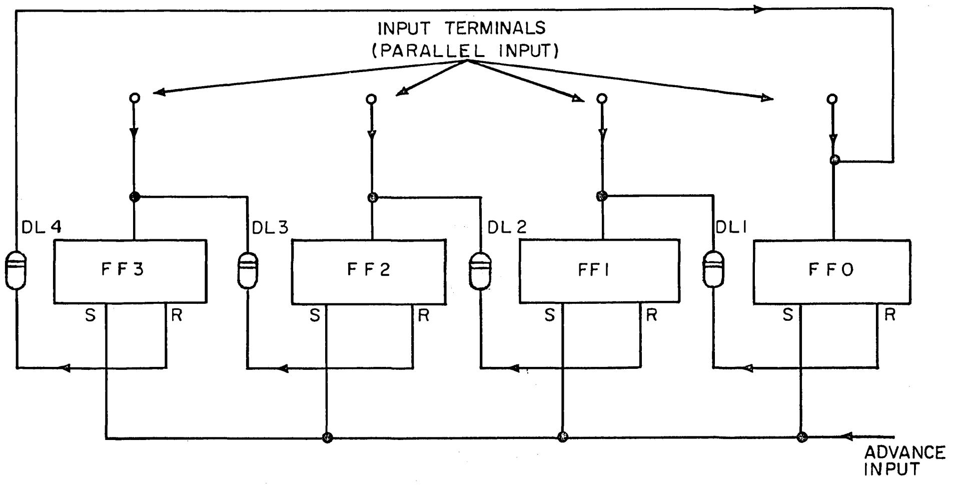
The next advance pulse will cause the "on" condition to be established in the next flip-flop. This action continues until the "on" condition has advanced from FF0 to FF3. A subsequent advance pulse will cause the "on" condition to be transferred either to FF0 or to the next flip-flop in the ring.

The circuit in figure 4-8 is essentially a ring counter to which input paths are added for the purpose of reading-in (in parallel) the desired binary digits. As stated before, the four flip-flops make up a register. Because the action within the ring counter is to shift one digit one place (either left or right but left in this case), an advance input pulse applied simultaneously to each flip-flop in the register, will cause the bit previously stored in a stage to be shifted to the next higher order flip-flop. The bit previously stored in FF0 will be shifted to FF1; that previously stored in FF1 will be shifted to FF2; the FF2 bit will move to FF3; and finally the bit previously in FF3 will be transferred to FF0. Thus, if 1011 has been stored before the arrival of the advance pulse, the register now contains 0111. The advance pulse flips FF3, FF1, and FF0 from S to R. Slightly later a pulse from delay line 1 flips FF1 from R to S. Delay line 2 transmits a pulse that flips FF2 from R to S. Delay line 3 does not transmit a pulse, and FF3 remains in the reset condition. Delay line 4 transmits a pulse that flips FF0 from R to S. Thus, the 0111 condition is stored. This type register is called a "shift register". Left-shift registers are used extensively in computers to perform multiplication (as will be shown in the next chapter). Right-shift registers are used in the division process.

The ways in which flip-flops can be used to perform computer operations are too numerous to consider here. The reader must therefore study flip-flops and the application of these circuits as presented in this discussion until he is certain of their operation. Only with a thorough understanding will he be able to comprehend other applications of this circuit which are not so elaborately explained.

DELAY LINES

It has been stated that one of the basic functions of the control unit is to time all operations within the computer. Further, it was stated that the time to execute an operation depends on the type of operation to be performed. For example, a multiple operation generally required more time than an add operation, a divide operation required more time than a subtract, etc. To perform in this manner, the control unit must contain circuits which will (1) generate the basic timing pulses (say in a stabilized multivibrator



124.32

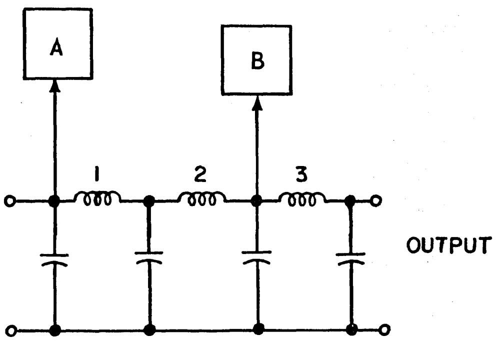
Figure 4-8.—Left shift register.

or crystal oscillator); (2) it must provide pulses to start an operation at a certain time; and (3) it must contain circuits which will count the timing signals and produce a pulse which will terminate a particular operation after a specified time.

The circuit in figure 4-9 (called an artificial transmission line, or delay line) is used to explain one of the basic timing principles. The action is explained in more detail later in this chapter.

If a voltage is applied to the input terminals of the line, a definite amount of time passes (dependent upon the number of LC sections) before the voltage appears at the output terminals. The LC sections thus give the line the ability to delay the output voltage.

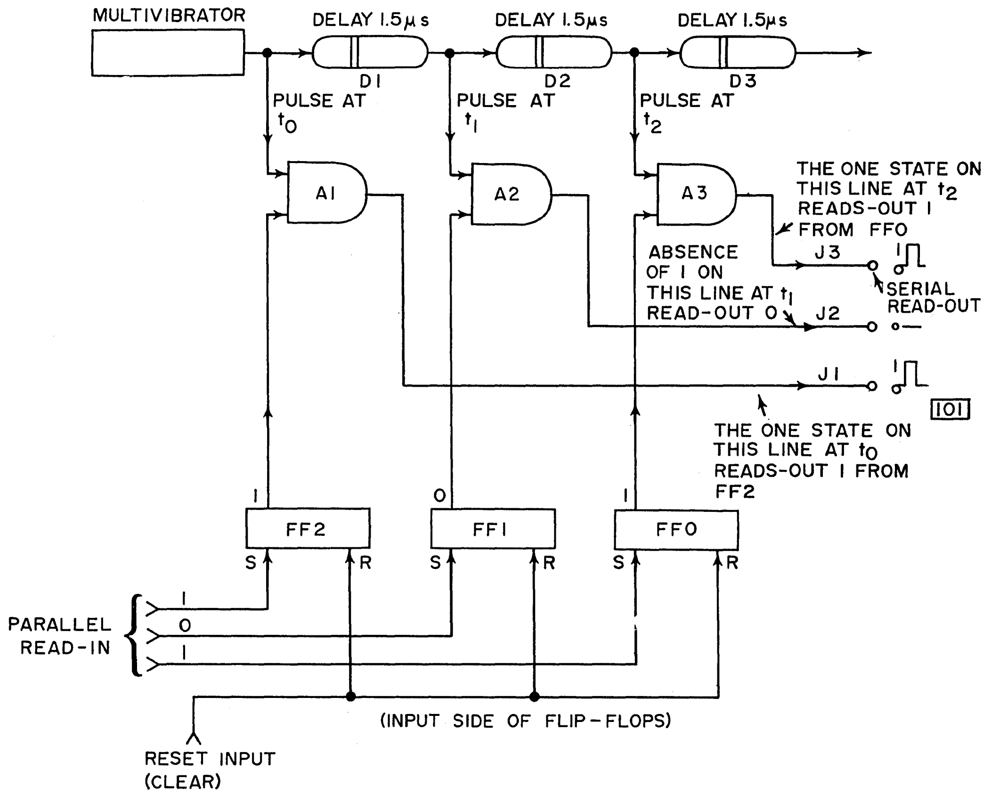
Assume that a voltage must be applied to the circuit in block B one or two microseconds after it has been applied to block A. This condition can be satisfied merely by constructing the 1 and 2 sections of the line for the desired delay.



124.33

Figure 4-9.—Delay line.

To avoid the bulkiness of an actual transmission line, an artificial line may be built of coils and capacitors. Such lines have approximately the same characteristics as actual lines but occupy a smaller space. This is the usual method of constructing delay lines.


MULTIVIBRATOR
DELAY 1.5μs
DELAY 1.5μs
DELAY 1.5μs
D1
D2
D3
PULSE AT t0
PULSE AT t1
PULSE AT t2
A1
A2
A3
THE ONE STATE ON THIS LINE AT t2 READS-OUT 1 FROM FF0
ABSENCE OF 1 ON THIS LINE AT t1 READ-OUT 0
J3
J2
J1
SERIAL READ-OUT
101
THE ONE STATE ON THIS LINE AT t0 READS-OUT 1 FROM FF2
1
0
1
FF2
FF1
FF0
S
R
PARALLEL READ-IN
(INPUT SIDE OF FLIP-FLOPS)
RESET INPUT (CLEAR)

124.34

Figure 4-10.—One method of using delay lines for serial read out.

Now suppose it is necessary to read into a register containing FF2, FF1 and FF0 (fig 4-10) in parallel form and to read the information out (at 1.5 μs intervals) in serial form.

One method of performing this operation is described below. Initially all flip-flops are cleared or reset by an input pulse on the reset line. The information to be stored (in the form of 1's and 0's) is fed over parallel lines to the appropriate flip-flops. A 1 input to any flip-flop will produce a 1 set output. In the following discussion the set output is taken to represent the state of the flip-flop.

Assume that the multivibrator feeds a single pulse to delay line, D1, and to the AND circuit, A1. Because the set output of FF2 is in the 1 state, the coincidence of the inputs to the AND element (A1) causes this circuit to produce a 1 output during the period t_0. This pulse is read out at J_1.

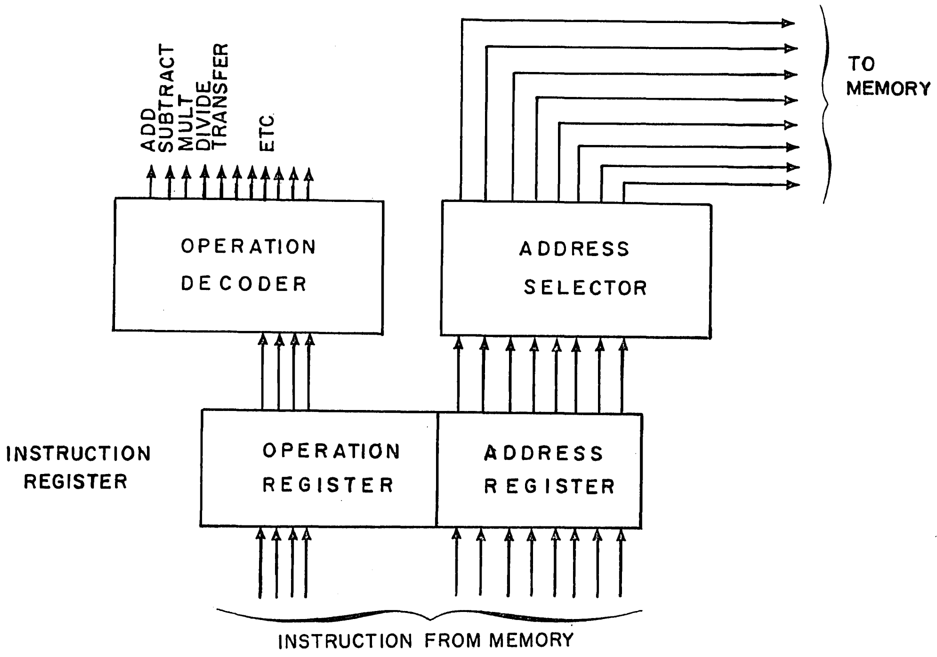
The pulse applied to D1 at t_0 emerges from the delay line at t_1 and is applied simultaneously to D2 and A2. The presence of this pulse at one input of A2 will not cause the AND element to produce an output (the set output of FF1 to A2 is 0) and the read-out from J_2 at time t_1 is 0. The pulse delivered from D2 at time t_2 is applied simultaneously to D3 and A3. Because the set output of FF0 is in the 1 state, the A3 AND circuit will produce an output to J_3 at time t_2. Thus, the serial read-out of the register is accomplished at the specified intervals, as 101.

CONTROL OPERATIONS

The two basic functions within the control sections are (1) to obtain instruction from memory, and (2) to execute these instructions. The control function performs these actions in two cycles-first fetch, then execute. The fetch cycle is performed under the direct influence of the stored program so that the instructions are read in a fashion which will lead to the correct solution of the problem.

INSTRUCTION REGISTER

Each instruction read from memory is fed to an INSTRUCTION REGISTER (fig. 4-11). This register holds the instruction throughout the execution cycle.



124.35

Figure 4-11.—Instruction register.

Operation Register

The instruction register is divided into two smaller registers; the OPERATION REGISTER, which receives and holds the operation code part of the word, and the ADDRESS REGISTER which receives the address of the operand. The operation code is fed to a decoder matrix or operation decoder which decodes the operation and produces a static output on a particular line depending on the type of operation requested. This output is held constant during the entire execution cycle and is generally applied to any number of switching elements (OR's, AND's, FLIP-FLOP's, ADDERs, etc.) throughout the computer. All OR circuits which receive this static input will produce an output, which, in turn, will cause some particular action as necessary to execute the instruction. AND circuits receive this static voltage as a single 1 input but will remain inactive to perform their given steps in the execution of the operation until a clock pulse or some other form of timing pulse is received at the second input. The decoder output is fed to all of the circuits in the computer as necessary to execute the particular operation.

Address Register

The address register, after receiving the address of the operand, enables an address selector, which, in turn, locates the operand and transfers it to an intermediate register (shown later). The intermediate register is used as a form of buffer for reading information into or extracting information from the memory section.

In a stored-program computer, instructions are read and executed sequentially unless altered by conditions derived while executing an instruction, or, unless directed to transfer unconditionally. Normally, an instruction counter reads the instructions into the instruction register in the programmed sequence, starting at 0 and progressing in numerical order, 1, 2, 3, 4, etc. During the execution of each instruction, the instruction counter is advanced by 1. At the end of the execution cycle, switching circuits in the control unit initiate a command "READ NEXT INSTRUCTION"; whereupon the instruction cycle is again started by an enabling signal which is sent to the instruction counter causing the next sequential instruction to be read into the instruction register. Many other operations (not pointed out at this time) are accomplished in preparation for the next instruction.

CONDITIONAL AND UNCONDITIONAL TRANSFERS

If conditions are derived which necessitate a jump or conditional transfer to some step in the program other than the next sequential step, certain signals will be generated and fed to switching elements in the control unit to cause the necessary jump to be executed.

The unconditional transfer, as the term implies, is accomplished without regard to prevailing conditions. It is initiated by an instruction which may state in effect, GO FROM PRESENT ADDRESS (say address 00030) to address 05275.

To illustrate the need for unconditional transfers, consider the following example. It is understood by this time that instructions and data are necessarily stored separately in memory. Normally then, a certain number of addresses (say the lower numbered addresses) are reserved for and contain instructions. We will assume that the data addresses begin at the next higher address immediately after the instructions, but that several of the highest numbered addresses are not used. Now if all instructions and data are loaded accordingly and it is later discovered that more instructions must be loaded in order to solve the original problem or to solve another phase of the problem we would logically decide to store the additional instructions in the vacant address rather than extract all data, load the additional instructions, and reload the data. In order to use these added instructions, we must instruct the computer control to jump to the address of the first added instruction "unconditionally."

SYNCHRONOUS CONTROL METHOD

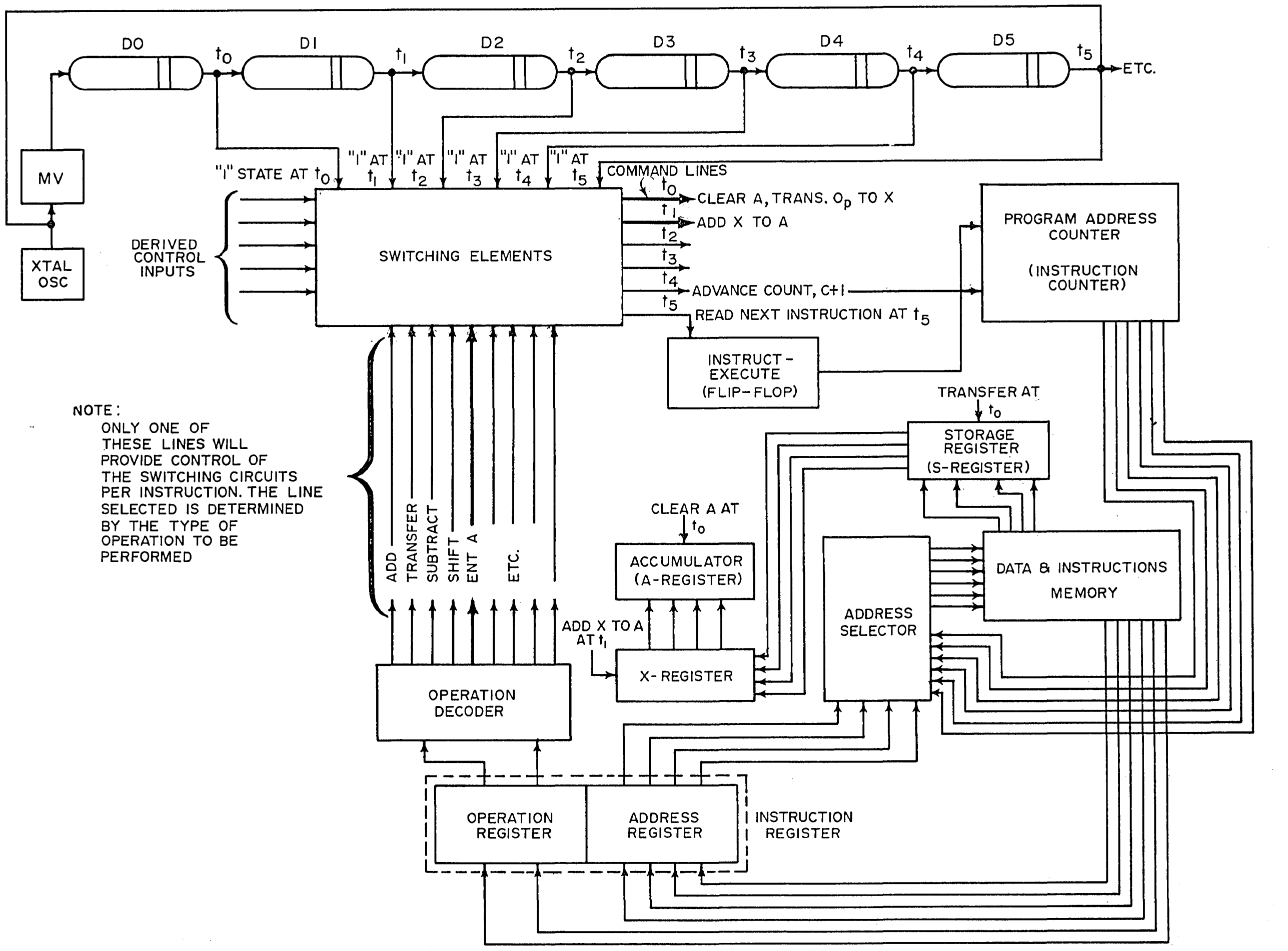
There are many ways presently being used in control units to execute instructions. Every computer seems to have some unique control feature peculiar to that particular machine. There is no one so called "best method" since the control method cannot, in most cases, be decided without considering several other factors; such as storage access time, input-output devices, and the nature and time required to perform arithmetic operations.

Synchronous and asynchronous methods of control are explained in the remaining part of this chapter. Synchronous control is a mode of computer operation characterized by a fixed time period for the execution of each operation. Conversely, asynchronous control uses varying amounts of time to execute its operations, depending, of course on the type of operation being performed. In the asynchronous control method the advance to the next command is signaled when the execution of the previous command has been completed.

The myriad operations performed in even the simplest control unit would be too great to explain in every detail. This is complicated by time considerations, i.e., showing all of the control signals which are fed throughout the computer during many small timed intervals. Thus, the control methods discussed below are hypothetical and are intended to show basically how the control unit orders and executes the major operations in the computer.

Consider the circuit in figure 4-12. A multivibrator receives trigger pulses from a crystal oscillator and provides input pulses to a delay line consisting of sections D0 through D5. The number of sections is arbitrarily chosen. The time required for a pulse to traverse the line is slightly less than the period of the free-running multivibrator so that the output pulse at t5 will arrive at the multivibrator input at the time this circuit is about to produce its next pulse. This action synchronizes the multivibrator and keeps the multivibrator-delay line arrangement compatible.

Output pulses from the line, taken at t0, t1, t2, t3, t4, and t5 are applied during their respective intervals to various switching elements



124.36

Figure 4-12.—Basic synchronous control operation.

throughout the computer. These circuits, (OR's, AND's, flip-flops, etc.) are too numerous to show in detail and are represented here by a single block from which a few of the outputs are shown on "command lines." The timed inputs, i.e., the pulses from the delay line at t0, t1, t2, etc., tell these circuits "when" to perform a given operation.

Now consider another aspect of control. As stated earlier, the instruction counter, or program address counter, as it is sometimes called, receives control pulses which are sent via the address selector to memory, thereby causing a particular instruction to be read from the memory into the instruction register. Each instruction word, as you recall, consists basically of the operation code and the address of the operand. In reading the instruction from memory, the operation code is fed to the operation register while the operand address is read to the address register.

Accordingly, the operation register feeds the operation code to the operation decoder, which, in turn, decodes the operation (as stated earlier) and supplies a static voltage output on a single decoder output line, the one selected, of course, being determined by the type of operation to be performed. This static voltage is fed to those switching circuits which are to perform some action in the execution of the instruction. In a sense, it is this static voltage which tells the computer circuits "what" to do. Knowing when to do each step, as dictated by the delay line inputs, and what to do, as dictated by the operation decoder output, the computer proceeds to execute the instruction.

The address register, immediately upon reception of its input, energizes the address selector circuits to cause the operand to be read into the S-register. The operand remains in this register until the instruction is executed or until it is transferred to some other register.

Now, consider the action in more detail. Given the following instruction "ENTER THE OPERAND IN THE A REGISTER", we will follow the major steps in its execution.

The instruction is read into the instruction register from memory under the influence of the instruction counter. The operation decoder decodes the operation code and activates the "ENT A" line. This line, in turn, is fed to all of the switching circuits involved in the execution.

The address selector, by instruction from the address register, fetches the operand from the section of memory which contains data and stores it in the S-register.

The arrival of the t0 clock pulse to the switching circuits causes a command line to be enabled at t0, and the following part of the operation is executed: CLEAR THE A REGISTER (accumulator), and TRANSFER THE OPERAND TO THE "X" REGISTER.

The accumulator receives data by adding data to its contents. Thus, clearing the accumulator returns the register to 000. Adding a number to the accumulator after clearing is therefore no more than entering that number in the accumulator.

The arrival of the t1 clock pulse to the switching circuits causes the ADD X TO A command line to be enabled. This command adds the operand to the accumulator and the execution is completed.

Note that the execution of this instruction (in this example) does not require the use of the clock pulses from t2 and t3. A more complex instruction requiring a multiply or divide operation may use all clock pulse enable inputs.

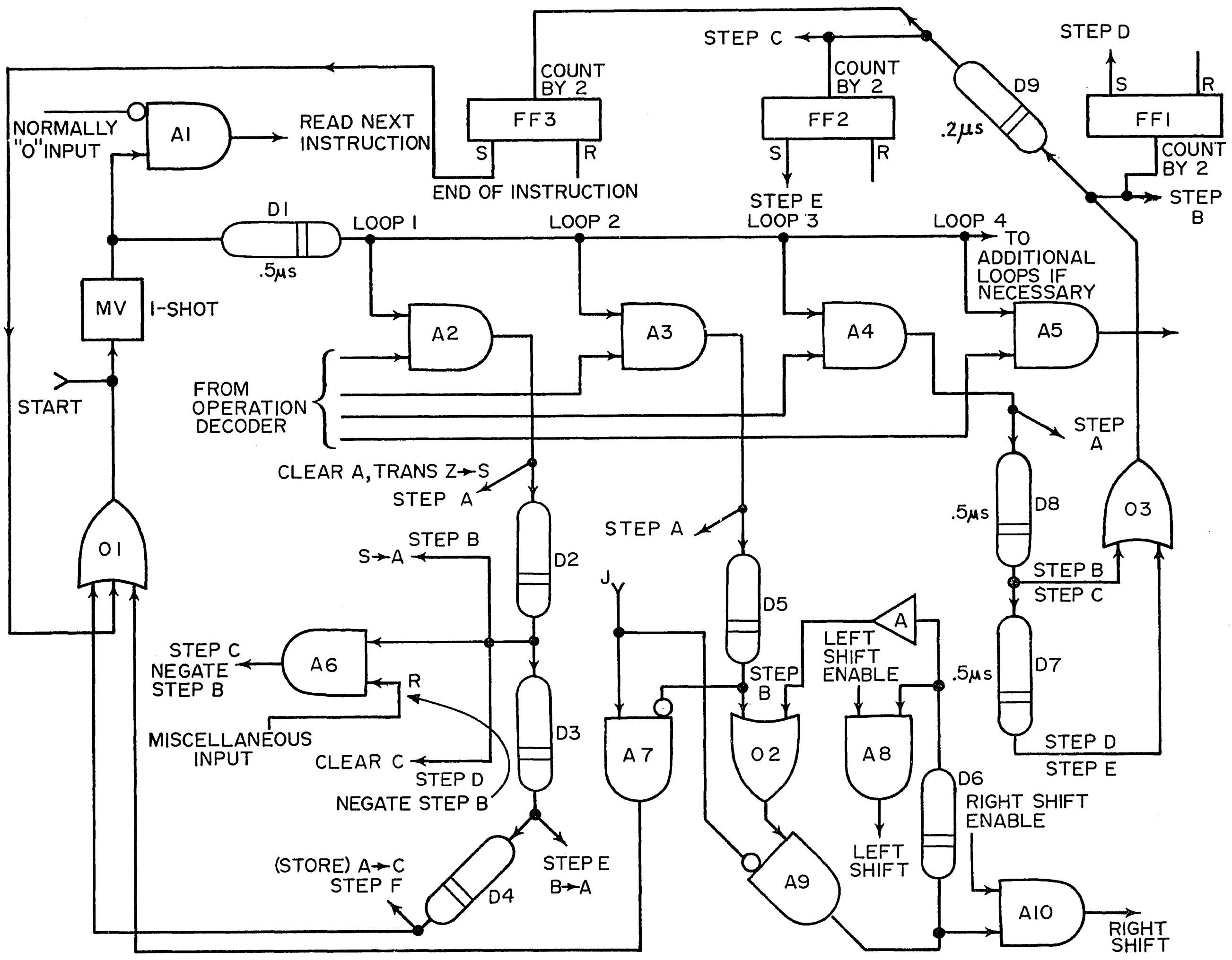
Also note that during the execution cycle the instruction counter receives a signal as initiated during clock pulse period t4. This action normally causes the instruction counter to advance its count by 1 in preparation for reading out the next instruction from memory. The "ENT A" instruction is completed. At t_5, the command to "READ NEXT INSTRUCTION" is initiated.

If a conditional transfer is initiated during the execution of an instruction, certain signals will be derived to cause the conditional jump. These signals are fed to the switching circuits as derived control inputs.

Because all operations are performed in synchronism with the clock pulses, this method of control is described as "synchronous control." Each operation requires a certain number of clock pulses, and consequently the time to complete any one of the various operations is an exact multiple of the clock pulse period.

ASYNCHRONOUS CONTROL

An example of asynchronous control is explained with the aid of figure 4-13. A one shot multivibrator simultaneously feeds a pulse to AND circuit A1 and delay line D1. The pulse at A1 in coincidence with a 1 input from the inverter, causes a 1 output at A1 to initiate READ NEXT INSTRUCTION. The D1 input is



124.37

Figure 4-13.—Asynchronous control.

delayed .5 μs (in this example) to allow sufficient time for the instruction cycle (the time required to read an instruction into the instruction register from memory and performs all other operations in preparation for the new instruction).

Loops 1, 2, 3, etc., are designed respectively to perform a single type of operation, although miscellaneous control inputs to a loop may alter the loop, and thereby permit other very similar operations to be accomplished. Thus, a control unit of this type will contain almost as many loops as the total number of types of operations it can perform.

The pulse from the multivibrator, after delay in D1, is applied simultaneously to one of the inputs of the AND circuit at the input of each loop. The loop selected to execute an instruction will be determined by the operation decoded output. Each decoder output line is tied to a different AND element, or loop, except where two or more similar operations are performed by the same loop; whereupon control of such similar operations will be directed from the decoder to the same loop.

Consider the execution of a three-address instruction as follows: ADD THE CONTENTS OF REGISTER S TO THE ACCUMULATOR: ADD THE CONTENTS OF REGISTER B TO THE ACCUMULATOR: STORE RESULTS IN REGISTER C.

Loop 1 is used to illustrate the timing sequence of operations progressing from steps A thru F. The actual operation is hypothetical and is used here merely to show how certain parts of an instruction are executed at certain times.

Step A begins as soon as a 1 output is produced at A2. This part of the sequence causes the A register to be cleared, and the contents of the Z register (intermediate storage register) to be transferred to the S register.

After the D2 delay, enabling signals are produced (simultaneously) to cause step B (transfer S $\rightarrow$ A), step C (negate step B, if a miscellaneous input is received at the lower input terminal of (A6) and step D (clear C). We will assume that the miscellaneous input to A6 is not received, and step B is executed. Transferring S $\rightarrow$ A (read S to A) is the equivalent of adding the contents of S to A as explained earlier.

After delay in D3, step E is executed and the contents of the B register are added to the accumulator (A).

Finally after delay D4, the contents of A which now represent the sum of operands stored in the S and B registers, are transferred to the C register (Step F).

Now consider the instruction: ADD THE CONTENTS OF B TO THE ACCUMULATOR AND STORE IN C. Note the similarity of this instruction to the one just considered. If step B (S $\rightarrow$ A) is eliminated, loop 1 can again be used and only the contents of the B register will be added to the accumulator. Thus, we assume here that the required miscellaneous input is received immediately after the delay in D2 and step B is negated. With this understanding a further explanation of the execution of this instruction is not necessary.

Note that at the end of the D4 delay, the output from this line is also applied through OR circuit 01 to the multivibrator. The 01 input triggers the multivibrator, and causes an output pulse to be fed to A1. This action causes the next instruction to be read from memory.

A right or left shift operation can be accomplished using loop 2 to produce the required number of pulses to execute the desired shift count. Step A is not involved directly in the shift action, but is shown here as an enabling output to all other circuits involved in the execution of all other phases of this operation, even those which adjust other computer circuits for the up-coming shift.

At step B, OR circuit, 02, passes a pulse into a ring circuit consisting of 02, A9, D6, and the amplifier. The circuit of A9 has an inverter at its input so that if a 1 input pulse is not received at J, the pulse from D5 will continue through 02, A9, the delay line D6, the amplifier, and back to 02. OR circuit, 02, again permits the pulse to re-enter the ring. The amplifier compensates for losses in the pulse amplitude each time around the ring.

Shift outputs are taken from the line during each ring count and applied to one of the inputs of AND circuits A8 and A10 respectively. A left enable or right enable static voltage is applied as a miscellaneous input to either of the two AND elements depending on the type of shift desired.

When the shift has advanced through the desired count, a 1 input pulse is applied at J. This "1" input simultaneously disables A9 (stopping the ring action) and produces an output from A7 through 01 to the multivibrator so that a multivibrator output will be generated and the next instruction will be read.

Because of the inverter at the right input of A7, the circuit is disabled only during the time when a pulse appears at the output of D5. An output pulse cannot be developed by A7 during this period, neither is such an output desired. However, after the trailing edge of the D5 output pulse, the inverter at A7 will again produce a 1 at the right terminal.

Loop 3 is designed to supply control pulses to various computer circuits at timed intervals. The particular operation performed is not important. Rather, the sequence and timing of control pulses as necessary to execute the operation is the significant consideration.

Here, the output pulse at step A causes the execution of the first part of the operation. After the delay in D8, a single pulse is passed through OR circuit, 03, to D9. Step B is executed before the D9 delay and is followed after the delay in D9 (.2 μs in this case) by the execution of step C. Count-by-2 circuits FF1 and FF2 are reset by the step B and step C enable pulses, and therefore do not produce the step D and step E enabling pulses at this time.

Note that the pulse from D8 is also fed into the D7 delay line. The delay in this line is long enough to permit steps B and C, totaling .4 μs, to be executed before the D7 pulse evolves. When the D7 output is developed, OR circuit, 03, produces an output pulse which initiates step D by setting FF1. Setp E will be executed after the D9 delay and the setting of FF2. Enabling pulses for steps B and C will be produced for each output pulse from 03.

In order to maintain the static operation decoder output, it is necessary to hold the instruction being executed in the instruction register throughout the execution cycle. The count-by-2 circuit (FF3) at the D9 output produces an ''end of instruction'' pulse only at its set terminal, thus it prevents the first pulse from D9 (after the execution of step C) from triggering the multivibrator and thereby prevents the start of an instruction cycle until after steps D and E have been executed.

Note that in this method of control the time required to execute an instruction varies, depending on the type of instruction.

CHAPTER 5

ARITHMETIC UNIT

INTRODUCTION

The obvious purpose of the arithmetic unit is to perform arithmetic operations. Though this is true, the operations performed in this unit are not limited to addition, subtraction, multiplication, and division. Other operations such as shifting, complementing, and comparing are also performed.

In order to understand how arithmetic operations are performed, it is necessary to acquire a knowledge of logic equations, diagrams, and symbols. These topics are treated in chapter 3 of this training course. The relationship between logic and mathematics makes possible the use of logic circuitry to perform the operations of arithmetic. When a problem in arithmetic is broken down to its basic operations, it becomes a problem in logic. Consequently, a circuit that is used to perform an arithmetic operation can always be broken down into a specific combination of logic elements.

For a simple illustration of this concept let us consider the arithmetic subtraction of two binary digits. It can be seen that there are three possible subtracting combinations of these digits as follows:

$$\begin{aligned} 0 - 0 &= 0 \\ 1 - 0 &= 1 \\ 1 - 1 &= 0 \end{aligned}$$

The fourth combination,

$$0 - 1$$

is not considered at this time.

INHIBITOR CIRCUIT

A circuit that will produce the correct result for each of the three possibilities is the ''inhibitor'' (fig. 5-1). For simplicity the action is illustrated by using the inhibitor symbol (fig. 5-1A). A truth table, showing the inputs and outputs from the circuit, is shown in figure 5-1B.

The inhibitor circuit produces an output when there is a signal on M but not S. An inverter connected in the S input path causes the S input to the AND-element to be inverted. (The inverter output to the AND-element will be 1 when the S input is 0, and 0 when the S input is 1.) Thus, a 1 output is produced when (and only when) the voltage pulses at the input represent $M\overline{S}$. For all other input combinations the inhibitor output will be 0. The actions of this circuit thus satisfy the basic requirements of a binary subtractor. Although this is a relatively simple illustration of the relationship between mathematics and logic, it is representative of logic principles used to perform more complex arithmetic operations.

The circuit diagram of one form of inhibitor is shown in figure 5-1C. Voltage divider R1-R2 permits a positive potential to be applied to the base of the PNP transistor of sufficient magnitude to produce collector current cutoff. A negative-going input at M produces a heavy current flow through R2 and the increased voltage drop across this resistor causes the base potential to become negative with respect to ground. The transistor conducts and a positive-going voltage is produced at the polarity marked terminal of the T2 primary. By design, the transformer secondary produces a negative-going output. This action occurs only when the M input is negative-going (a 1 condition in this case) and the S input is 0.

Now consider the action when M and S inputs (both negative) are received simultaneously, i.e., when two 1 inputs are applied. The 1 input at M causes a drop across R_1 which is negative to ground and tends to make the transistor conduct. At the same time however, the voltage induced at the polarity marked terminal of the

T_1 secondary is positive to ground. This action keeps the base positive with respect to ground. The transistor does not conduct, however, and consequently, no output pulse is produced when M and S are both in the 1 condition. Stated another way, a 1 input at M will produce a 1 output if an S input is not applied simultaneously.

A second form of inhibitor circuit is shown in figure 5-1D. Here the circuit uses an NPN transistor connected in a grounded emitter circuit. The input circuit (emitter-base) is reverse biased and the collector current is normally cut off. You will recall that changes in collector voltage have little effect on collector current. Thus, inputs across A do not produce a transistor collector current, and the greater portion of the input pulse is developed across R_L at the output.

If positive-going input pulses are simultaneously applied to the A and B inputs, the pulse input at B will produce a forward base emitter bias permitting a heavy collector current. In this condition the collector to ground voltage is low (negligible in this case) and practically all of the input pulse is developed across R1. The small voltage developed across R_L in parallel with the transistor is not of sufficient magnitude to be regarded as an output pulse. Thus, when the input combination of pulses represents $A\overline{B}$ an output pulse is developed across R_L. No output pulse is produced for any other combination of input pulses; this circuit performs as a binary subtractor.

Although an inhibitor is basically a subtractor, its uses are not limited solely to subtractor circuit operations. The inhibitor is used in the exclusive OR circuit. The truth table for the Exclusive OR function is shown in figure 5-1E.

THE EXCLUSIVE- OR CIRCUIT

A conventional OR circuit with inputs A and B produces an output signal if:

(1) A signal exists at A: or
(2) A signal exists at B: or
(3) A signal exists at both A and B.

Sometimes a computer needs a circuit that will produce an output only for condition (1) or condition (2). This type circuit, called an exclusive-OR circuit produces an output signal at F (see the truth table in the fig. 5-1E) if a signal exists at A (but not at B) OR at B (but not at A). No signal is produced at the output if a signal exists at both inputs. In other words, an output is obtained only if the inputs are different. Symbols make this more obvious:

$$F = A\overline{B} + \overline{A}B.$$

Before continuing, let us reexamine both of the terms at the right of the above equation. You can see that $A\overline{B}$ and $\overline{A}B$ are the symbols for an inhibitor. Therefore an exclusive-OR circuit is simply a set of two inhibitors connected by an OR, (fig. 5-1F).

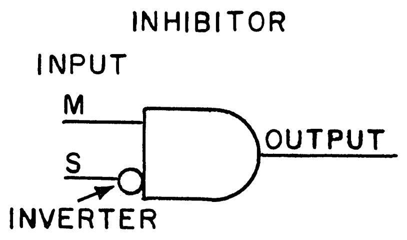
The same circuit can be simplified to two inputs as shown in figure 5-1G. A third connection of this function is shown in figure 5-1H. Here both inputs feed simultaneously into an OR circuit and an AND circuit. The OR output is the "regular" input to an inhibitor circuit, while the AND output becomes the inhibiting input. Note that A and B, appearing simultaneously, activate the AND circuit to inhibit an output at F.

A study of the circuit in figure 5-1I will reveal that an input signal (i.e., a positive pulse) at A only, or one at B only, always produces an output signal. A signal at both A and B, however, produces no output, because each input inhibits the other. (Both emitters swing as far positive as their corresponding bases hence neither is forward biased into conduction.)

An exclusive OR circuit using a single transistor and 6 diodes is illustrated in figure 5-1J. The transistor in the circuit is normally cut off. When there is no input at A or B, diodes CR1 through CR5 are nonconducting and CR6 is conducting.

When the A and B inputs are 0, CR6 remains in the conducting state, CR1 through CR5 remain nonconducting and the output is 0. If both A and B are in the 1 state (positive), CR1 and CR2 conduct and a positive potential is developed across R1 of sufficient magnitude to cut off CR6. This action tries to produce a positive going voltage at F. However, the positive A and B inputs are also applied to CR3 and CR4 and a positive voltage is developed at the base of the transistor (with respect to the emitter) which is sufficient to cause the transistor to conduct. This action causes CR5 to conduct a transient pulse through C1, causing the voltage across R2 to remain the same as it was when CR6 was conducting and the voltage at F to remain in the 0 state.

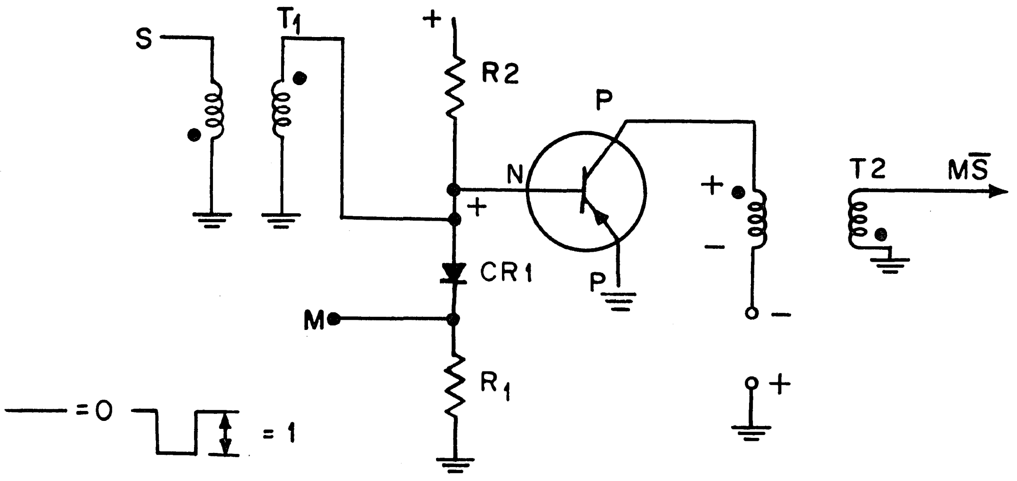
Now consider the action if either A or B is in the 1 state. First note that if CR6 is cut off



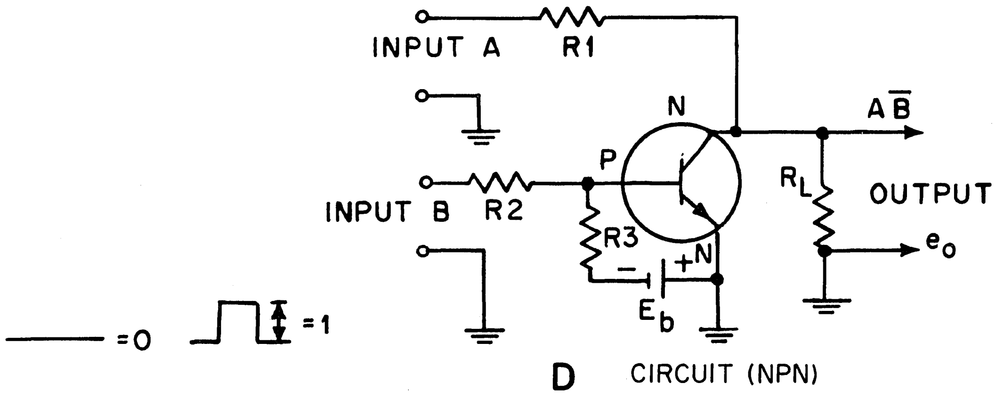
A SYMBOL

INPUT MINUEND (M) (A)	SUBTRAHEND (S) (B)	OUTPUT DIFFERENCE ($M\overline{S}$) ($A\overline{B}$)
0	0	0
1	0	1
1	1	0

B TRUTH TABLE



C CIRCUIT (PNP)

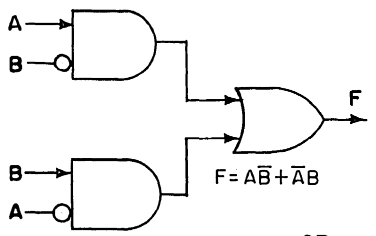


D CIRCUIT (NPN)

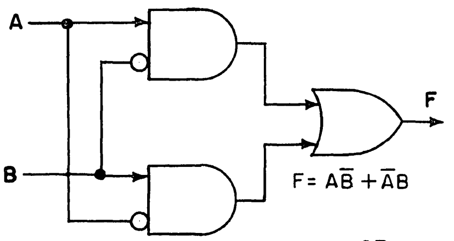
Figure 5-1.—Inhibitor.

INPUT		OUTPUT
A	B	F
1	1	0
1	0	1
0	1	1
0	0	0

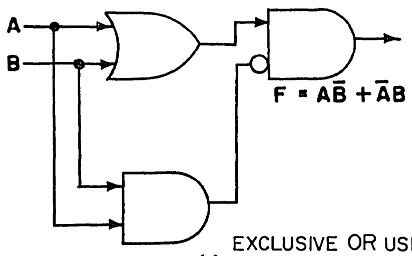
E TRUTH TABLE



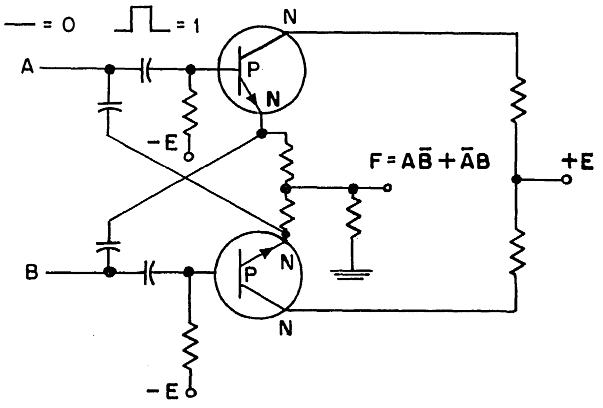
F EXCLUSIVE OR USING FOUR INPUTS



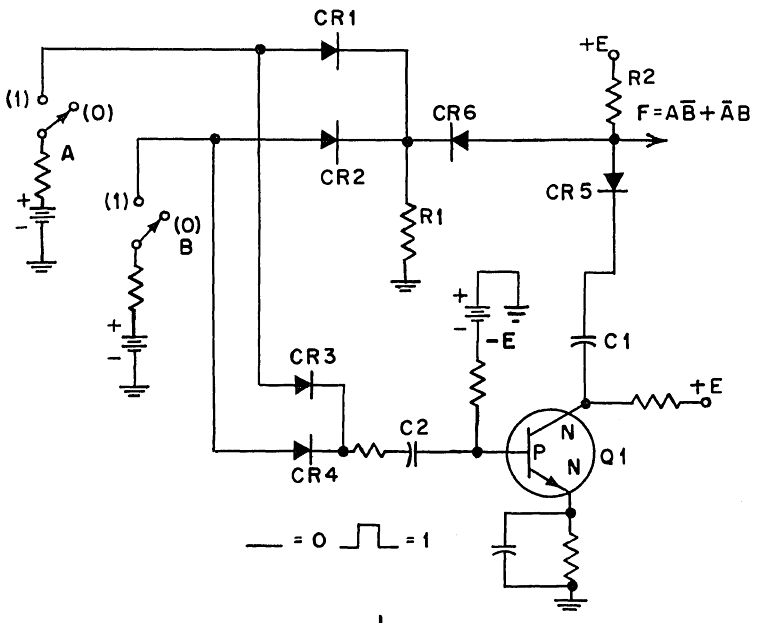
G EXCLUSIVE OR USING TWO INPUT



H EXCLUSIVE OR USING AND, AND INHIBITOR CIRCUIT



I EXCLUSIVE OR USING TWO TRANSISTORS



J EXCLUSIVE OR CIRCUIT USING A SINGLE TRANSISTOR AND DIODES

124.38

Figure 5-1.—Inhibitor—Continued.

and CR5 remains in its normal cut off state, (either A or B input alone is not sufficient to cause the transistor to conduct) the voltage at F will rise. If A is in the 1 state (and B is zero), the conduction of CR1 produces a voltage across R1 which causes the cathode of CR6 to go positive by an amount which cuts off CR6. This satisfies the condition $A\overline{B}$ and a 1 state is produced at F. The same action occurs when B is 1 and A is zero, and the 1 output at F indicates the condition $\overline{A}B$. Thus the exclusive OR function is performed.

ADDERS

A binary adder circuit responds to binary numbers (in the form of on-off pulses received at two or more of its input terminals) in a manner which produces the sum of the received pulses at its output.

The addition process in adder circuits is complicated somewhat when "1" inputs are received in a manner which creates a "carry" to the next higher level. Adder circuits which are not capable of advancing the carry digit are called "half-adders." Conversely, adder circuits which can advance the carry digit to the next higher level are called "full-adders."

Before studying either of these circuits, consider the following addition of 0011 and 1010. The augend (1010) is the number to which another number is to be added. The addend (0011) is the number which is to be added to another number (the augend). The carry is the number generated when the sum of two or more binary digits exceeds 1. Thus, 1 plus 1 is 0 with a 1 carry.

	BINARY	DECIMAL
AUGEND	1010	10
ADDEND	0011	3
CARRY	1	
	——	——
	1101	13

All of the steps in this addition can be summarized by the following three rules:

1. If the addend and augend bits in a column are different, write 1 in the sum.
2. If both addend and augend bits are 0, write 0 in the sum.
3. If both addend and augend bits are 1, write 0 in the sum and carry a 1 to the next significant column.

These three rules can be expressed even more concisely in the form of the following truth table.

Table 5-1.—Conditions for the Addition of Two Binary Numbers.

ADDEND	AUGEND	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The possible conditions for the addition of two binary numbers depicted in table 5-1 do not include all possibilities. When the addend and augend contain more than two columns, a circuit designed to execute such additions must be able to accept and add a carry digit from a previous column. Table 5-2 shows all possibilities which may be encountered in the addition of such numbers where C represents a previous carry which is now to be added to the sum of the addend and augend in that column, C_n represents a new carry which is to be added to the next column, and S represents the sum.

Table 5-2.—f(A, B) Showing Previous Carry, New Carry, and Sum.

A	B	C	S	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

From the truth table (table 5-2) it can be seen that the minterm equation (condition of the variables for which f = 1) for the sum (S) is:

$$S = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,\overline{C} + A\,B\,C$$

and that the minterm equation for the carry digit is

$$C_n = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

The truth table (and consequently the equations) indicates that the sum is 1 when only one input (either A, B or C) is 1, or when all inputs (A, B and C) are 1. The condition when A and B are 1 is not present in the sum equation since the sum output for this condition is 0. The C_n equation indicates that a new carry is generated whenever any two of the variables A, B or C is 1 or when all variables (A, B and C) are 1.

The C_n equation can be simplified as follows:

$$C_n = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

Simplifying C_n

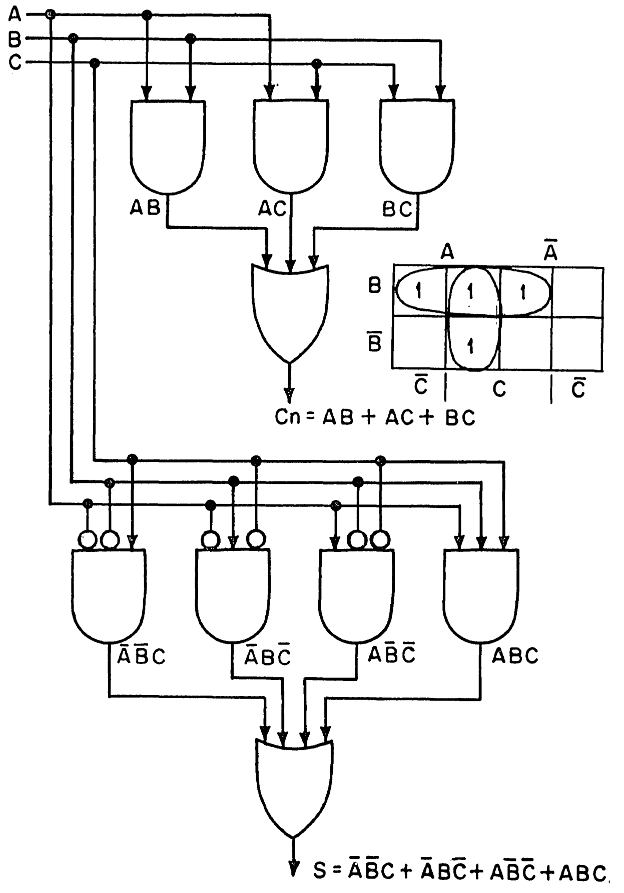
$$\begin{aligned} &= \bar{A}BC + A\bar{B}C + AB(\bar{C}+C) \\ &= \bar{A}BC + A\bar{B}C + AB(1) \\ &= \bar{A}BC + A\bar{B}C + AB \end{aligned}$$

The Veitch Diagram Simplification is constructed as shown in figure 5-2. Thus,

$$C_n = AB + AC + BC$$

AND, OR, and INHIBITOR circuits can be combined as shown in the lower portion of figure 5-2 to produce a 1 condition at the sum output when any one of the conditions shown in the sum equation exists at the input. Likewise, the logic circuit arrangement in the upper portion of the figure produces a 1 carry output to the next higher order when any of the conditions contained in the carry equation exist at the input.

Remember, that this circuit is designed to accommodate only two binary digits plus a carry digit contained in a single column. The addition of several columns of digits would require a similar circuit for each column (for parallel operation) or sufficient time between the application of binary inputs during serial operations to permit the carry function to be executed.



124.39

Figure 5-2.—Three-input adder.

SERIAL AND PARALLEL OPERATION

Adder circuits can be designed to accept input pulses in either serial or parallel form. In parallel operation, each bit of a binary number is carried on a separate transmission line. In serial operation the binary information is carried in the form of a series of timed pulses. The relative advantages and disadvantages of each form of data transmission are fairly obvious. Parallel operation is much faster because all bits are transmitted simultaneously, while serial transmission can pass only 1 bit at a time. On the other hand, serial transmission is much cheaper than parallel and requires less equipment. However, a true evaluation of the two methods is not quite so simple. For example, because the carry digit must be accumulated in the next higher order, the addition of N number of digits involved in an addition in a parallel machine is not necessarily accomplished N times as fast as can be done by a serial machine. Neither is it true that a parallel machine requires N times as much equipment. Thus, it would not be accurate to say that either of the two types of machines has a net advantage over the other except where all features of the system are known and evaluated.

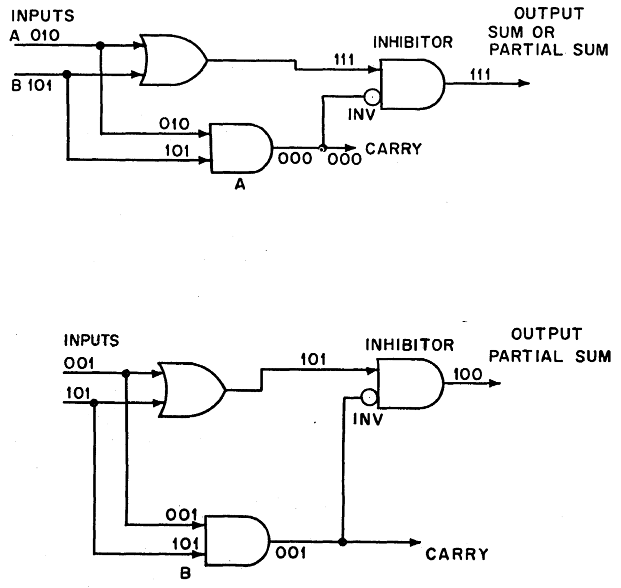
The choice of serial or parallel operation is also affected by the type of storage and the accessibility of stored data. The time required to read-up data from storage and to write information in storage are important considerations which add to the total time required to complete an arithmetic operation in both serial and parallel machines. This time is inherently available in serial machines because of the time between pulses in a train, but must be made available in parallel machines by introducing delay periods.

HALF-ADDER

We will first consider a form of adder circuit used in serial machines. A serial half-adder circuit is shown in figure 5-3. The OR circuit (fig. 5-3A) produces an output when there is an input pulse applied to either the A or B input. The lower AND circuit produces an output when A and B input pulses are received simultaneously, a condition which produces a carry. Note that this is the same circuit as that for the exclusive OR of figure 5-1H.

First consider the action with 010 at the A input terminal and 101 at the B terminal as shown. Although the inputs are applied simultaneously to the OR circuit and the lower AND circuit, only the OR circuit operates to produce an output. The two simultaneous 1 inputs necessary to produce a 1 output from the lower AND circuit do not occur in this example and thus no carry is produced. The output of the OR circuit (111) is therefore the true sum of the binary numbers appearing at A and B. The inverter at one terminal of the second AND circuit (inhibitor, discussed earlier) causes this input to be in the 1 state when the lower AND circuit output is 0 (as in this case) and the 111 output from the OR circuit is passed through the inhibitor to the half-adder output.

The addition process is not so simple when two 1 inputs occur simultaneously in the numbers to be added as shown in figure 5-3B. Here, the OR circuit produces a 101 output (which is not a true sum of inputs A and B) while the AND circuit produces a 001 output. The 1 input to the inverter at the inhibitor input causes this circuit to block the passage of the 1 output from the OR-circuit in the right (least significant) column, in this case, and the inhibitor output is 100. This, again, is not the true sum of the input binary numbers (001 and 101). Further, a true sum of these numbers cannot be obtained in a half-adder circuit. In order to obtain the true sum, the carry generated by the AND-circuit must be utilized in a subsequent half-adder circuit (2) as shown in figure 5-4.



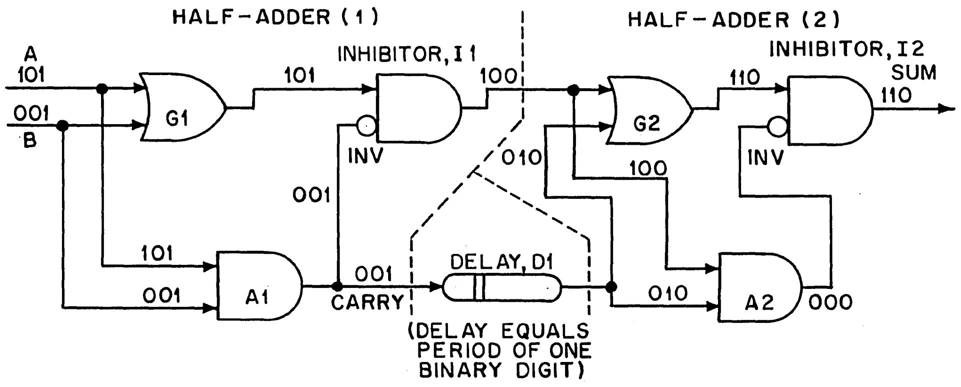
124.40

Figure 5-3.—Half-adder.

FULL ADDER

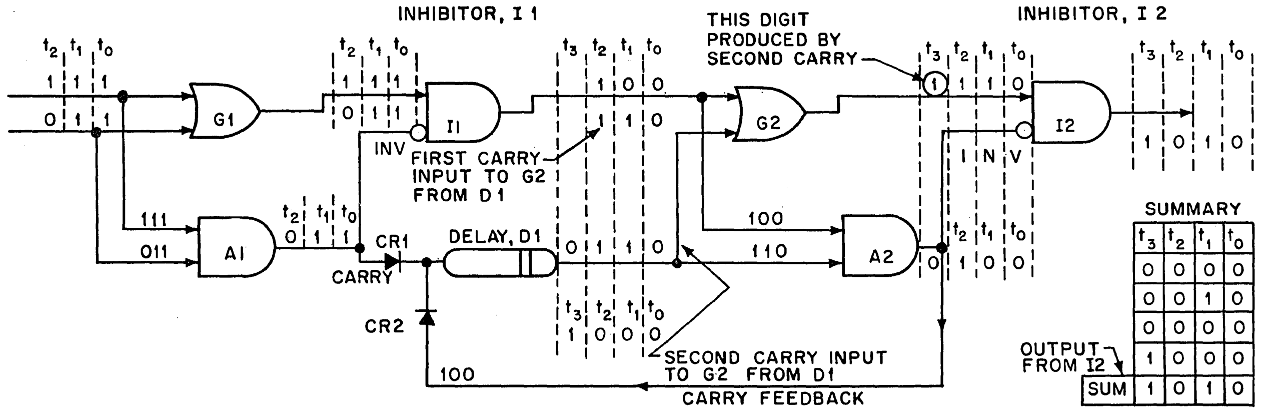
In this circuit, the actions to produce the 100 inhibitor I1 output and the 001 carry output from A1 are the same as described above. The 100 output of I1 is fed to the upper terminal of OR circuit G2. The carry output from A1 is delayed one bit-time (a delay equal to the period of one binary digit) by D1, so that the 1 output from D1 will be applied to the lower terminal of G2 displaced one position to the left—to the next higher order column. This action advances the carry to the next higher order, and thus produces a 110 output from G2. Because there is no further carry generated at the output of A2, inhibitor, I2, passes the number 110 to the sum output terminal. This is the true sum of the A and B inputs. The circuit is called a "full-adder" because it is able to produce a true sum of two (or more) inputs by generating and utilizing the carry when two 1 inputs occur simultaneously.

The full-adder circuit in figure 5-5 is designed to advance a carry (in serial addition) through as many columns as necessary. This



124. 41

Figure 5-4.—Full-adder.



124. 42

Figure 5-5.—Full-adder for accumulating multiple carry digits.

action is accomplished by circulating a carry digit through the carry feedback circuit comprising A2, CR2, and D1, each time a 1 condition exists at the A2 output. The following example will illustrate this feature.

Consider the addition of 111 and 011 as shown. The serial columns are applied at times/To, T1, and T2, respectively. OR-circuit, G1, produces a 111 input to one terminal of inhibitor, I1, with one pulse applied during each time interval. AND-circuit, A1, produces an inhibitory input to I1 during time intervals t0, and t1 so that the output of I1 is 100.

The carry digits (011) from A1 are also fed through CR1 (an isolating diode) to delay line, D1, where each digit is delayed one bit time. The output (110 corresponding to time intervals t2, t1, and t0 reading from right to left) is applied to the lower input terminal of G2, causing the output of this circuit to be 110 during the time intervals shown.

Note that the input to G2 during the intervals t2, t1 and t0 is also applied to one terminal of AND circuit A2, and that the first carry output (110) is applied to the other terminal. Thus, A2, produces a 100 output during intervals t2, t1, and t0.

The 1 output of A2 during period t2 causes the 1 input to I2 during the t2 interval to be inhibited, and the output of I2 during this period is 0. The I2 output during intervals t0 and t1 appear uninhibited. Thus, the output from t2 to t0 is 010.

The presence of a 1 in the A2 output indicates that a carry is yet to be added to one of the remaining columns before the true sum can be produced. The carry digit is fed through CR2 to D1 where it is delayed and shifted into the period t3. The second output of D1 (shown as the second carry input to G2) is passed through G2 during the t3 interval and combined with the serial train already at the output. This action produces the true sum (1010) of the binary inputs at G1. Note that a carry has been advanced from the first column to the fourth. In a similar manner, this circuit can advance the carry through any number of columns as required.

A parallel adder circuit which uses flip-flops as the basic elements is treated in chapter 4 of this text. In the discussion of parallel addition it is shown that a carry digit is generated when any flip-flop changes from the set to the reset condition and that this carry is fed through a delay circuit to the next higher order. The delay circuit prevents the accumulation of the carry digit in the next higher column until after the addition of the augend and addend in that column.

Addition can be accomplished with a circuit which is designed to perform only subtraction operations. Thus we can obtain addition by subtraction. Consider the following example:

Suppose it is desired to add A + B by subtracting only. If we first subtract A from 0 we obtain -A. If we then subtract B from -A we get -A-B, which by simplification is -(A+B). Now subtract - A-B from 0 to obtain A + B which represents the sum.

BINARY SUBTRACTER

A basic binary subtracter is a device that accepts the minuend and subtrahend digits at its input and produces a difference and borrow at its output. A truth table representing the conditions arising from the subtraction of 2 binary digits is shown in table 5-3. This function is illustrated earlier in this chapter by the use of inhibitor circuit, fig. 5-1.

Table 5-3.—Single-Column Binary Subtractor.

MINUEND (M)	0	1	0	1
SUBTRAHEND (S)	0	0	1	1
DIFFERENCE (D)	0	1	1	0
BORROW (B)	0	0	1	0

The Boolean equation which represents the conditions which produce a difference in table 5-3 is:

$$D = \overline{M}S + \overline{S}M.$$

All other conditions produce a 0 difference. The borrow shown in the table exists when $B = \overline{M}S$.

As was true with addition of binary digits, more than 1 binary column is usually involved in the arithmetic process. Thus, the table in 5-3 although valid, is very limited. When more than one column of digits containing minuend and subtrahend is to be subtracted, each digit of the minuend is decreased by the amount of the subtrahend digit in that column, and if the minuend is reduced to a value less than 0 in the process the minuend digit of the next higher order must be reduced by 1—the process by which the borrow is obtained. This is the method of subtraction most familiar to us and, in fact is a method in wide use in computer circuits.

This table is also useful in a process called ''half-subtraction'' which does not take into account a borrow from a previous order.

Table 5-4.—Subtraction Table Showing Minuend (M), Subtrahend (S), Borrow (B), Difference (D) and New Borrow (B_N).

M	0	1	0	1	0	1	0	1
S	0	0	1	1	0	0	1	1
B	0	0	0	0	1	1	1	1
D	0	1	1	0	1	0	0	1
B_N	0	0	1	0	1	0	1	1

Now consider a more complete subtraction table (table 5-4) which shows the minuend (M), subtrahend(S), borrow (E, which is advanced to that column from a lower order not shown), difference (D), and the borrow (B_N) which is the new borrow generated in the column in which the subtraction is taking place. From this table the difference equation can be derived by considering the conditions which produce a 1 as follows:

$$D = M\overline{S}\overline{B} + \overline{M}S\overline{B} + \overline{M}\overline{S}B + MSB$$

simplifying

$$D = (M + S + B)\ (\overline{MS + SB + MB}) + MSB$$

Although a subtracter unit could be mechanized from this equation, the process of subtraction can be more easily accomplished by complementing and adding. The complement method is the most widely used and is therefore treated in detail.

The complement method of subtraction is used in both serial and parallel machines. This method is used in serial machines as illustrated in figure 5-6A, by complementing the subtrahend number enroute from storage before it is applied to a serial adder. The minuend number is not complemented. Thus the action in the serial adder produces the true difference.

The block diagram arrangement in figure 5-6B is more representative of present day methods of serial complement subtraction. In this case an accumulator is used which contains the minuend digits. Upon receiving control pulses, the number stored in the accumulator is fed to the adder circuit as the minuend. As in the previous example the subtrahend number is fed from storage (when the control pulses are received) through the complementer to the serial adder. The result of the addition is fed to the accumulator for storage.

Parallel subtraction figure 5-6C is accomplished in much the same way. The information from storage is fed through the complementer (on parallel lines) to the parallel adder. After receiving control pulses, the accumulator feeds the minuend to the adder. The results of the addition are stored in the accumulator.

End-around carry is accomplished when necessary by adding 1 to the 0 column. The 1 digit that occurs in the 2^n column, called the overflow digit, is dropped.

MULTIPLICATION

Like all other arithmetic operations, multiplication can be accomplished in computers in several ways. One of the most commonly used methods is multiplication by accumulation. This method basically involves left-shifting of the multiplier and adding whenever a one bit is encountered in any one of the multiplier orders. A similar and probably more familiar method of multiplication used with the decimal system is done by using over-and-over addition.

Consider the example using 236 as the multiplicand and 52 as the multiplier in the decimal number system. If we add 236 using the hundreds, tens, and unit columns in that order 2 times, and 236 entered into the thousands, hundreds and tens columns 5 times, the sum (or product) will be displayed as 12272. Similarly, any number can be multiplied in this manner.

A circuit which is designed to perform multiplication of binary numbers by the accumulation method is illustrated in figure 5-7. The number of switching elements used is determined by the number of digits to be multiplied. The multiplicand is applied on lines A through E, and remains as the static input throughout the multiplication process. The multiplier digits are applied as control inputs on lines W, X, Y, Z.

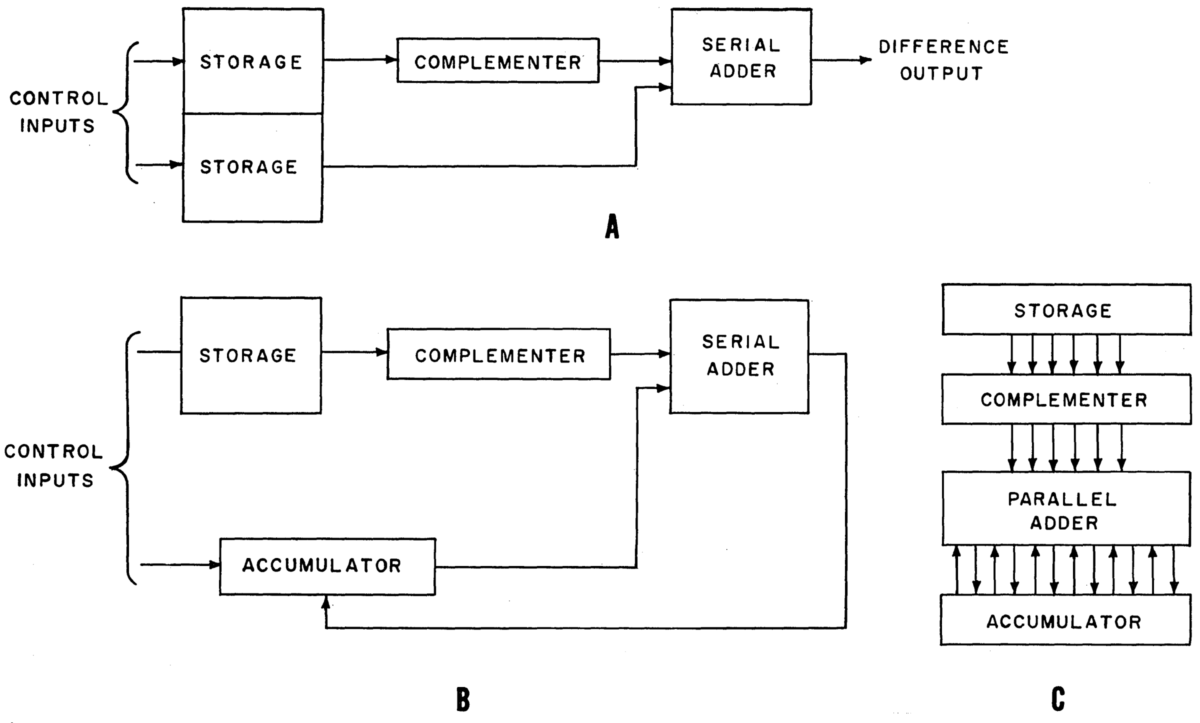
If a 1 bit is present in the first order of the multiplier a shift is not required and the multiplier input lines (multiplication control lines) X and Z are placed in the 1 state. A careful study of the AND and OR elements will reveal that a nonshifted output is developed at the lower terminals I through M. This output represents a partial product which is fed to adder circuits in the accumulator. Now, if the second order digit of the multiplier is a 1 bit, a shift of 1 place to the left is desired before addition and lines W and Z are in the 1 state. Thus, a left shift is produced (1 place to the left) in the upper part of the circuit and is passed through the lower section under the influence of the Z input. The output from terminals H through L represent a second partial product which is fed to adder circuits in the accumulator.

If the third place digit of the multiplier is a 1, the multiplication control lines X and Y are placed in the 1 state. This produces a third partial product output between terminals G and K which is fed to the adder circuits in the accumulator.

Similarly, the multiplication control lines W and Y are used to generate the fourth partial product output between terminals F and J which is added in the accumulator to produce the true product.

SERIAL MULTIPLICATION BY REPEATED ADDITION

The arithmetic unit contains several registers usually referred to as the X, Q, and A registers. These registers are capable of storing or holding a computer word of a length determined by design. In most cases, during multiplication, the Q register (fig. 5-8), holds the multiplier, the X register the multiplicand, and the A register (the accumulator) holds the sum or partial product. Because the product of two



a. Subtraction by complementing and serial addition.

b. Subtraction by complementing and serial addition using an accumulator.

c. Subtraction by complementing and parallel addition.

124.43

Figure 5-6.—Subtraction.

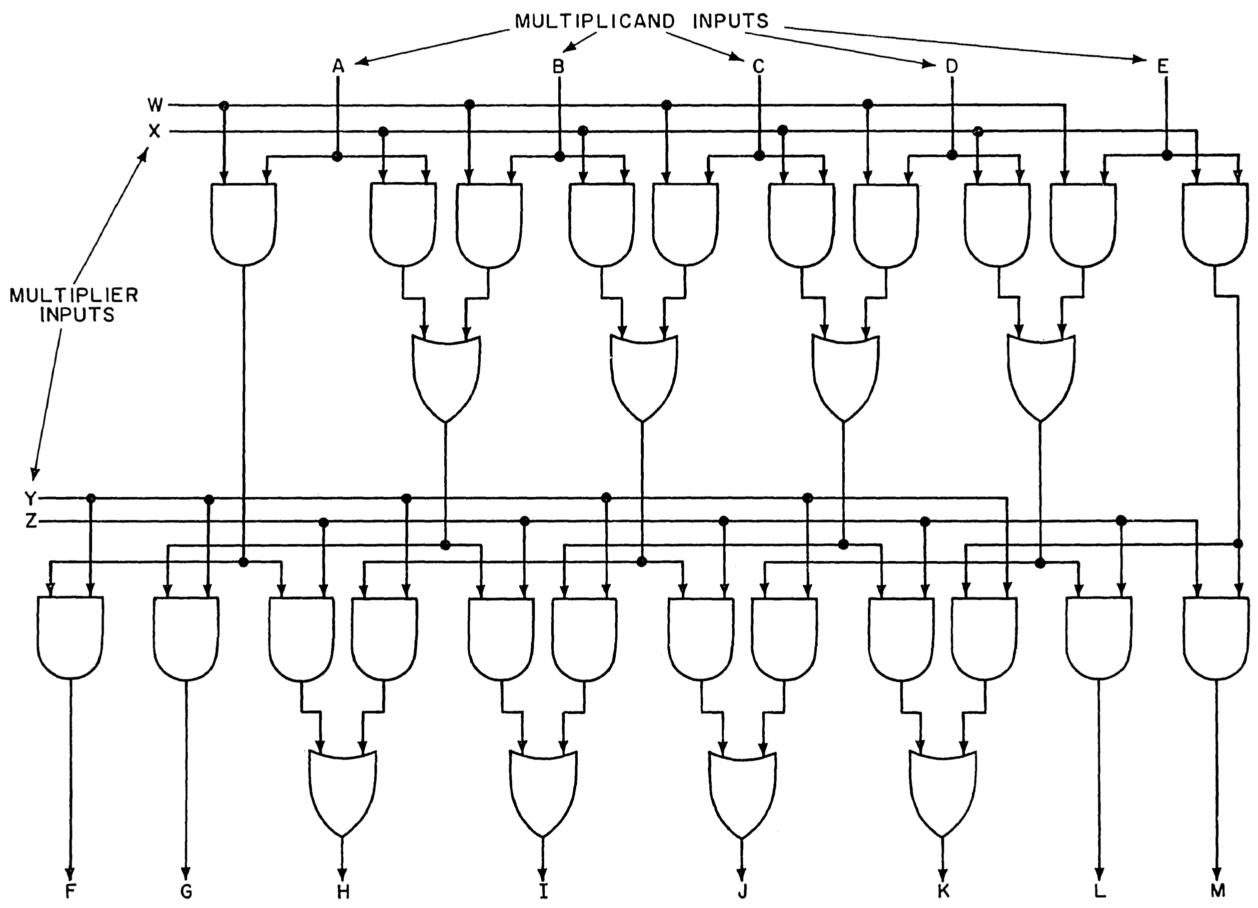
numbers each containing n digits, can contain as many as 2n digits, the accumulator must be capable of holding twice as many digits as either the multiplier or the multiplicand.

The X register reads its input (the multiplicand) into the serial adder in serial form at the same time that the accumulator input is being read through in serial form. Initially the accumulator is cleared to 0.

The multiplier or Q register is actually a counter which determines the number of times the particular multiplicand must be added to yield the correct product. If the multiplicand is to be multiplied by 4, the Q register will initially be set to 4 and will count backwards 1 digit at a time, with the multiplicand and the accumulator contents being added in the proper order for each count.

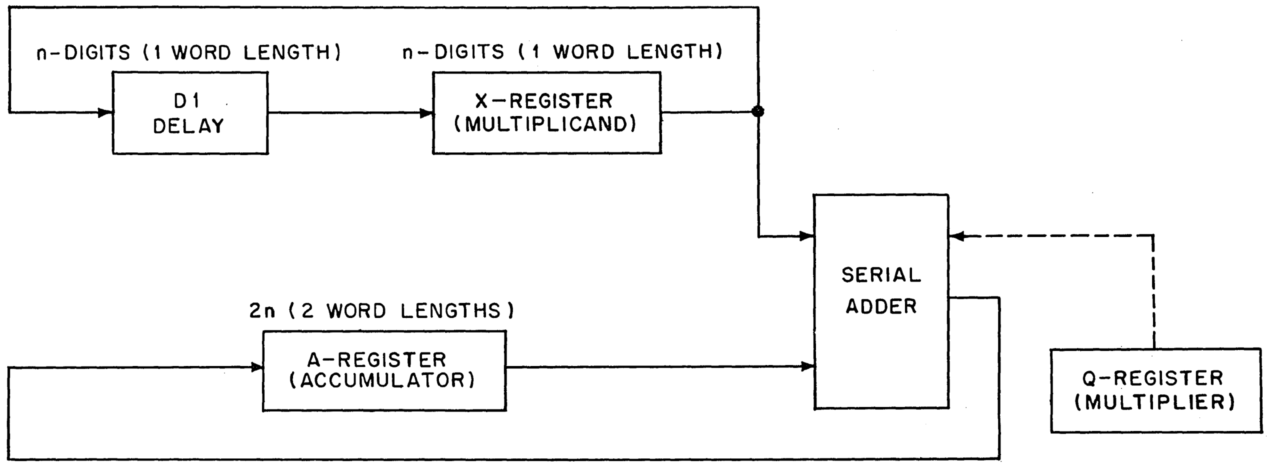
Because the accumulator is initially set to 0000.... the first addition will produce the multiplicand in the accumulator. The second addition is delayed for 1 word length by D1 to compensate for the difference in word lengths of the accumulator and the X register inputs. The delay assures that at the beginning of each count the least significant digit of the Q register and the least significant digit of the A register will arrive in the serial adder simultaneously.

After the first addition, the contents of the Q register is decreased by 1, and, after delay, the second addition begins by again adding the contents of the X register to the accumulated sum. A second delay now takes place in D1, again to ensure that the lowest orders of the X and A registers will enter the serial adder on the third count at the same time. The process continues until the contents of the Q register is reduced to 0 whereby the addition has been repeated the number of times dictated by the multiplier and the accumulator contains the product.



124.44

Figure 5-7.—Parallel left shift logic circuit.



124.45

Figure 5-8.—Serial multiplication by repeated addition.

A similar method which can be used with parallel adders and right shifting is illustrated in figure 5-9A in the form of a flow chart.

Before proceeding with this method first observe the example below. It can be seen that the same product is obtained when the addition of the multiplicand is accomplished using the digits in the multiplier from right-to-left, or from left-to-right.

```
PROBLEM:            234  Multiplicand
                   x123  Multiplier
                   ----

  234                  234
   234                 234
   234                 234
    234               234
    234               234
    234              234
  -----              -----
  28782              28782

Equivalent of           Equivalent of
left-to-right shift     right-to-left shift
```

Thus, multiplication can be (and is) performed by either left-shifting or right-shifting the multiplier.

The blocks in the flow chart in figure 5-9A represent the sequence of events necessary to arrive at a desired result, in this case the product. In general, flow charts actually provide a graphical presentation of a procedure. All of the steps involved in arriving at this procedure are accomplished under the direct influence of the control unit.

Assume that the multiplicand is 11111 (31) and that the multiplier is 101 (5) in the following example. The control sequences are as follows:

Transfer the multiplier (101) to the Q register (fig. 5-9B).

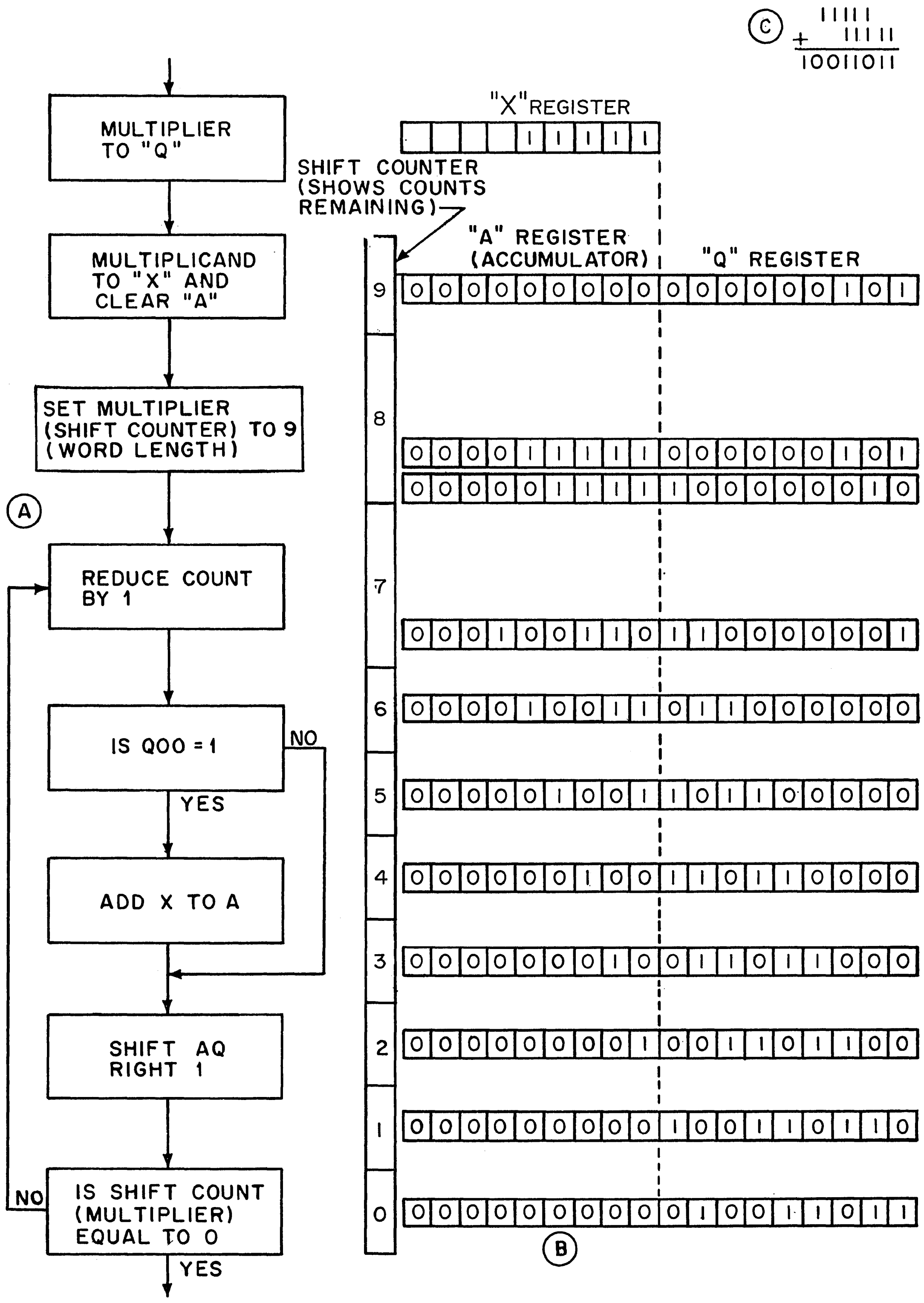
Place the multiplicand in the X register and clear the accumulator (A-register). Set the shift counter to equal the word length (9 bits in this case). The next command is to reduce the shift counter by 1 (this does not produce a shift in the AQ register at this time). Now examine the lowest order bit in the Q register. If this bit is a 1, i.e., if $Q_{00} = 1$, add X (multiplicand) to the A register. This action places 11111 in the accumulator. The next command is to shift the contents of the A and Q registers to the right 1 bit. In executing this function the A and Q registers are joined together as a single 18 bit register and the least significant digit in the A register is shifted into the higher order bit position in the Q register. This action drops the one bit in the lowest order of the Q register and the 0 in the second order is now the least significant digit in the Q register.

At this point we examine the shift counter, to determine if the count has been reduced to 0. If the answer is "no" as it will be in this case since 8 counts remain, the command is issued to reduce the count by 1, and a subsequent command examines the Q register to determine if the lowest order bit is a 1. In this case the answer will be "no" since the 0 bit has been shifted to the least significant bit position. Because this answer is "no" it is not necessary or required that the "add X to A command" be issued and this step is bypassed as indicated in the flow chart. The next command therefore shifts all bits in the A and Q register to the right one bit position, and a 1 bit now appears as the least significant digit in the Q register. The following step again checks the shift counter to see if the multiplier count has been reduced to 0. The answer is again "no" as there are 7 additional counts. Thus, the command to reduce the count by 1 is again issued. Upon checking the Q register to see if the lowest order bit is 1, the answer will be "yes," and the command will be issued to add X (multiplicand) to A. This action adds the multiplicand to the previous accumulator sum as shown in figure 5-9C.

A check of the shift counter reveals that the count has not been reduced to 0 and that 6 counts still remain. Because each of these counts produce a "no" answer when the Q register is checked to determine if the lowest order bit is 1, the shift AQ right 1 command will be issued in sequential order as each bit is checked until the shift counter shows the multiplier count to be reduced to 0. At this time the lowest order bit in the product will be in the lowest order position of the Q register, followed at the left by each higher order bit, revealing the highest order bit in the product in the 8th bit position to the left. Thus, the content of the Q register is the true product.

DIVISION

Division can be accomplished by repetitive subtraction as illustrated in the following example using 36 (as the dividend) and 12 as the divisor. The dividend is reduced by the amount of the divisor for each subtraction.



C
11111
+ 11111
10011011
"X" REGISTER
SHIFT COUNTER (SHOWS COUNTS REMAINING)
"A" REGISTER (ACCUMULATOR)
"Q" REGISTER
MULTIPLIER TO "Q"
MULTIPLICAND TO "X" AND CLEAR "A"
SET MULTIPLIER (SHIFT COUNTER) TO 9 (WORD LENGTH)
A
REDUCE COUNT BY 1
IS Q00 = 1
NO
YES
ADD X TO A
SHIFT AQ RIGHT 1
IS SHIFT COUNT (MULTIPLIER) EQUAL TO 0
NO
YES
B

124.46

Figure 5-9.—Multiplication by addition and shifting.

```
        12 )36
 36
 12  first subtraction
 --
 24
 12  second subtraction
 --
 12
 12  third subtraction
 --
  0
```

The number of subtractions completed is the quotient, in this case, 3. In binary form the same example would be.

```
100100    (36)
  1100    (12)   first subtraction
------
 11000    (24)
  1100    (12)   second subtraction
------
  1100    (12)
  1100    (12)   third subtraction
------
000000    (0)
```

A block diagram for parallel repetitive subtraction is illustrated in figure 5-10A. You will recall that binary subtraction can be accomplished by complementing the subtrahend and adding. This is the process used in the circuit of figure 5-10A. The divisor is stored in the X register, the dividend is in the accumulator (A register) and the quotient, that is the number of times the divisor is taken from the dividend, is stored in the Q register. Because it is necessary to subtract the divisor from the dividend several times, depending on their relative magnitudes, the X register output is applied to the complementor and recirculated so that it is again stored in the X register. This makes possible the read-in of the divisor to the complementor as many times as is necessary to reach the final quotient.

When dividing by the repetitive subtraction method it is possible to obtain a positive remainder, a negative remainder, or 0. When a positive remainder is obtained the subtraction at that level is valid and a bit is entered into the Q register. However, when a a negative remainder is obtained it implies that the divisor was larger than the dividend and that the subtraction processes have been carried 1 step beyond that necessary to obtain the final integral quotient. When a 0 is obtained the divisor will also be subtracted from the 0 and a negative remainder will again be obtained. This, too, implies that the subtraction has proceeded one step too far. Thus, it is necessary to utilize a sensor circuit which is capable of determining whether the remainder is positive or negative. If the remainder is positive the sensor must feed the divisor input to the complementor to permit the divisor to be complemented and added. The sensor element must permit complementing each time the remainder is positive.

When a negative remainder is obtained (as a result of a 0 difference or a divisor which is larger than the dividend) the sensor must feed the correct voltage to the complementor to prevent complementing so that the amount subtracted on the previous step will be added. Thus the step beyond that necessary to obtain the integral quotient is nullified.

Although this is a rather simple procedure it is not very practical when we consider the amount of time necessary to divide a large dividend by a small divisor.

A more practical method involves division by the familiar long-hand method. In this method we make an inspection to determine the number of times the divisor can be subtracted from the highest order quantity which is greater than the divisor within the dividend and enter this number as the first digit in the quotient. We then get the remainder and shift the divisor one position to the right. Here we determine how many times the divisor can be subtracted from this portion of the dividend and enter this number as the second highest order digit in the quotient. After subtraction to obtain the remainder we again shift the divisor to the right one position and repeat the process until the remainder after the subtraction from the lowest position is less than the divisor. This yields the final quotient. This process is illustrated below using decimal numbers as follows:

```
     3117.
 15/46763.0
    45
    --
     17
     15
     --
      26
      15
      --
      113
      105
      ---
        8
```

In binary division, (and common to the shifting method of division) some procedures must be used to determine the number of higher order bit positions of the dividend into which the divisor can first be entered to yield a 1 bit in the quotient. In some computers, an attempt is made to subtract the divisor from the higher orders of the dividend. If the result is positive the dividend is larger than the divisor, and a 1 bit is entered in the quotient. If the result is negative, the machine automatically negates this step and the divisor is shifted 1 bit position to the right. This action, in effect, halves the divisor, and, because the dividend now contains 1 bit position more than the divisor, the subtraction is valid and the division proceeds. This procedure is illustrated below.

(Check for position of first bit in quotient. When remainder is negative, i.e., not a real number, this step is nullified.)

```
                                       1011 (12)
                            1100(12)/10000100 (132)
                                     1100
                                     ----
                                     -100
                                     10000100
Result of a right shift ──►           1100
                                      --------
                                      1001
                                      1100
                                      ----
Not a real number ─────────►          -101
                                      10010
                                       1100
                                       -----
                                        1100
                                        1100
                                        ----
                                        0000
```

Unless the relative magnitudes of the dividend and divisor are restricted to values between two extremes, it is possible that the first subtraction at the higher order can produce a 1 bit and a remainder which is greater than the divisor. An example is shown below:

```
      1
011/111100
    011
    ---
    100
```

This inevitably will produce an erroneous result. The error can be prevented by restricting the minimum value of the divisor, the maximum value of the dividend, or both. In the above example the divisor could be restricted to 100, and/or the dividend to 101. In any case, the relative magnitude of the two should be such that the remainder will be less than the divisor after the subtraction of the divisor from the highest orders of the dividend.

In some computers (fractional computers) the quotient is always less than 1 and all significant digits appear to the right of the radix point. In these computers the above stated problem is solved, by ensuring that the divisor is always larger than the dividend.

The division process illustrated in the flow chart of figure 5-10B is similar to that discussed earlier for multiplication except that left-shifting is used instead of right-shifting and a subtraction is the arithmetic operation rather than addition. The AQ register holds the dividend, and the X register the divisor. The shift-counter contains the number of subtractions performed, which represents the true quotient (not shown in the figure).

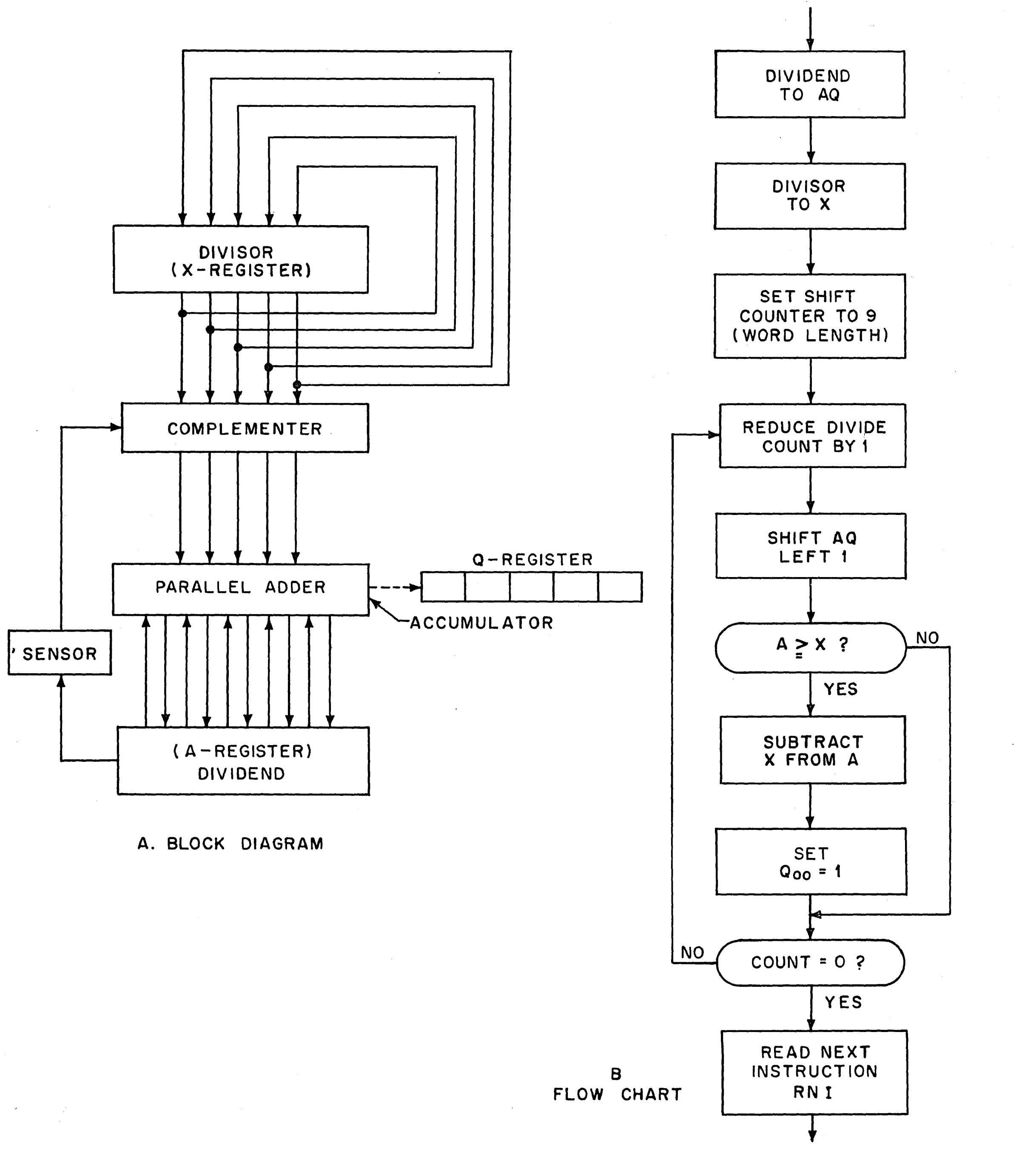
The contents of the AQ register are shifted left one bit position after each subtraction. The highest order bit in the A register is dropped. At the completion of the division process the quotient is contained in the Q register.

Note the comparisons after each left-shift of the AQ contents to determine if A (the dividend) is greater than or equal to X (the divisor). If $A > X$ then X is subtracted from A. This step is followed by setting Q_{00} to 1, that is, setting the lowest bit in the Q register to 1. If the $A > X$ comparison results in a "no", a subtraction does not take place.

At the end of the subtraction process (as indicated when the shift counter reaches 0) a "yes" result is produced for the interrogation "is count = 0". Thus, the order "read next instruction" (RNI) is initiated.

ENCODERS AND DECODERS

It has been stated that computers are designed to operate on data which is in the binary form, that is, either 1 or 0. In most computers, however, it is not practicable to enter numbers and other information into the computer in binary form, as this would require the programmer to spend too much time in the detailed effort of accurately representing large numbers or complicated alphabets and symbols.



127.47

Figure 5-10.—Parallel repetitive subtraction.

Likewise it is not usually desirable to present the final computer output in binary form as this requires too much time in reading and interpreting. Thus it is necessary to perform conversion on both input and output information.

Computers are generally equipped with ENCODERS (units which change discrete inputs into a combination of coded outputs) and DECODERS which transform the internal binary data to its more conventional form at the computer output. Thus the programmer can use the familiar decimal system or the octal system to program the computer, depending on the encoder to convert the data to the binary form as required internally by the computer. Also the output of the internal circuits of the computer is changed by the decoder back into the familiar decimal form. The basic principle involved is illustrated in figure 5-11.

The simple decimal-to-binary encoder (fig. 5-12) receives any one of 10 decimal numbers at its input and produces a binary-coded decimal output. This action can be seen by a study of the circuit using OR logic. There are more complex types of encoders, as will be studied in a later chapter.

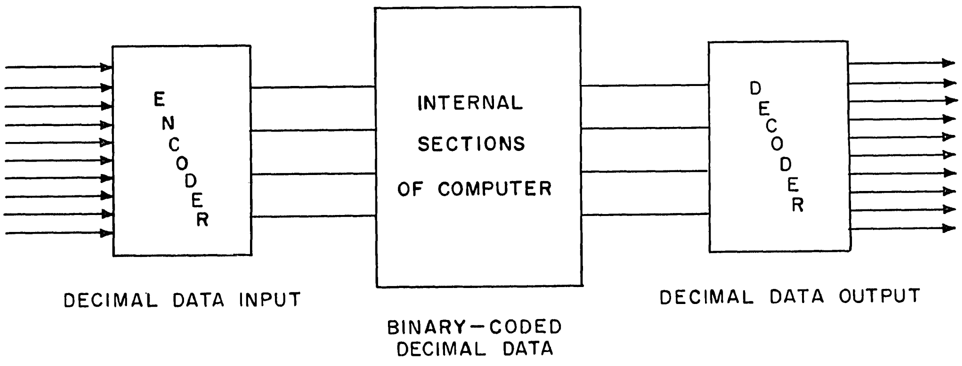
The basic principle of binary-decimal decoding (fig. 5-13) uses flip-flops which provide the inputs to the AND element which in turn produces the decimal output. This type circuit produces an output only when the right combination of flip-flop outputs is applied to the inputs of the AND element. Appropriate use of inverters makes possible the decoding of any binary-coded decimal digit.

Although only binary-coded decimal conversions are illustrated here, similar procedures can be used to encode and decode numbers in all number systems.

A more advanced form of encoding and decoding uses a matrix to translate between number systems. This, by the way, is how the matrix obtained its name. In mathematics, "matrix" means a set of terms which operate on one type of number to produce a second type; in other words, a number translator.

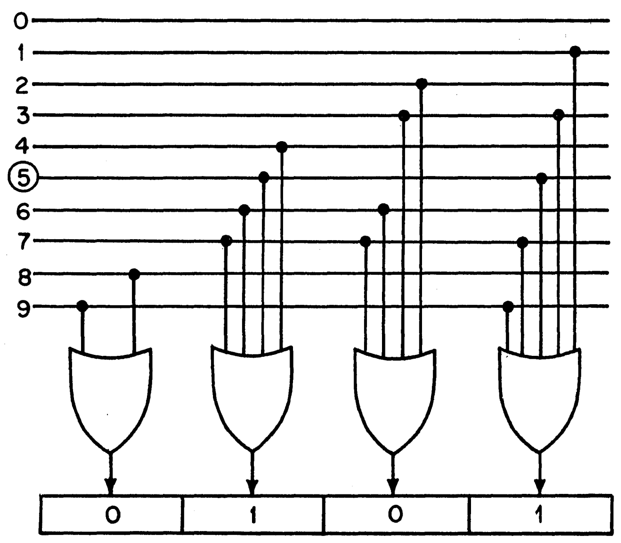
As discussed earlier, binary arithmetic uses only the two digits, 1 and 0. The manner in which these digits are arranged determines their numerical value. The advantage in using the binary system in digital computers is that the two digits (1 and 0) can be easily represented by the two possible stable states of certain electronic circuits and components.

The operation of a matrix is based upon certain principles which were first investigated by Mathematician James Sylvester in the mid-nineteenth century. You will recall that for any particular circuit heretofore discussed, a 'single output is produced if some specific combination of inputs exists. Further, there is a fixed number of possible inputs combinations (depending upon the number of inputs used) each one of which is called a "minterm". A circuit with two inputs, for example, has four minterms: AB, $\bar{A}B$, $A\bar{B}$, and $\bar{A}\,\bar{B}$. A three input circuit has eight minterms: ABC, $AB\bar{C}$, $A\bar{B}C$, $\bar{A}BC$, $\bar{A}\,\bar{B}\,\bar{C}$, $A\bar{B}\,\bar{C}$, $\bar{A}\,\bar{B}C$, and $\bar{A}B\bar{C}$. Thus it can be said that for "n" inputs, there are 2^n minterms. Consequently, for 2 inputs there are 2^2 or 4 minterms; for 3 inputs there are 2^3 or 8 minterms; and for 4 inputs there are 2^4 or 16 minterms, etc. Each minterm indicates one of the possible input combinations that can occur.



124.48

Figure 5-11.—Basic principle of encoding and decoding.



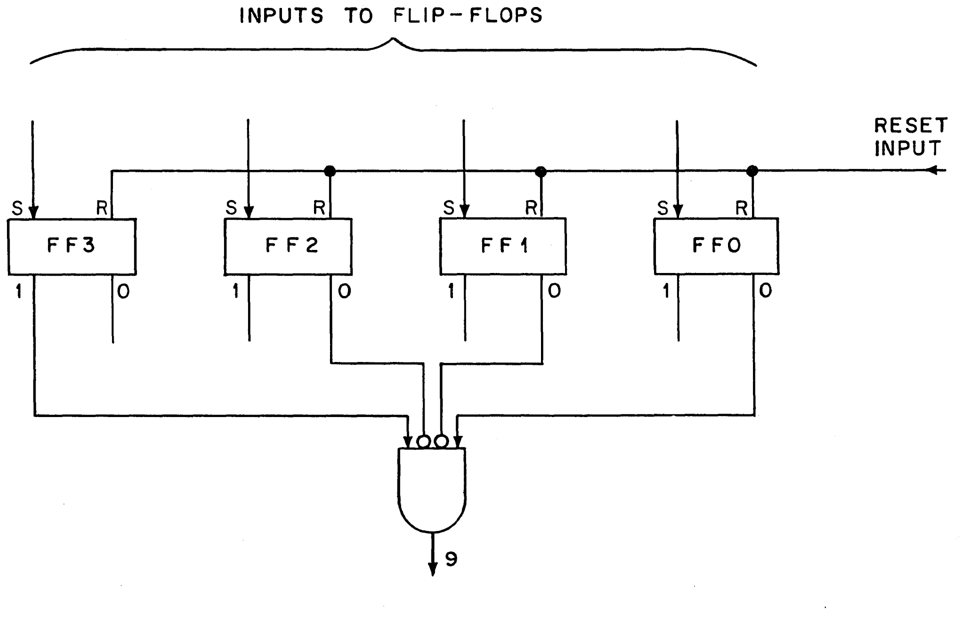
124.49
Figure 5-12.—Decimal-to-binary encoder.

The logic matrix differs in one respect from other circuits discussed in this chapter in that a matrix with "n" inputs has 2^n outputs—one output for each minterm. It produces a signal at one specific output terminal for any one combination of inputs.

Consider the simplified illustration in figure 5-14. The matrix has three inputs, A, B, and C, each of which can be in either the 1 or 0 state. Each combination of digits represents a binary number, so that ABC is interpreted as 111, $\overline{ABC}$ means 000, $\overline{A}B\overline{C}$ means 010, etc. In the illustration, a signal exists at A and at B, but not at C, representing 110. The appearance of this combination of signals at the input will produce a signal on one specific matrix output. By connecting this output to a numbered indicating device, such as a lamp, the binary number 110 is directly converted to its decimal equivalent, 6.

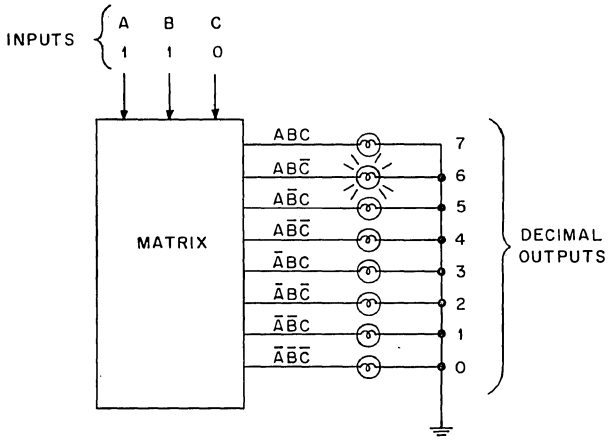
In general, each minterm represents a specific number. For example, $\overline{ABC}$ = 000 = 0, $\overline{AB}C$ = 001 = 1, $\overline{A}B\overline{C}$ = 010 = 2, ... and so on to ABC = 111 = 7. In order to decode higher numbers, the number of inputs and outputs must be increased. Because a logic matrix has an output for each minterm, and every minterm represents a binary number, each output can be labeled with the appropriate decimal value.

Consider the circuit in figure 5-15, which is used to decode the binary coded decimal number 1001 (9). When the magnitude of the voltage at any one of the outputs decreases below the bus voltage to that shown, one or more of the diodes conduct. Because the resistances are



124.50

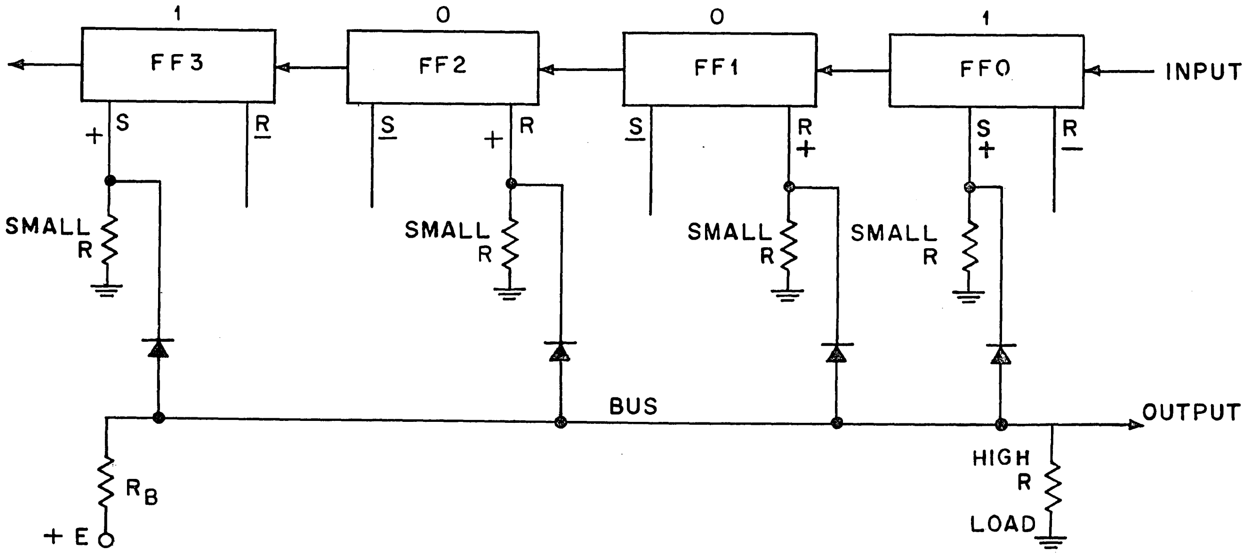
Figure 5-13.—Decimal-to-binary decoder.



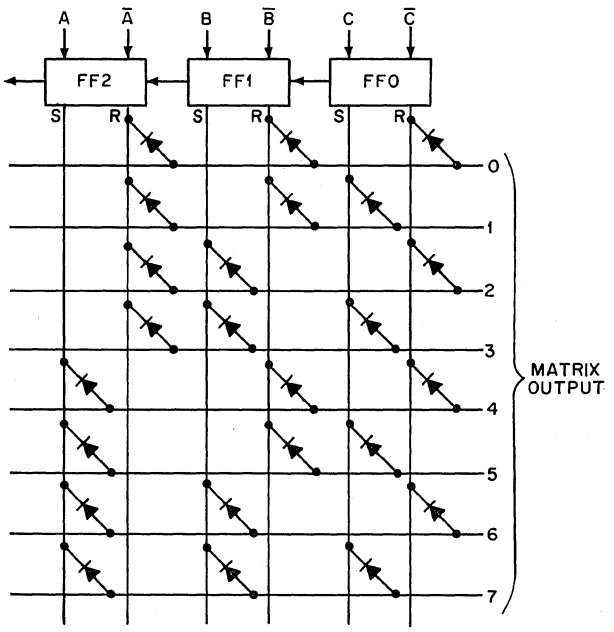
124.51
Figure 5-14.—Three input matrix.

small and in parallel, most of the applied voltage is distributed across R_B. A set output is represented as 1 and a reset output is zero. When, and only when, the flip-flops are in the condition shown, that is from left-to-right set, reset, reset, set, all diodes will be cut off and because of the high R load the voltage at the output rises in the positive direction. This rise in voltage indicates the presence of a binary coded decimal 9 at the input. Similarly, any binary number can be decoded by connecting the diodes to the proper flip-flop terminals.

A more versatile form of diode matrix is illustrated in figure 5-16. Here, diodes are connected to flip-flop terminals in a manner which will decode any one of 8 conditions representing 000 through 111. The diodes are normally conducting (low voltage or zero output) except when the associated flip-flop output is positive going. (This back biases the diode). Thus, a 0 is produced at the matrix output (rise in voltage on the zero level) when all flip-flops are in the reset condition. (Reset terminal is positive going.) A 1 output is produced when both FF2 and FF1 are reset and FFO is set. A 5 is produced when FF2 and FF0 are set and FF1 is reset. In other words the set outputs of FF2 and FF0 and the reset output of FF1 are positive going so that all diodes on the 5-level are cut off and the output voltage on that level is positive going. A study of the matrix for various combinations of flip-flop outputs reveal the conditions which exist when representing any decimal digit from 0 to 7. Higher order numbers can be decoded by increasing the number of flip-flops and connecting additional diodes to indicate higher counts.



124.52
Figure 5-15.—Simple flip-flop decoder.



124.53

Figure 5-16.—Diode decoder matrix.

CHAPTER 6

MEMORY AND STORAGE UNITS

A computer is designed to perform specific operations looking toward the solution of a problem. But before it can solve the problem, it must be instructed or directed as to the steps necessary.

Before the stored-program computer can solve a problem, or perform an operation, it is first necessary to store, in memory, a sequence of instructions (called the "program") and all figures, numbers, and other data which are to be used. With this information, the computer begins its operations, starting at the first instruction in the program or at any other instruction dictated by the operator or determined by conditions previously established.

MEMORY DEVICES

Memory requirements of the computer are largely responsible for the use of binary rather than decimal arithmetic. The circuits designed to perform arithmetic operations using binary numbers are simpler than those which would be required for decimal arithmetic since only two conditions, rather than ten, are represented. Likewise, the storage of binary numbers is simpler since any on-off device can be held in either one of its two stable states to indicate the 1 or 0 condition, respectively, of the bit represented.

Of the many types of storage devices which have undergone extensive study for use in the memory section of computers in recent years, few have emerged as practical. The most commonly used storage devices include; magnetic cores, electrostatic tubes, magnetic drums, magnetic tapes, acoustic delay lines, magnetic disks, magnetic cards and thin film.

The principle involved in representing on-off conditions using magnetic devices is easily understood if we let the magnetized condition of the material (or any small portion thereof) represent the "1" condition, and the nonmagnetized or demagnetized portions represent the "0" condition. In the case of magnetic cores, each core is small and the entire core is magnetized in one direction to represent a "1" state and in the opposite direction to represent the "0" state. Magnetic drums and tapes represent data as small magnetized or nonmagnetized areas indicating 1's and 0's respectively, or areas of one polarization for 1's and of the opposite polarity for 0's on the surface of the device.

Electrostatic storage involves the development of a small static positively charged area on the face of a cathode-ray tube to represent a "1" condition and a negatively charged area (or an area of zero charge) to represent a "0" condition. In many applications, conventional cathode-ray tubes of the type used in oscilloscopes or in commercial television sets are used.

Storage of binary digits in acoustic delay lines is accomplished by using a column of liquid (usually mercury) into which mechanical vibrations are introduced to represent the "1" condition. The absence of these vibrations represents the "0" condition. A detailed treatment of each of the types of memory devices is given below.

ACCESS TIME AND CAPACITY

Two terms are usually used to describe any storage device. These are access time (how fast data can be extracted from the storage device) and capacity (the amount of data that can be stored). The form of the stored data can be considered to be a third factor involved in the description.

In magnetic storage devices associated with digital computers, magnetic fields are used to represent binary 1's and 0's with current pulses being translated into magnetic fields and vice versa, so that data can be used by both the storage device and the computer.

Access time is determined by the nature of the device. In certain devices, such as cores, equal amounts of time are required to read out any group of stored bits. A core storage system is thus a device with random "access." Any core can be selected for data extraction, and the desired data is available in a fraction of a microsecond.

When magnetic tapes, drums, and discs are used, a fixed amount of time is required to read or write on the surface. However, there is a second and variable amount of time required to position the desired material under the read-write head (discussed later). There is an expression among computer engineers that states: "You always know where your desired data is; it has just passed under the head." This means that you have to wait for the next complete cycle of surface travel (say for a drum to revolve) in order to position the desired data under the head. Surface storage devices are usually classified as nonrandom access devices. The access times given for these devices are often equal to half of the maximum possible access time.

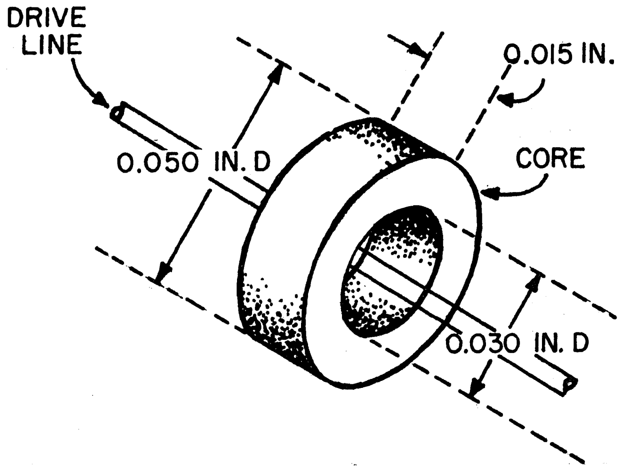
MAGNETIC CORES

Magnetic cores are generally constructed by either one of two methods. The first type of core, called a tape-wound core (not shown), is fabricated by wrapping a tape of magnetic material around a nonmagnetic toroidal form. A toroid is a term used to designate a doughnut-shaped solid object.

The second type is called a "ferrite" core. It is made by molding finely ground ferrite (a ceramic iron oxide possessing magnetic properties), into a toroidal form. The ferrite particles are then heat fused or "sintered" by the application of heat and pressure.

A representative tape-wound core has a diameter of one quarter of an inch. The tape that is used for the winding usually has a width of about one-eighth of an inch and a thickness on the order of one-thousandth of an inch.

A representative magnetic core is shown in figure 6-1. Its outside diameter is 0.050 inch; its inside diameter is 0.030 inch; and its thickness is 0.015 inch. The ferrite core is magnetized by the field produced by a current flowing in a wire (drive line) that is threaded through



124.54

Figure 6-1.—A magnetic core.

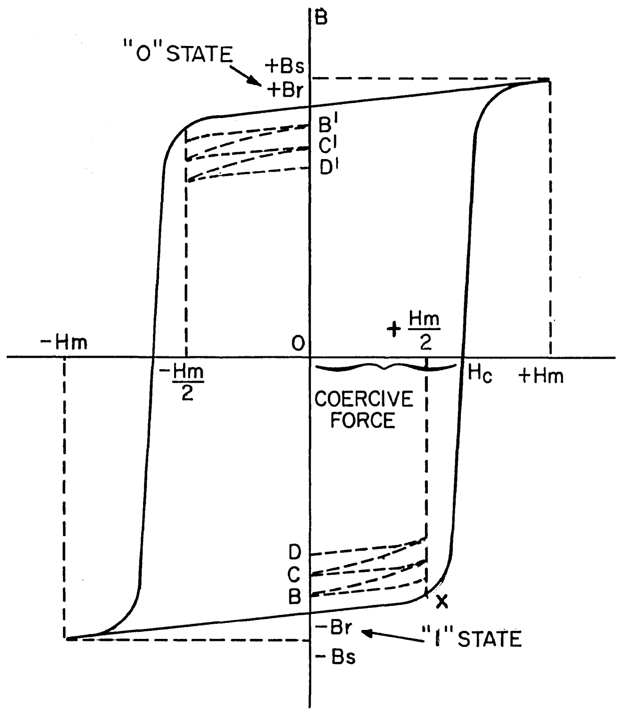
the core (explained later). It retains a large amount of this induced flux when the current is removed. Flux lines can be established clockwise or counterclockwise around the core, depending upon the direction of the magnetizing current. These two unique states represent "0" and "1", respectively.

In this discussion it is assumed that the time required to switch a core from one state to the other is approximately 1.2μs. The drive pulse is presumed to be 2μs in duration with a total of .8μs rise and fall time and 400 milliamperes maximum current.

MAGNETIZATION

The state of magnetization of a core is shown on the hysteresis loop in figure 6-2 which plots magnetic flux density in gausses (B) as a function of the field (induced by the current) in oersteds (H).

In this diagram it is assumed that the core is already in some state of magnetization, such as shown at $-B_r$. If a current flow with a direction (+) that produces a mmf (H) of a given magnitude, $+H_m$, is applied to the drive line (fig 6-1), the flux density increases to saturation as indicated by $+B_s$. When the current is removed, the flux density retained by the core drops slightly along the curve to the level indicated by $+B_r$ (remanence), or the residual flux density. This state is arbitrarily designated here as the "0" state. Another pulse of $+H_m$ would now merely shift the core to $+B_s$ again,



124.55
Figure 6-2.—Hysteresis loop of a ferrite core.

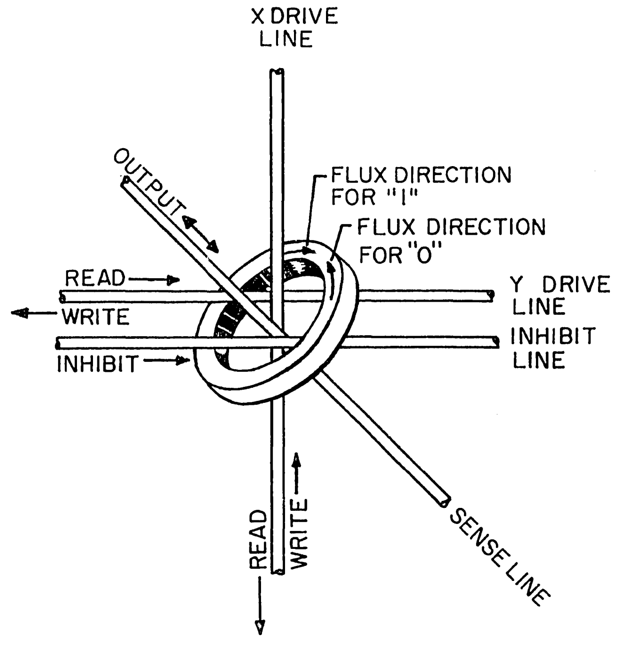
and after the pulse is removed, the core would return to $+B_r$.

If a current pulse of the same magnitude, but in a direction to produce a mmf $-H_m$, is applied to the drive line, the flux density shifts along the curve to $-B_s$ causing a reversal of the flux in the core. When the current pulse is removed, the residual flux density is at $-B_r$, the state that is arbitrarily designated here as the "1" state.

CORE WINDINGS

In order to be able to apply more than one drive current and to sense or inhibit changes in the core condition, it is necessary to use several core windings (drive lines). These windings are indicated in figure 6-3.

Any change in the flux of a core induces a voltage in all wires passing through the core. Hence, the induced voltage on the sense line (winding) is sampled to see if the core switches from $-B_r$ (fig 6-2) to $+B_s$ when $+H_m$ is applied. If a large induced voltage is sensed

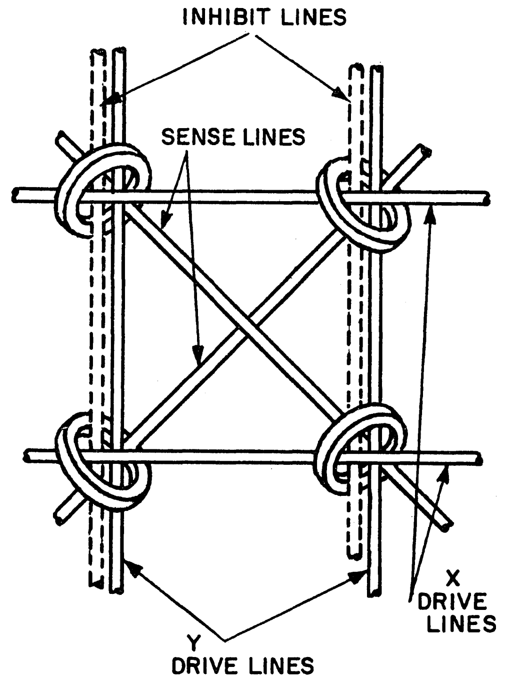


124.56
Figure 6-3.—Magnetic core showing "X", "Y" inhibit and sense lines.

(over 50 millivolts), the core was in the "1" state and has been switched from $-B_r$ to $+B_s$ (the "0" state). Because the contents of the core are determined in this manner, the current pulse corresponding to mmf $+H_m$ is called a "Read pulse." Because the condition stored before sensing is destroyed during read-out, a memory utilizing this type of magnetic core storage element is referred to as a destructive-readout memory. If the data which was stored is to be used again, a restore (or rewrite) function is initiated to return the core to its original state.

OPERATION OF A CORE MATRIX

In operation the memory section contains thousands of cores which are arranged in a matrix. The windings through the cores actually consist of single wire conductors as shown in figure 6-4. The conductors threaded through cores along the horizontal ("X") axis are called "X" drive lines. The conductors threaded through cores along the vertical ("Y") axis are called "Y" drive lines.



124.57
Figure 6-4.—Basic construction of a magnetic core matrix.

The sense line (fig. 6-5) is threaded through all the cores in the matrix (four matrices are shown- one in each quadrant of the board). Maximum voltage is induced in the sense line when a core switches from 1 to 0.

The inhibit (I) lines (or windings) may be threaded vertically (parallel to the X drive lines) or horizontally (parallel to the Y drive lines). A pulse on the inhibit line prevents switching of any core on that line although a drive pulse is present on the associated X or Y drive line. In some cases the inhibit lines are vertical in some sections and horizontal in others to equalize the loading effect on drive lines by the inhibit lines.

HALF-CURRENTS

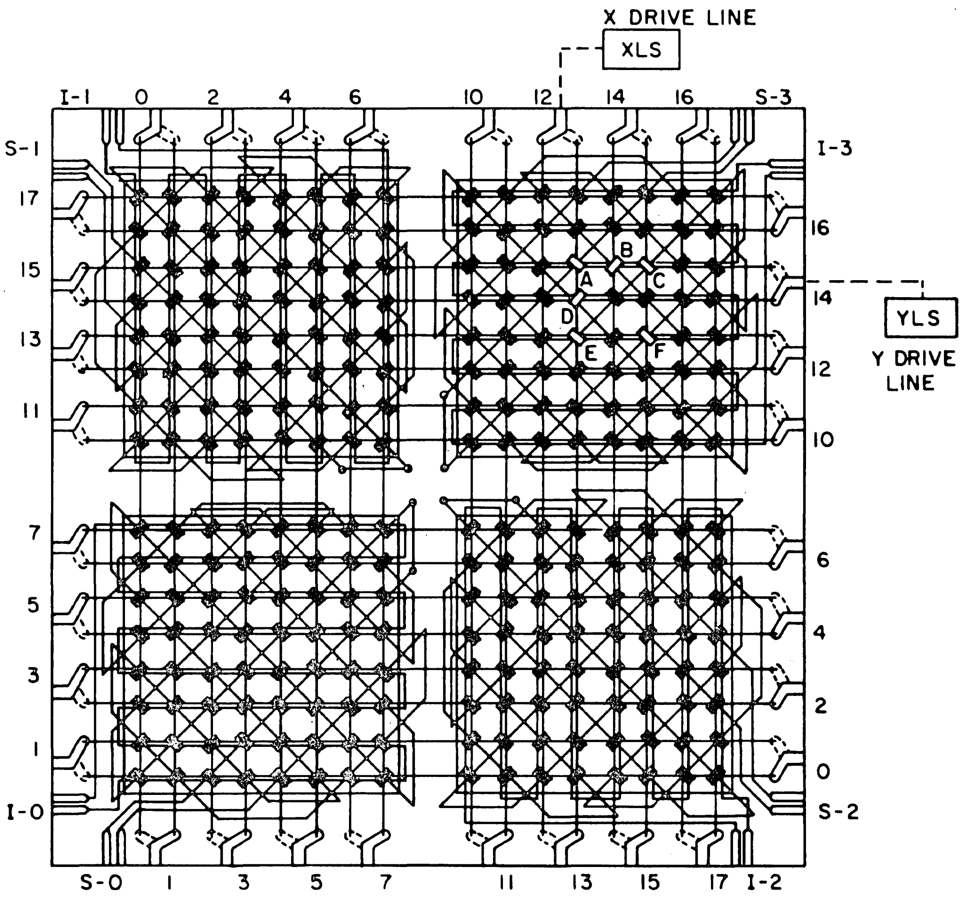
The operation of core matrices depends on the ability of each core to distinguish between current levels on its read/write windings (R/W drive lines fig. 6-3). Each core in a plane is linked by four windings. Two of these windings, the "X" and "Y" drive lines, determine the address of the core. To operate on the addressed or selected core, "half-amplitude" current pulses are applied to each selected drive line so the core at the intersection of the two selected drive lines is the only core that receives a net mmf, or full amplitude pulse, of H_m. All other cores on the two selected drive lines receive only the half-strength field associated with the half-amplitude current on one drive line.

As seen in figure 6-2, the coercive force, H_c, is the mmf required to reduce the core magnetism to zero. The drive currents are selected so that $H_m/2$ (half current) is less than H_c and, as a result, is insufficient to switch the core. The sum of the two drive currents, $H_m/2 + H_m/2 = H_m$, is greater than H_c and switches the selected core in just over one microsecond.

When a core receives a half-current pulse, the mmf produces a change in the flux density of the core. Assuming that the core is in the "1" state at $-B_r$, a half-read pulse causes the core flux to shift along the hysteresis loop to the point x limited by $H_m/2$ and then return to a slightly lower remanent value, such as point B when the mmf is removed. Since the core is now operating in a slightly smaller loop, further half-current pulses again reduce the remanent flux to point "C". This effect soon reaches a limit as at point D. When the core is in the "0" state ($+B_r$), half-write pulses produce a similar effect.

The shift in flux, caused by a half-current pulse, induces a small voltage in the sense winding. The amplitude of this voltage is a function of the squareness of the hysteresis loop. The squareness ratio, R_s, is defined as the ratio of the flux density value at $H_m/2$ to that at $+H_m$. Values of R_s range from a practical limit of 0.7 to an ideal limit of 1.0. A representative squareness ratio value for cores used in computers is about 0.9. A core with a low R_s has a greater shift in flux for a given half-current pulse.

In an earlier discussion it was pointed out that the computer must store large numbers of instruction words in memory—the exact number depending on the particular program and the amount of data involved in executing the programmed steps. In a given computer if each instruction word contains 48 bits and if the program to be stored contains 4000 instruction words, 192,000 cores must be available in memory for instructions alone. Several times this number of storage locations is sometimes needed for storing the data involved. Actually some computers contain over a million cores. Although



124.58

Figure 6-5.—A simplified magnetic core board, 16 by 16 array.

this is a large number of cores, their small size and inherent ability to switch from one binary state to the other make them especially suitable for use as computer storage components.

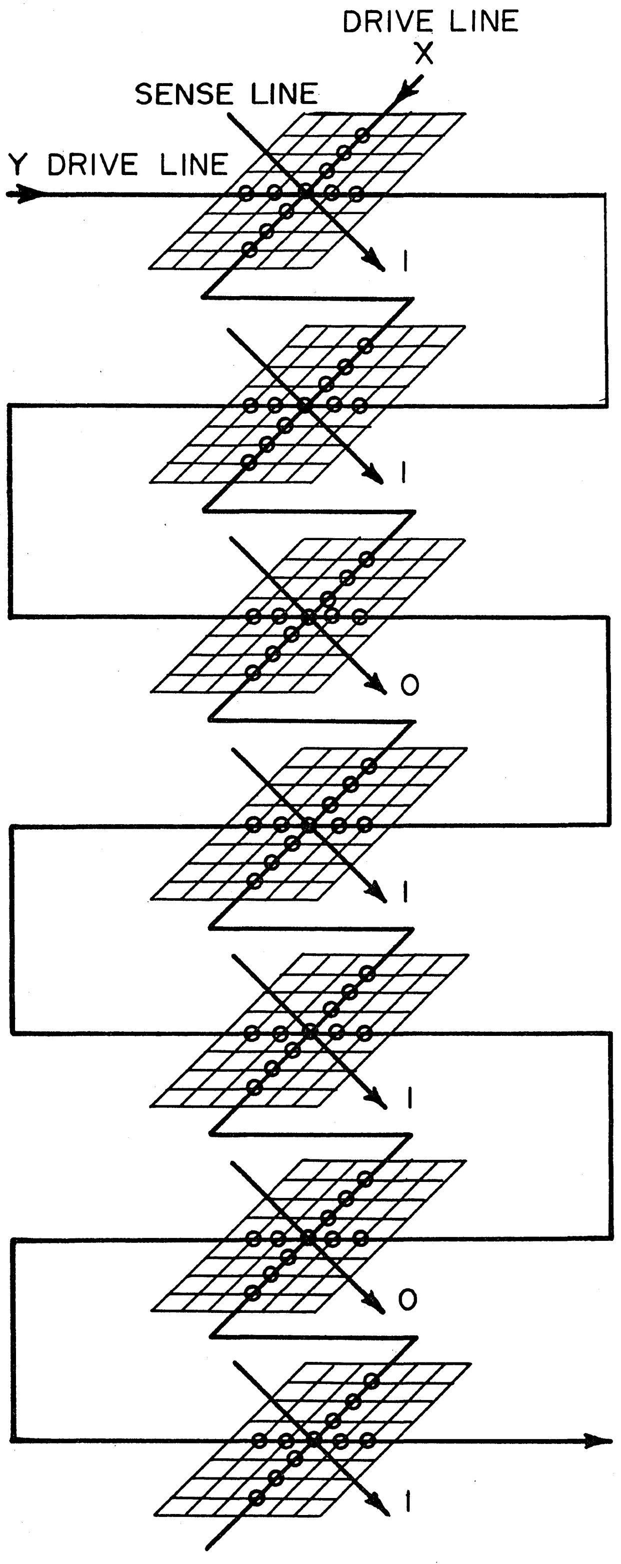
READ-OUT

Several memory core planes are illustrated in figure 6-6. It is assumed that the "X" and "Y" drive lines are pulsed to read-out data previously stored in the cores located at the intersection of the drive lines. The sense lines read the data from each of the cores simultaneously so that the entire output (top to bottom 1101101) is read in one bit time (parallel). Also, the data can be read serially—one bit of the word in each of seven bit times. In operation one memory plane exists for each bit in the word.

The sense output is amplified and sent to specified registers where it enters into certain arithmetic or other types of operations. Reviewing briefly, the drive pulse drives from a zero to a 1 during the write cycle and from a 1 to zero during the read cycle.

The X drive pulse and the Y drive pulse are each equal to 1/2 H_m. Two pulses are required simultaneously to either set or reset the core. This action is called the coincident pulse type of operation of magnetic core memories.

The selection of a particular Y drive line and X drive line will select a particular core and this in effect is its so called address. Reviewing briefly, the drive pulse drives from a zero to a 1 during the write cycle or from a 1 to zero during the read cycle.



124.59
Figure 6-6.—Memory read-out.

TERMINOLOGY

The use of magnetic cores as described provides high-speed, random-access, nonvolatile storage. A storage component is considered nonvolatile if it retains its binary state when all power is removed from the equipment. The term high-speed memory is defined relative to the time required to gain access to data in memory when other types of storage elements such as magnetic drums or tape are used. It may be defined in terms of the shortest access time of two or more systems which use the same type of storage element. In all cases, the term ''high-speed'' is relative.

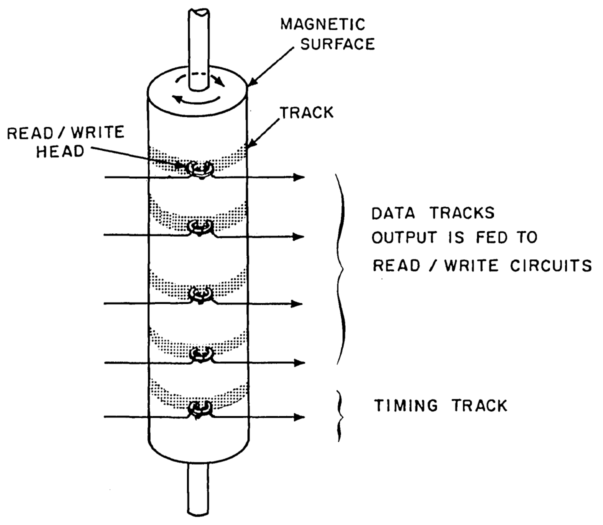
With some types of memory devices, the time required to read-out a certain bit of data may be considerably longer than the period required to read-out other data as mentioned earlier. This is particularly true where magnetic tapes or drums are used as the main storage medium. On the other hand, the use of magnetic cores as the basic memory components permits any data in memory to be read-out in approximately the same time. Thus, if two data words are randomly chosen, the access time (time required for read-out) for both words will be substantially the same. Because of this feature, magnetic-core storage may be referred to as random-access storage.

MAGNETIC DRUMS

Magnetic drums provide a relatively inexpensive method of storing large amounts of data. A magnetic drum (fig. 6-7) can be made by using either a hollow cylinder (thus the name ''drum'') or a solid cylinder. The cylinder can consist entirely of a magnetic alloy, or it can have such an alloy plated upon its surface. Many drums are made by spraying on a magnetite, such as iron oxide. The surface is then covered with a thin coat of lacquer, and buffed.

A representative drum has a diameter of 5 to 20 inches. The surface of the drum is divided into tracks or channels which encircle the drum. A number of read and write heads (at least one for each track) are used for recording and reading. The drum is rotated so that the heads are near but not touching the drum surface at all times.

Drum-driving motors (not shown) range between one-fifth of a horsepower to one and one-half horsepower. Larger drums are driven more slowly than smaller ones. Driving speeds



124.60
Figure 6-7.—Magnetic drum.

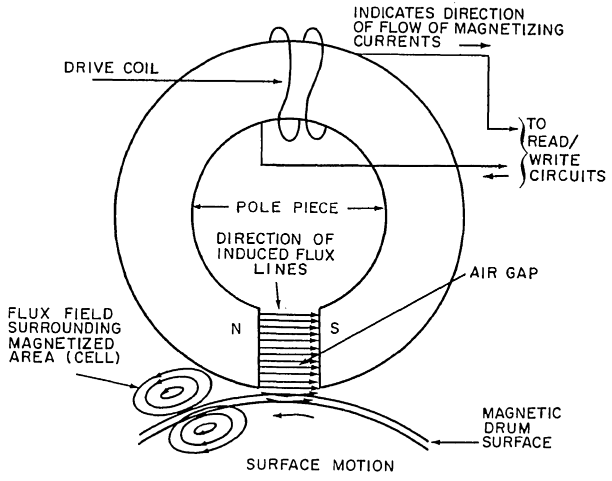
range between 120 and 20,000 rpm. Some drums weigh about 50 pounds, while others weigh well over 500 pounds. The access time decreases as the diameter is decreased and as the speed of the drum increases.

READ/WRITE HEADS

The read/write heads are placed about 0.002 inch from the surface of the drum. This spacing is very critical. Spacing changes, such as those produced by drum wobble, vary the reluctance of the path and hence cause variations in signal level. This can produce faulty data and can result in intermittent operation. In order to avoid this, the space between the drum and read/write head must be kept constant. Thus, the drum must be perfectly balanced and the bearings must permit very little wobble. In fact, if the drum ever makes contact with the read/write head, the drum will be grooved and ruined.

Drums usually have a few tracks that are reserved for identifying the data written across the length of the drum. These tracks hold drum addresses. The principle is explained later.

During the write process (fig. 6-8), currents through wires wound around a pole-piece induce small magnetic fields on the drum surface within the associated track. Each of these magnetized areas is called a "cell" and is capable of storing either a "1" or "0" state.


INDICATES DIRECTION OF FLOW OF MAGNETIZING CURRENTS
DRIVE COIL
TO READ/ WRITE CIRCUITS
POLE PIECE
DIRECTION OF INDUCED FLUX LINES
AIR GAP
FLUX FIELD SURROUNDING MAGNETIZED AREA (CELL)
N
S
MAGNETIC DRUM SURFACE
SURFACE MOTION

124.61
Figure 6-8.—Operation of read/write head.

The direction of current in the drive coil determines the polarity of the induced magnetic field.

The pole-piece is constructed of a high-permeability material which is capable of reversing its magnetic field quickly when a minimum amount of reverse driving force (measured in gilberts) is applied.

During a "write" operation the flux in the air gap passes through the high-retentivity magnetic surface and produces small magnetized spots on the drum surface. Thus, the magnetic field that existed during the instant when the magnetic surface passed the air gap causes a residual field to be stored on a small area of the magnetic surface.

During a surface magnetic reading operation the magnetized areas move past the air gap of the head. Some of the flux from the cell on the moving magnetic surface passes through the pole pieces and a low amplitude voltage is induced in the drive coil.

In some applications, separate read and write heads are used. When the head is used for both read and write operations, suitable external circuitry must be used to isolate the associated driving and reproducing circuits.

Each read/write head is connected in a separate circuit so that read or write operations can take place on any or all tracks simultaneously. The data which appears under all heads at the same time form a "slot."

In some applications, the magnetized spots may represent ones and unmagnetized spots zeros. In other applications ones and zeros are represented as magnetized spots of opposite polarity.

RETURN-TO-ZERO METHOD

In a recording system which uses a pulse to indicate a 1, and no pulse to represent 0, the pulse-no-pulse representation of the binary word 101101 would appear as shown in figure 6-9A. This method is called a positive-pulse return-to-zero system and is abbreviated RZ.

In some applications, the voltage does not return to zero after each bit but only when the data to be represented requires such a change. This method of representing data is called the "non-return-to-zero" method, and is illustrated in figure 6-9B.

NON-RETURN-TO-ZERO METHOD

The non-return-to-zero method is used where compactness of data is important. Note that in this method the change from 1 to 0 or from 0 to 1 requires only one change in voltage level. Using the return-to-zero method, each change from a 0 to 1 requires two changes in voltage level, i.e., from zero to maximum voltage and from maximum voltage to zero. Thus, more time is required using the return-to-zero method.

SERIAL READ-OUT

To further clarify this point, consider the action for serial reading of the data represented in figure 6-9C using the upper AND circuit and the clock pulses (B_1). The operation of the AND circuit to produce the output ($F=A_1 B_1$) is understood. Because each bit of data using this method (return-to-zero-method) rises from the zero level to maximum voltage and remains at that level for a given period (long enough to permit the clock pulse to be ANDed with each bit of data in coincidence), and return from the maximum level to zero, the time between clock pulses is necessarily long, and the overall time to read a single bit is long in relation to the actual AND operation, which, in itself, is the read process. The unique way in which the non-return-to-zero method lends itself to faster reading is seen by studying the waveforms in figure 6-9D and their actions to produce the output ($F=A_2 B_2$) from the lower AND circuit. Note that the output produced in the 5μs read period for the return-to-zero method is still 110111 (6 bits), whereas the non-return-to-zero output produced in the same period is 111100000011 (12 bits). Thus, serial data can be read in a shorter period using the non-return-to-zero method since the clock pulses can be applied at a higher clock rate.

WAVEFORMS

The pulses shown in figure 6-9E better illustrate the flux and read voltage. You will note that both a positive and negative voltage are induced in the read head each time the flux pulse stored on the magnetic surface passes the air gap. This occurs because the magnitude of the induced voltage is proportional to the rate of change $\frac{d\phi}{dt}$ of the flux and its polarity to the direction of the flux change.

If it is assumed that the flux rises as it approaches the read head, it follows that it falls as it leaves the head. This produces a read voltage output pulse of one polarity followed by a second pulse of opposite polarity. This condition is of little consequence since either of the pulses (the positive or negative) can be easily eliminated.

ADDRESSING

A timing track (fig. 6-7) on the drum contains a series of permanently recorded timing signals which are used to locate any drum slot. Some drums use two or even three timing tracks. The timing tracks are used for synchronization purposes and are sometimes called control or clock tracks.

A drum with a diameter of 20 inches would have a circumference of 62.8 inches. If the magnetic properties of the drum surface permit the storage of 300 bits per inch the total number of timing bits (using a single timing track) would number 188,400. Because bits can be stored on all tracks at the same density, 188,400 data bits can be stored in each of the four data tracks shown and the drum capacity is 753,600 bits.

SYNCHRONIZATION

In chapter 4, it is pointed out that all computer operations take place under the influence of the clock or timing pulses which are generated in a stabilized oscillator. It is again emphasized that the timing pulses establish the time scale

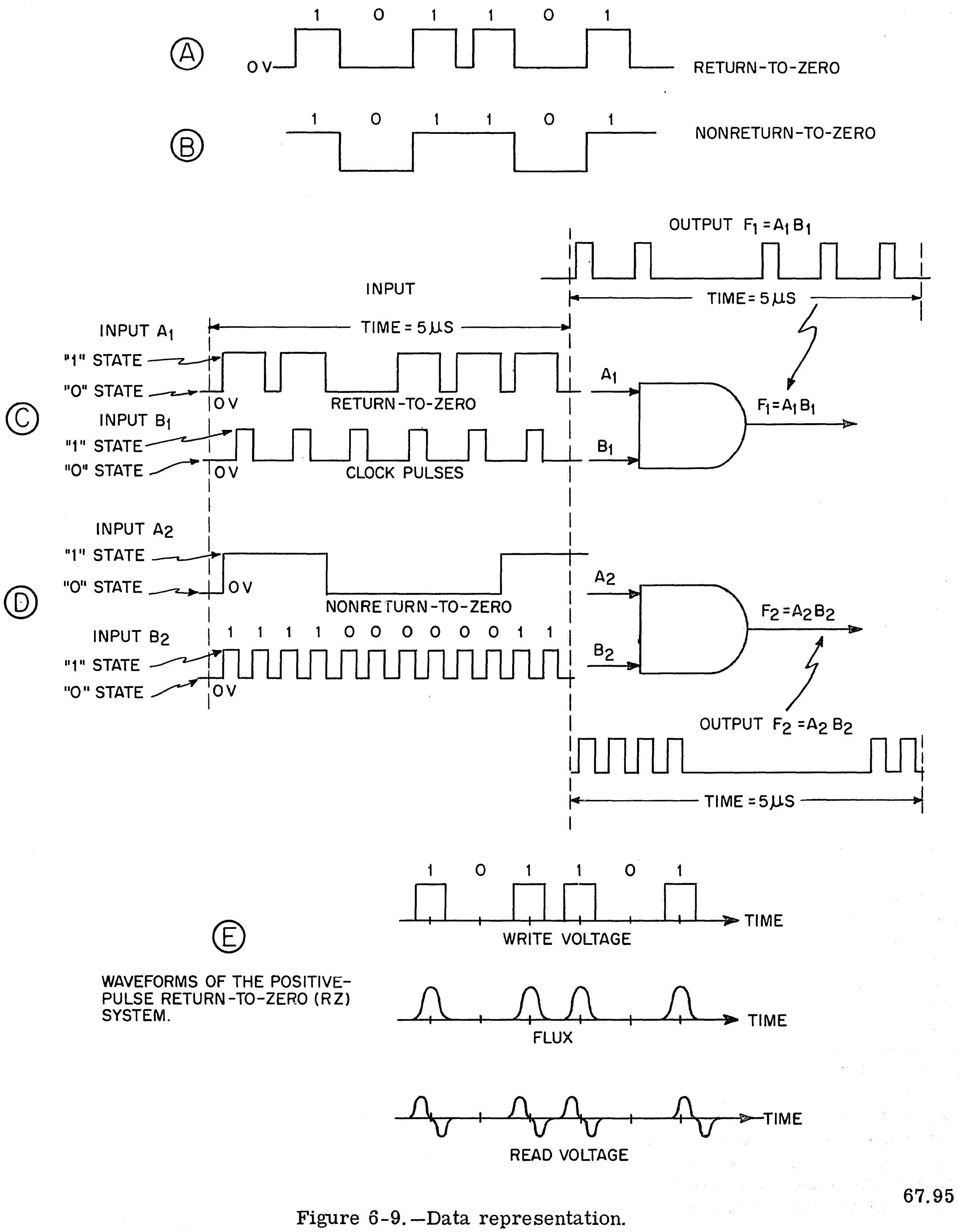


Figure 6-9.—Data representation.

to which all circuits throughout the computer are synchronized. When core memory is used all data is stored in the cores in a static condition and can be located at a given place at any instant and easily read from that location in serial or parallel form to represent the same data that was stored in that location.

Data stored on drums is in constant motion (dynamic). Transfer of data is therefore complicated.

Timing pulses are not used to synchronize the drum speed (which may vary slightly from time to time), and thus some method must be used to ensure that data read into the drum memory in a given bit position will be read from memory with the same time reference. (It is understood that in the case of dynamic storage if data read into memory is not read out with reference to the same time scale, the information read is of little or no significance.) The probability of an incompatible time relationship between the drum speed and synchronizing (clock) pulses makes it necessary to establish some means of compensating for variations in drum speeds.

DATA REPRESENTATION

Before studying the methods used to ensure accurate transfer of data from the drum, it is first necessary to understand clearly the method used to represent data and how clock pulses are used with data represented. To explain this we will consider a system which uses the return-to-zero method of pulse representation. Three methods of data transfer are represented in figure 6-10.

Each computer word contains digits, the number dependent upon design. In figure 6-10A it is shown that a word contains a number of bits, again, dependent on design. Once the bits-per-digit and digits per-word are established in a computer which uses a fixed word length, each computer word will contain the same number of bit positions. Thus, all words in a fixed word length computer are of constant duration.

Bit Parallel-Digit Parallel

The method of data transfer as illustrated in figure 6-10A is called the bit parallel-digit parallel or fully parallel method of transfer. With this method a word can be transferred (either recorded or written) in one bit-time. Obviously this method provides for fast transfer, but requires the use of separate lines (and associated circuitry) for each bit contained in the word.

Bit Parallel-Digit Serial

The method illustrated in figure 6-10B is called bit parallel-digit serial. To illustrate how this method of notation is derived, each bit has been assigned a number and it is understood that bits represented up and down the page are transferred simultaneously (four at a time) while bits represented across the page are transferred in sequential bit-times.

Bit Serial-Digit Serial

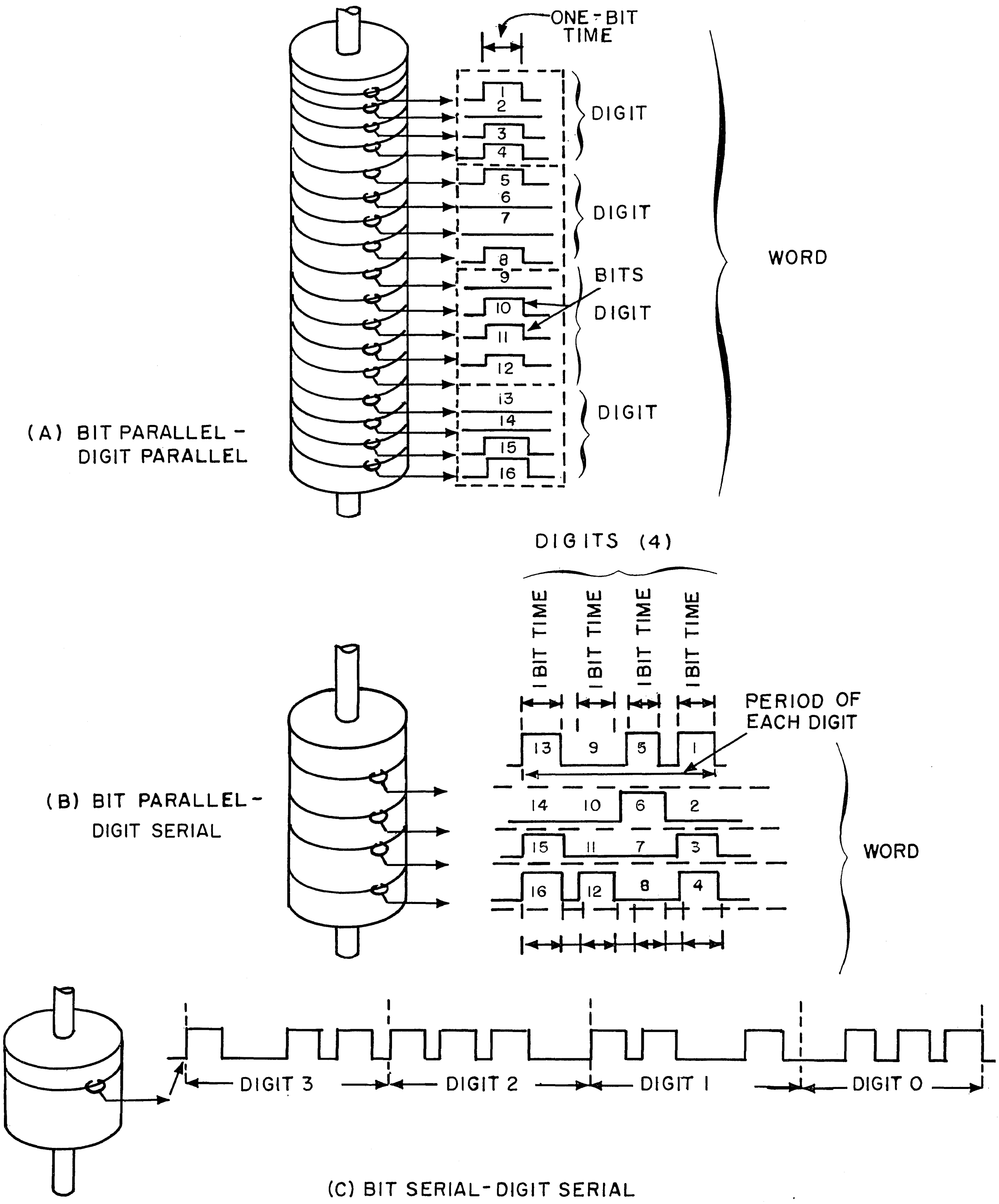
In the fully serial, or bit serial-digit serial, method of transfer, all bits in the word (and therefore all digits) are transferred sequentially. All bits are transferred over a single line and use a single transfer circuit. Although circuit requirements are simplified by using this method, more time is required to transfer a single word.

Because of time consideration as discussed above, fully serial computers are not widely used. In many cases, however, fundamental timing techniques used in computers (both serial and parallel) are sometimes more easily or more clearly explained assuming serial operation and using serial circuitry. For example, if it were considered necessary to explain in detail the operation of all circuits used in the transfer of bit parallel-digit parallel data, it would be simpler to explain the action by using a single circuit and a single (recurring) bit input.

TIMING

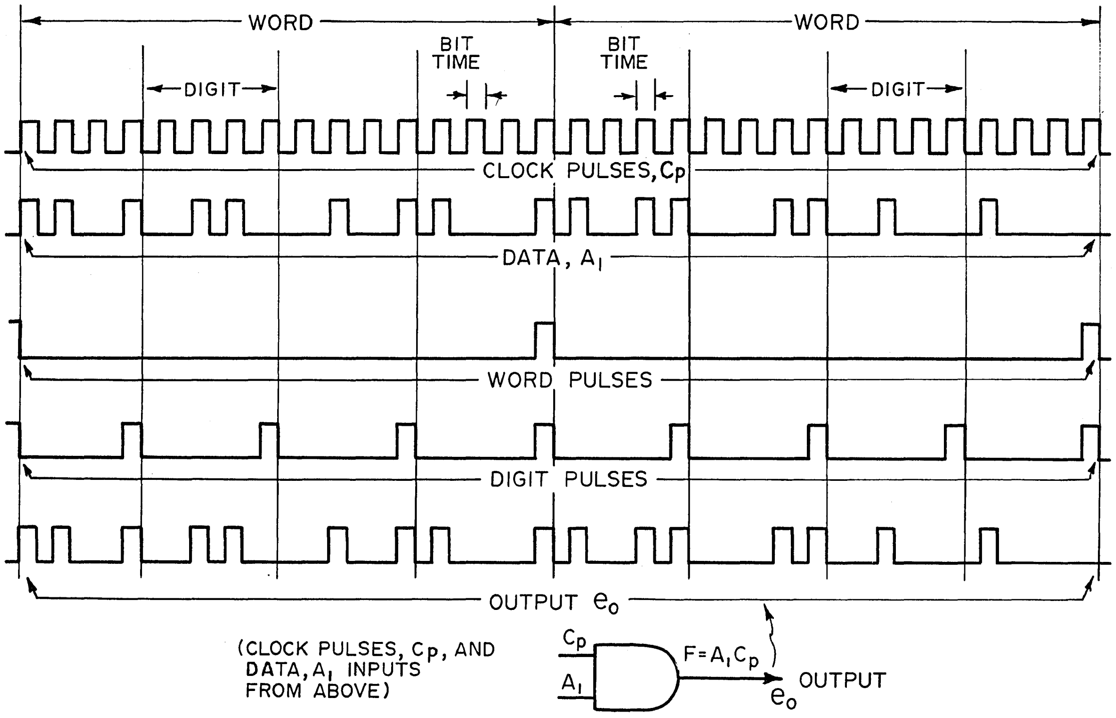
Now study the relationship (fig. 6-11A) between the various pulses that exist within the computer, particularly those used for control (clock) purposes. In this example, which shows the bit serial-digit serial method of transfer, the duration of a clock pulse is taken as a bit time.

Four bits make up one digit. Likewise, four digits make up one word. The choice of the number of bits and digits which comprise a word is arbitrary. In practice, a parity bit (discussed in chapter 3) is contained in each digit to detect the loss of a bit or errors resulting from inproper coding. Where sign-bit

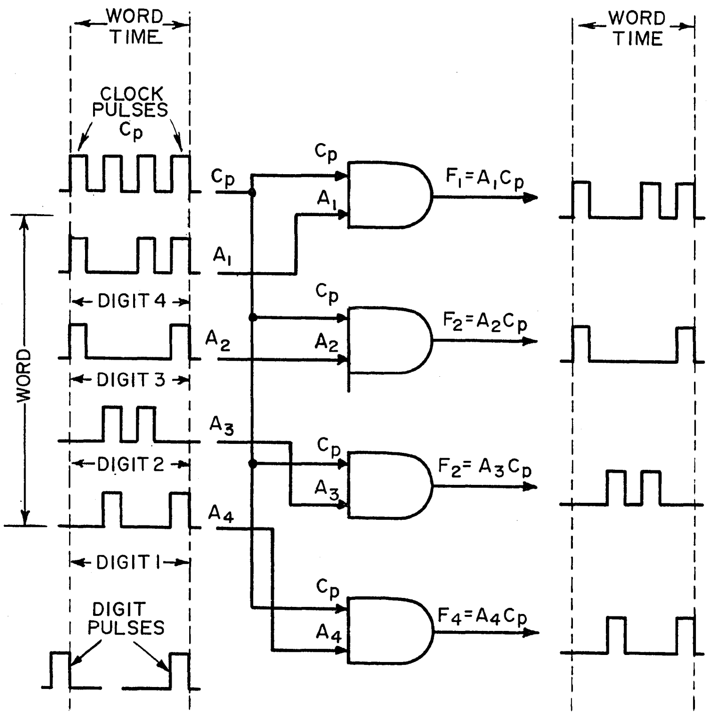


124.63

Figure 6-10.—Methods of data transfer.



A



B

124.64

Figure 6-11.—Timing.

notation is used, one or more of the bits contained in the word will be used to indicate the sign of the number represented by the numerical digits.

The AND element illustrates symbolically how the input data is transferred. The bit parallel-digit serial method of representing and transferring the same data is illustrated in figure 6-11B. Here, four AND circuits are used in the transfer even though the complete transfer takes place in one-fourth the time required for the bit serial-digit serial method (fig. 6-11A).

Now consider the process of data transfer when drums are used at the main storage. Particular attention is directed to the digit and word pulses. The series of clock pulses in figure 6-11A have little significance alone and will be of little value if applied to the AND element to cause the transfer of the data pulses (A_1) unless some other input is fed to the drum to cause the beginning of the transfer to correspond with a word-time and end of the transfer to occur at the termination of a word-time or a given number of word times.

In practice, the timing pulses which are permanently recorded on one or more timing tracks (as mentioned earlier) consist of word-time, digit-time, and bit-time pulses. With this arrangement, any changes in the drum speed will be accompanied by a corresponding change in the time between timing pulses.

During the recording (writing) process, each word placed on the drum is made to begin when a word pulse comes under the head of the timing track. Likewise, the read-out or transfer of a word from the drum occurs in synchronism with the timing pulses. Thus, by permanently recording the timing scale on the drum, the possibility of poor or jumbled read-out (caused by a change in the relationship of the basic timing signal at the time of read-out as compared to the time when the data was written) is eliminated.

SERIAL OPERATION

Each of the drum tracks is numbered and divided into sectors. Each sector stores one computer word. When using serial operation, only one track at a time is written upon or read from. The action takes place on a given track in a given sector. Because each sector is the same length as a computer word, a track which stores 768 bits will accommodate sixteen 48-bit words and is therefore divided into 16 sectors. In the serial method of drum operation, a specific address is referenced by indicating the track and sector numbers. For a 48-bit word, 12 bits may be used to specify the track number, 12 additional bits to indicate the sector number, and the remaining 24 bits may be used for data.

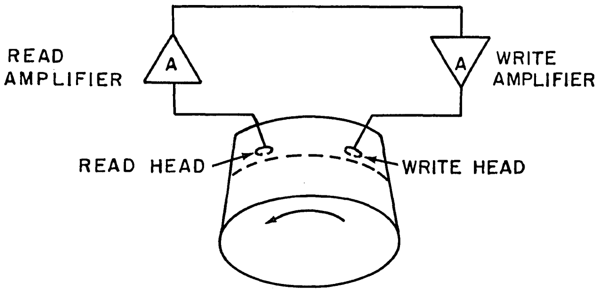
Use of Three Timing Tracks

Drums which use three timing tracks frequently use one track for bit pulses, one for word pulses and one for indicating the sector number. A given sector (on the sector track) contains the number of the next sector which will come under the read head. In this manner, once the proper track is selected and the sector determined, the read-out of the desired data from the proper sector can begin at the onset of a word pulse, by comparing a coded arrangement of sector pulses with those read from the drum. When the coded pulses (in static form) and the pulses read (dynamic pulses) coincide, a signal is issued to begin read-out at the time of the next word pulse. (In some computers a space bit is provided between words. The above example is not valid when this method is used.)

When serial operation is used in conjunction with separate read and write heads properly spaced near the surface of a drum, (fig. 6-12) the data written onto the drum is re-read some short time later (during a given drum revolution) and is constantly re-circulated. The drum surface is used as a delay medium. This arrangement is called a "dynamic register" since the data stored is constantly changing its position in the storage medium.

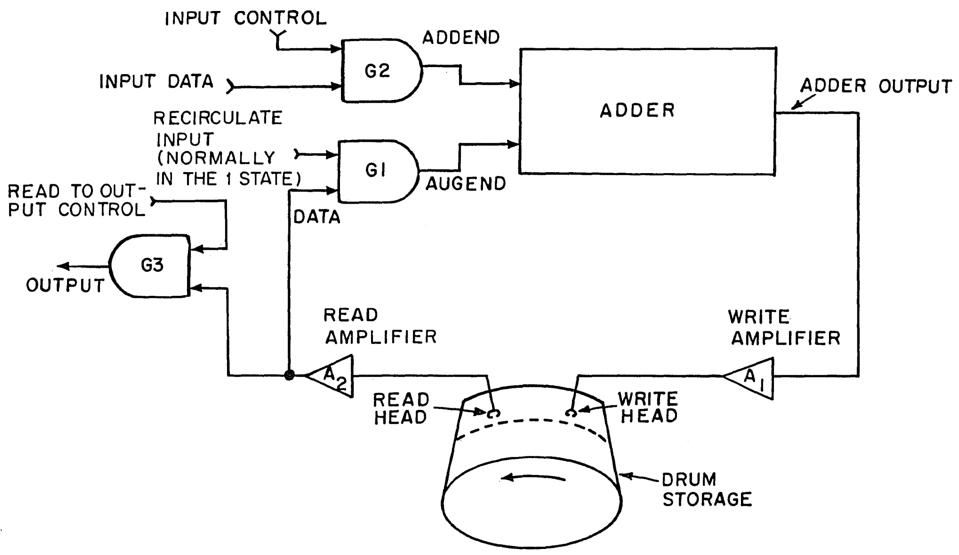
Dynamic Register

A dynamic register may be used as the accumulator as illustrated in figure 6-13.



124.65

Figure 6-12.—Dynamic register.



124.66

Figure 6-13.—Dynamic register used as the accumulator.

Here, it is assumed that data (to be used as the augend) is already circulating in the loop comprising the drum, A2, G1, the ADDER circuit, and A1. This data continues to re-circulate (without alternation) as long as two conditions exist: (1) the re-circulate input to G1 must be maintained in the high or "1" condition, and (2) the signals to the input control and input data terminal of G2 must not permit an addend input to the adder. The ADDER circuit is not explained in detail here. It is sufficient to say that the ADDER will permit the augend data to pass to its output unchanged if the addend input is maintained in the "0" state.

Assuming the re-circulation loop to be enabled, if the input control and input data terminals to G2 are in the state which permit an addend input to the ADDER, the re-circulating data (augend) and the addend (the new input data) are combined in the ADDER and the output is the sum, or difference of the two inputs. (The output must be interpreted with regard to the type of addition which takes place in the ADDER and whether the addend represents the complement of some number.) The output of the ADDER becomes the new contents of the register.

If it is assumed that a series of addend numbers (each one word in length) is applied to the ADDER, and that the result of each previous addition is permitted to re-circulate, the number stored on the drum at the end of the series is an accumulation of sums and the register is appropriately called the "accumulator." (The circulation loop, as described previously, comprises the drum, A2, G1, the ADDER circuit and A1 in, figure 6-13.)

Clearing the register is accomplished by placing a "0" input on the upper or recirculate terminal of G1. This action prevents the old or stored data read from the drum via A2 from passing through G1.

In many cases, it is desired to store new data on the drum while clearing the old. This is accomplished by enabling the input control terminal of G2, disabling (reducing the re-circulate input to the "0" state), and applying the new data to the input data terminal of G2. The new data is passed along the addend line to the ADDER. The ADDER circuit, by design, will permit the addend to pass to the ADDER circuit output unchanged. The data is subsequently passed through write amplifier A1 to the write head, where it creates the necessary flux about the write head to cause storage of the new data on the drum. Thus, clearing or erasing old data from the drum and recording or writing new data on the drum are accomplished simultaneously.

MAGNETIC TAPES

A third type of storage device is magnetic tape (similar to the tape used with commercial tape recorders) which is used mainly for secondary storage, i.e., it is used for storing reference data and to provide large capacity, long-access time storage. The magnetic surface of tapes usually consists of a thin coating of red or black iron oxide on a metal or plastic backing. Tapes are approximately 0.0022 inch thick and range in length up to 2400 feet. Most computer tapes use channels or "tracks" along the length of the tape. A separate read-write head is used for each track, so that a number of bits can be written or read simultaneously.

One type of tape used in digital computers has seven tracks, and is thus capable of reading or writing seven bits at one time. The seven bits across the width of the tape is referred to as one "column" of data. A spacing is automatically provided between columns.

Tape travels across the read-write heads at speeds ranging from 50 to 200 inches per second. If it is assumed that a tape travels across the heads at a speed of 75 in per sec, data can be recorded at a rate of 18,600 bits per second, one bit every 53.8 microseconds. This means that $\frac{18,600}{75}$ or 248 bits are written on each inch of a particular track (speed 75 inches per second), and that 248 x 7 or 1,736 bits are written on each inch of the seven-track tape. The number of bits which can be recorded per inch of tape is referred to as the "bit-density." Of course, there are spaces on a tape where no data is stored. These spaces are used to separate the "words" of one group of stored data from another. These groups of data are often called "records" or "blocks." A single "block" or "record" of data is separated by an inter-record gap before and after each "block" of data.

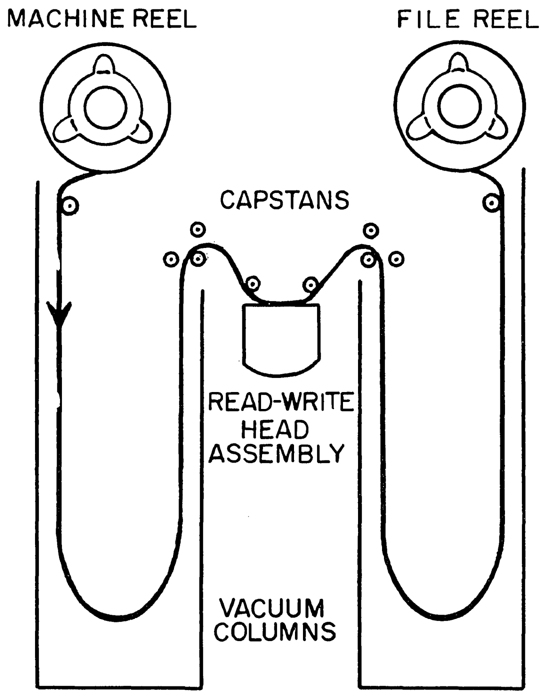
The spaces between records correspond to the amount of tape that passes under the read-write head while the tape drive is coming to a stop at the end of each writing operation. An inter-record gap followed by a single character is used to indicate the end-of-file (end of tape). The "end-of-file" space is sometimes a reflective material on the tape which ranges in length from one tape to another from 3/8 inch to 3 3/4 inches.

A 2400-foot reel of seven-track tape can store all of the information contained on approximately 20,000 eighty-column punched cards. Newer tapes have "bit densities" of the order of 800 bits per inch and hence larger capacities.

TAPE DRIVING MECHANISM

The basic tape-driving mechanism (fig. 6-14), consists of a group of motors, magnetic clutches, linkages, and capstans, (rubber rollers with metal hubs). These components are used in conjunction with electrical control circuits to stop, start, rewind, and drive the tape past the read-write head assembly.

Two reels are used. The machine reel holds the tape and the file reel pulls in the tape after it has been written upon or read. This arrangement is similar to the one that is used with rolls of film on a motion picture projector. Tape motion is produced by pressing the tape against a motor-driven capstan. But using solenoid-operated mechanical linkages, it is possible to press the tape against either the forward-driving capstan, the reverse-driving capstan, or a nonrotating stopping capstan.



49.162
Figure 6-14.—The basic tape drive mechanism.

The mechanical inertia of the reels imposes limitations on the starting and stopping of the tape. In order to solve this difficulty, a large loop of tape is used on both sides of the read-write head assembly. These loops act as buffers or overflow banks to minimize stresses on the tape. When the tape is started, tape can be drawn out of one bank by the drive capstan, fed through the read-write head assembly, and added to the loop on the other side. Meanwhile, the reels can be accelerating up to full speed in order to wind out more tape into one bank and to take up slack from the other bank. In this manner, tape passes the read-write head assembly at the correct speed during the time that the tape reels are accelerating or decelerating.

Vacuum columns are used to hold the loops of tape that are used as banks, and to keep the tape taut so that it does not buckle at the read-write head during the starting or stopping of the tape reels. The vacuum columns are vertical containers with rectangular cross sections. Air is pumped out of the bottom of each column. As a result, the pressure of the air on the inside surface of the loop keeps tape taut and reduces air friction.

TAPE CODE AND ERROR DETECTION

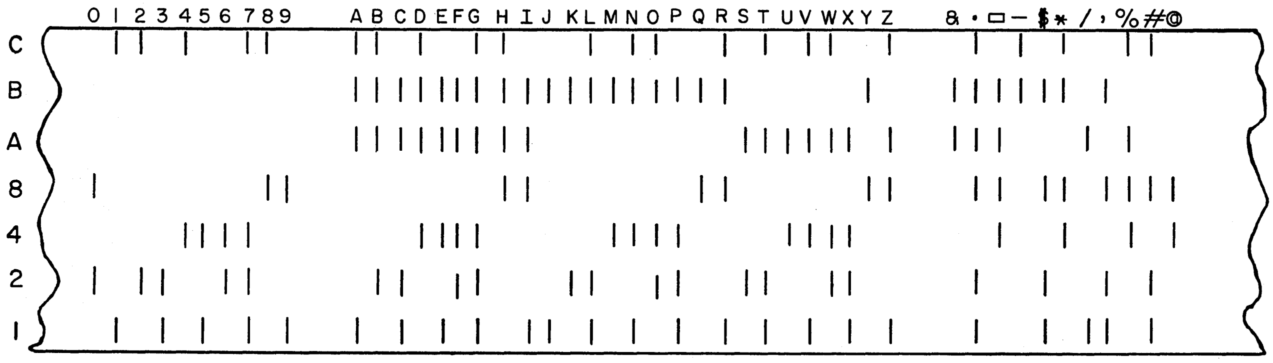
The seven-channel tape code (an IBM code) is illustrated in figure 6-15. Some special characters are shown. Because the code is used to represent both alphabets and numerals, it is called a "seven-bit Alphanumeric" code.

The channels or tracks are referred to as C,B,A,8,4,2, and 1. The seven-bit code is popular mainly because it is a wide tape, yet it can be handled easily. Wider tapes would require drive mechanism with greater mechanical abilities. This would create new problems in starting and stopping the mechanisms.

Some tape systems use a single read-write head for each of the seven channels while others use read and write heads separately. As can be seen in figure 6-15, the "C" bit for any column is 1 whenever the total number of 1's represented in the B,A,8,4,2, and 1 tracks for that column is odd. If any vertical column contains an odd number of 1's when the "C" bit is included, an error will be indicated.

As each record is written, the number of 1 bits in each of the seven (longitudinal) channels is recorded. At the end of each record, a bit is added to all tracks which contain an odd number of 1 bits. When reading the tape, an odd number of bits in any one of the seven tracks of any record indicates an error. Thus, checks are made across both vertical and horizontal (or longtitudinal) dimensions of the tape to ensure accuracy.

Tapes which use a single read-write head cannot detect errors during a continuous recording process. A reading of the tape after recording will reveal the presence of errors, if, in fact, errors exists. Tapes which use separate read and write heads can perform a read operation (for error detection) as the data is being recorded. Data recorded on the tape at the read head is written (or checked) an instant later at the write head. The use of separate read and write heads is therefore faster, since errors are almost immediately detected and may be corrected before the total recording process is completed. Similar accuracy or validity checks are made each time the tape is read.



124.68

Figure 6-15.—Seven-bit alphanumeric code.

Another common method of using the seven-bit code involves the use of six bits to represent two octal digits, such as 111_8 011_8 (73_8) and the seventh bit for parity checking. This method is used for representing numbers only.

MAGNETIC DISK

Magnetic disk memory involves the use of a number of iron-oxide coated disks (which resemble phonograph records) arranged in much the same way as a record stack in a modern "juke box." All of the disks are continuously revolving and spaced apart so that a recording head which is driven by an access mechanism can be positioned between the disks.

Data is recorded at a certain address on a specified disk. When read-out of a particular bit of data is desired, the recording head is automatically positioned and the data is read "serially" from the surface of the selected disk. Air ejected from the recording head onto the disk creates an upward draft which maintains a given spacing of the head from the disk.

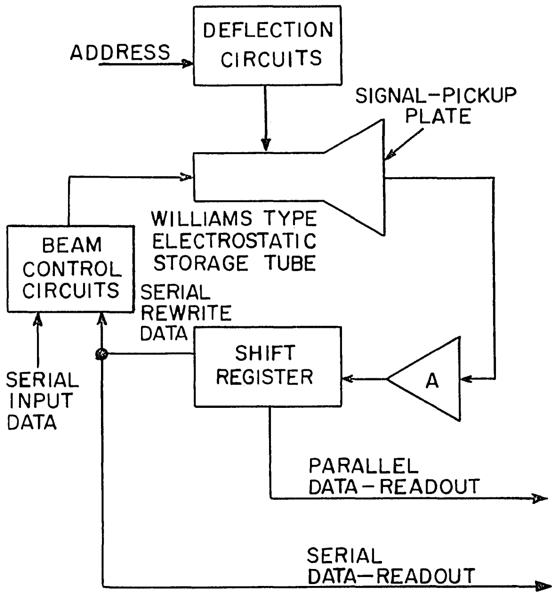
The main advantage of magnetic disk storage is its high storage capacity obtainable from a bank arrangement which contains several disks. The access time for magnetic disks memory banks is generally less than that for magnetic drums. Some magnetic disk systems can store up to 5,000,000 coded digits.

ELECTROSTATIC STORAGE

Electrostatic storage, as performed with the aid of cathode-ray type tubes and associated circuitry, can provide a memory with access time of the order of a few microseconds and storage capacities of about 1,000 bits per linear inch.

WILLIAMS TUBE

A representative tube that is employed for this purpose is the Williams tube. The Williams tube is used in a system arrangement similar to the one shown in figure 6-16.



124.69

Figure 6-16.—Block diagram of electrostatic storage using a Williams tube.

Construction

The overall construction and deflection system of the Williams tube is very similar to that of the familiar television picture tube. However, the tube possesses two additional features. The first is that poor conductor material, instead of the usual fluorescent material, is used to coat the inside of the face of the tube. The second feature is that a metal "signal-pickup" plate is placed in contact with the outside of the glass face of the tube. Consequently, a CAPACITOR is formed, consisting of the inside coating, the glass face, and the signal-pickup plate.

This capacitor is charged and discharged by the electron beam within the tube. Let us first consider how the charging operation is accomplished. If the electron beam is greatly accelerated by a high potential at the accelerating anode of the tube, electrons in the interior coating are "knocked" loose by the impact of the beam electrons. This phenomenon is known as secondary emission. The interior coating acquires a positive charge as the liberated electrons are drawn away to a nearby, positively charged anode. Since this coating consists of a poor conductor material, the positive charge remains within the very small area of beam impact.

Writing

By gating the electron beam on and off (by means of the control grid) as it is swept across the face of the tube, it is possible to store charges that represent the value of voltage at different instants of time. This procedure produces "writing." If the gating voltage that is applied to the control grid is a serially applied binary word, the word will be stored in the form of charged areas across the face of the tube. These areas represent charges on many individual tiny capacitors that share one common electrode which is the signal-pickup plate.

Reading

Reading (capacitor discharging) is accomplished by using an electron beam sweep voltage which is not gated on and off (as was true in the writing process). The discharge path of each one of the individual capacitors (charged areas) is through the electron beam.

Note that there is a relationship between time, which is provided by the beam sweep, and the physical location of the individual capacitors.

As the beam sweeps across the tube face, the flow of each pulse of discharge current through the signal-pickup plate indicates the presence of a stored charge within a given area.

The series of pulses thus produced is usually applied to a shift register (not shown). The shift register shifts at a repetition rate that is controlled by the tube sweep voltage. Consequently, if NO PULSE appears at the instant corresponding to a bit-position in the original stored binary word, a binary zero is shifted into the register. After all of the pulses have been collected, the original binary word (as it had been stored in the tube) is now present in the shift register, and is ready for any subsequent parallel or serial readout and use. Readout from the register involves a nondestructive sampling of the voltage levels contained in the register. Thus, readout of a given bit of data can be repeated as many times as necessary.

Readout of data from the Williams tube is destructive and the stored charge is "destroyed" during the readout process. This data can be restored by writing it back into the tube. This is accomplished by feeding the contents of the shift register through the beam-control circuits to the control grid of the Williams tube in order to repeat the writing operation.

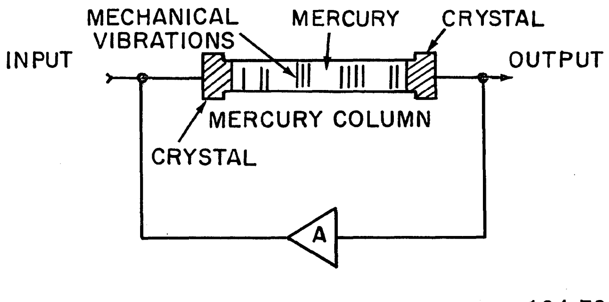
Computer words are written across the screen at a density up to 1000 bits per inch. Selection of a given location on the face of the tube for either reading or writing requires accurate deflection of the electron beam. The location of each word is therefore identified by an "address": in terms of its X and Y coordinates. This information which is digital, is converted into an analog voltage to permit the beam to be directed to any spot on the screen.

COMPARISON OF ELECTROSTATIC AND MAGNETIC CORE STORAGE SYSTEMS

When comparing electrostatic and magnetic storage systems, the time required to select a given address (access time) is slightly greater using the electrostatic storage method than for magnetic core memories due to the increase in time necessary to produce a voltage of the accuracy required to locate the desired data. Further, once data is stored in cores, it is seldom (if ever) necessary under normal operating conditions to re-record the data lost as a result of leakage. The retentivity of the core stores the data indefinitely. Using the Williams tube (electrostatic storage), data stored must be periodically re-recorded to compensate for leakages in the glass dielectric between the insulation material coated on the tube face and the metal pickup plate outside of the tube.

ACOUSTIC DELAY LINE

One of the earliest methods of data storage is the acoustic delay line. One form of acoustic line is the mercury column (fig. 6-17). The mercury column (sometimes called a mercury delay line) consists of a tubular mercury column and two piezoelectric crystals, one mounted at each end of the column. A pulse or group of pulses representing data is applied to one crystal which transfers mechanical



124.70
Figure 6-17.—Mercury column storage.

vibrations to the mercury column. These vibrations travel through the mercury to the receiving crystal at the other end. Here, the receiving crystal translates the vibrations back into electrical pulses which are amplified and returned to the input crystal or gated to output circuits. Data in the loop is made to circulate in the loop as long as is necessary. Thus, the mercury column (and its associated amplifier) forms a loop which can store its input data.

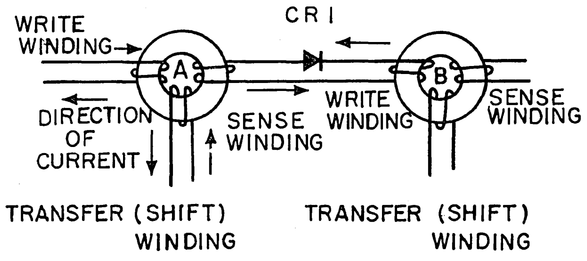
As the length of the mercury column is increased in an effort to store larger numbers, a stable column temperature (which is required in order to control the velocity of the pulse through the mercury) becomes more difficult to achieve. Likewise, the longer column the longer the access time, i.e., the longer the time required by the computer to locate and transfer information to or from storage. These factors limit the use of mercury columns as storage devices.

OTHER USES OF MAGNETIC CORES

Aside from receiving wide usage as the basic components of the memory section of computers, magnetic cores are used in other applications, such as ring counters, shifters, and circuits used for data transfer.

CORE TO CORE TRANSFER

In the following discussion of a core-to-core transfer (fig. 6-18), actual core windings are represented, rather than single wires, in order to simplify the explanation. It is assumed that core A is already in the "1" state and that this 1 is to be transferred to core B which is in the "0" state.



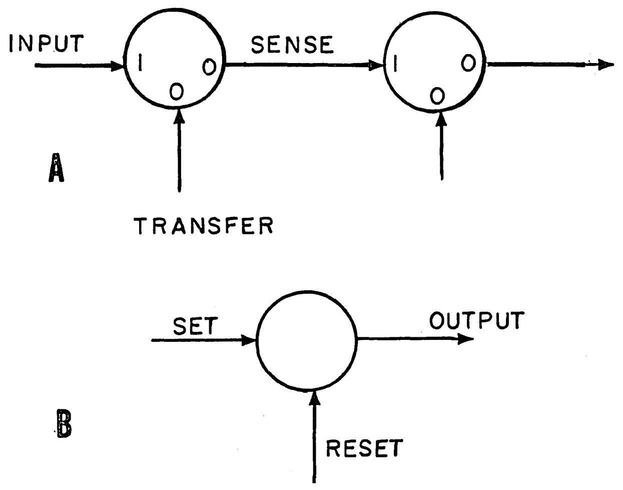
124.71
Figure 6-18.—Core-to-core transfer.

An input pulse on the transfer winding (in the absence of a pulse on the write winding) clears core A to zero. The polarity of the voltage induced in the sense winding (by the shift in the condition of core A) is correct (plus toward the anode of CR1 and minus toward the cathode) to forward bias CR1 and a current flows in the write (on input) winding of core B. This action switches core B to the 1 state.

A subsequent write pulse input to core A will restore this core to the 1 condition, but will cause an induced voltage in the sense winding of the opposite polarity to that described above and CR1 will be reversed biased. The high back resistance of the diode prevents core B from switching back to the "0" state.

CORE SYMBOLOGY

Cores are sometimes represented in shifting (transfer) circuits and flip-flops as shown in diagrams (A) and (B) of figure 6-19. The core is represented by the circle. The 1's and 0's inside the circle in figure 6-19A signify the action which takes place within the core structure when a drive pulse appears on the associated input. For example, if a pulse is applied at the input line at A, a 1 condition will be established in the core. If a drive pulse is applied to the transfer (shift) line, the core is returned to the "0" state. Finally, the "0" at the output (sense) terminal indicates that a pulse appears in the output line when the core switches from 1 to 0.



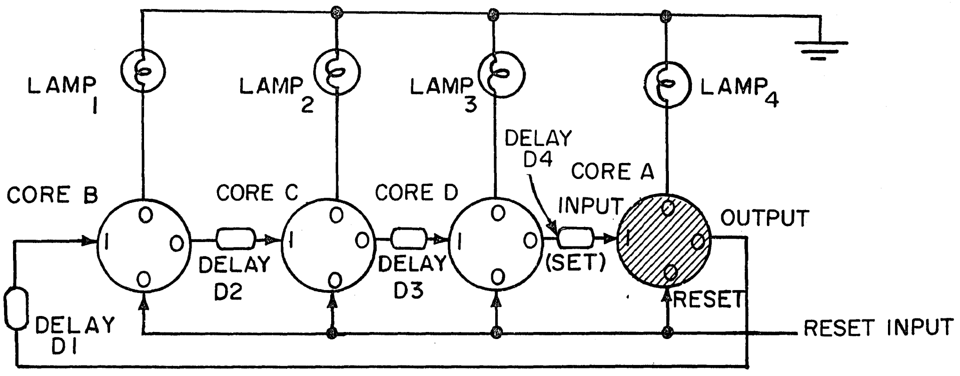
124.72
Figure 6-19.—Core symbology.

Magnetic cores are conveniently used as flip-flops (fig. 6-19B) if the 1 condition of the core (magnetized) is taken as the SET condition of the flip-flop and the "0" (demagnetized) condition is taken as the RESET condition. Cores used as flip-flops use the familiar SET, and RESET terminals and an OUTPUT terminal. An input pulse on the SET terminal places the core, (and the OUTPUT terminal) in the SET or 1 condition. A RESET input clears the core to 0 and the output is zero.

RING COUNTER

Because cores can be used as flip-flops, it follows that cores can also be formed into storage and shift registers and counters. A magnetic core ring counter is shown in figure 6-20.

If it is assumed that the shaded core (core A) is in the 1 state, lamp 4 will be "on." The first reset input will switch core A to zero extinguishing lamp 4, and produce an output which after delay in D1 is fed to core B. This core, in turn, switches to 1 and lamp 1 goes "on." The next reset input pulse will switch core B to zero and produce an input to D2. After delay in D2, an input is fed to core C causing this core to switch to the 1 state. The action continues to count 4 (the module of this particular counter) whereupon the count process is repeated.



124.73

Figure 6-20.—Magnetic core ring counter.

CHAPTER 7

INPUT/OUTPUT DEVICES

Input and output devices provide the computer with a communications link to the outside world. The input units supply data to the computer, and output units print the final data or cause it to be displayed at the output.

Conventional input devices read coded data into computers from punched cards or punched paper type (by holes punched in various positions of the cards or tape to represent data), or from magnetic tape (by magnetized areas on the tape). In some special military applications, the computer input is received from special purpose devices such as radar sets, gun platforms, missile-guidance systems, or tactical display consoles. In scientific digital computers, the input device may consist of a keyboard, while the output device may consist of a plotting board or an electric typewriter.

Data may be presented at the output in printed form (English or numerals), in plotted form (such as maps and graphs), on punched cards, paper tapes, magnetic tapes or oscilloscopic displays. Outputs in still other forms are available for special applications.

A computer must perform many repetitive arithmetic operations. In early computers the speed with which these operations could be performed was limited by the speed of the relatively slow electromechanical input-output units.

The problem then was to develop a method of feeding data into and reading data out of the computer at speeds compatible with the speed of the computer's internal operations. This is difficult to accomplish. Thus, two methods were developed to ease the problem. The first method, which is in wide usage today, is to program the computer so that it performs other internal operations on the available data while additional data is being fed in. This, however, is actually only a partial solution to the problem since input devices must be used directly to read in the initial programmed instructions. After feeding in the initial instructions, further instructions may be ''boot-strapped'' (discussed in chapter 2) via a buffer unit, thereby releasing the other sections of the computer for use in other operations. Only after all instructions have been read into and stored in memory can this method be effective.

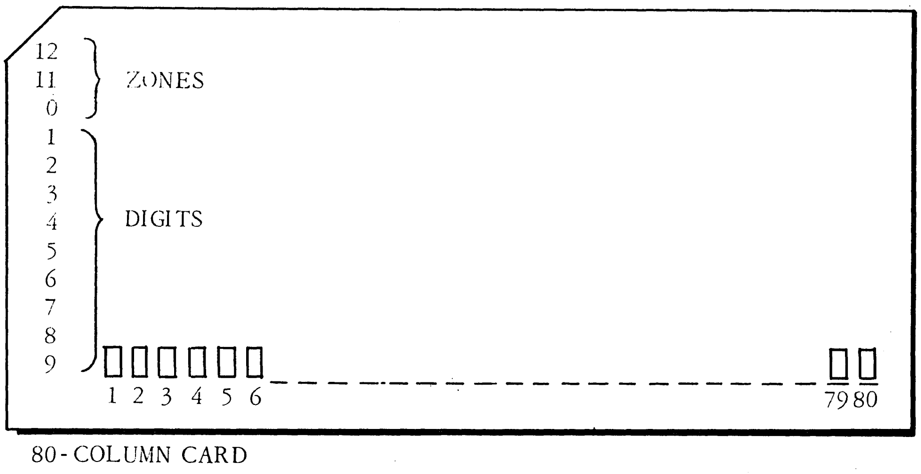
A second method uses external devices (which are designed to operate at slower speeds) to perform many of the menial data processing and handling operations. The processed and condensed data from the external devices are then transcribed to magnetic tape (one of the fastest input/output media available) so that the maximum computer input speed can be attained. When using this method, the computer can perform other jobs while the input data for one job is being prepared externally. This method can also be used for output data acquisition where the output data is written on one tape unit and then disconnected from the computer. The data can then be transcribed more slowly into other data forms (such as punched cards or punched paper tape) without slowing up the computer.

Input/output devices (such as paper tape or punched card machines) which are operated by computer control are called ''on-line'' input/output or peripheral equipments. Peripheral devices which are not controlled by the computer and which do not feed their data directly into the computer are called ''off-line'' peripheral equipments.

It is possible to have more than one tape unit reading into or out of the computer at the same time. When using this procedure, separate input/output channels must be provided to and from memory, along with separate control circuits. The number of input/output devices used with a given computer is determined by the needs and/or applications of that computer.

CARD-HANDLING EQUIPMENT

Thin punch cards (fig. 7-1) are used extensively for keeping records, storing data, and



124.74

Figure 7-1.—Sample punched card.

computing. Small holes are punched into the cards to represent (by their relative positions) numbers, letters of the alphabet, and various auxiliary symbols. You will recall that combinations of letters and numbers are referred to as alphanumeric characters. The card code is explained later. Card-handling machines are used to process these 3 1/4 by 7 3/8-inch cards.

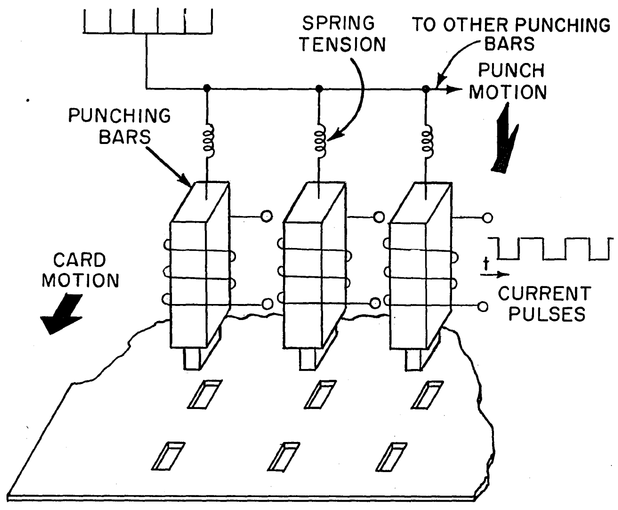
Punched cards are designed to accommodate 80 and 90 alphanumeric characters respectively. Card-handling machines which process these cards can also be used to compute data as well as to transfer data carried on cards to some other medium. Some small-scale computers have been made by combining appropriate card-handling units.

Some conventional card-handling units in use are card punches, readers, sorters, card converters (or translators), and verifiers.

METHODS OF WRITING AND READING WITH CARDS

The actual operation of writing on a card is accomplished by means of a number of solenoid-driven punching bars or rods, as shown in figure 7-2. The number and locations of the punching bars or rods that are actuated for any single punching operation are determined by the number or letter that is being recorded.

The standard 80-column card uses various combinations of punched holes in a maximum of 12 rows and 80 columns for coding any number from 0 to 9, any letters from A to Z, or any one of an assortment of special symbols. Thus, a card puncher used for IBM cards actuates one or more of 12 punch bars for each



124.75

Figure 7-2.—Card punching mechanism.

character. The single-character column array of punches is often called a punching station. The outstanding features of the coding that is used to indicate the various characters will be described later.

One method of card reading consists of passing a card between a contact plate and spring-mounted electrical contacting fingers or brushes. If a hole is punched in the card, the brush makes contact with the plate underneath, as shown in figure 7-3. As the cards move past a column of contacts (often called a reading station), the contacts close a selective toggle switch each time a hole is encountered. Thus the switch closes when a hole is present, and opens when no hole is present. Card-equipment devices use this open-close indication to actuate other control and data-processing circuitry. There are as many contacts as there are code areas in a single column on the card being read.

Another method of reading (fig. 7-4) uses the card as an optical mask that is located between a source of light and a matrix of photoelectric elements. This is the more rapid of the two methods. When a hole is encountered, light passes through the hole and energizes the circuits to interpret this hole as a 1 condition. The absence of a hole prevents light from passing, and no output is produced. This causes a 0 condition to be read from the card.

Notice that with both methods, the operation of reading is accomplished by means of selective on-off indications.

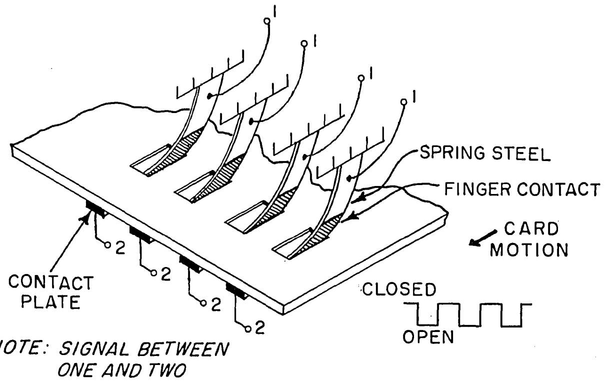
Card Codes

A widely used card code for military applications is the Hollerith Code, illustrated in figure 7-5. A table of this code is shown in figure 7-6.

The first, second, and third longitudinal rows from the top of the card are called the 12's, 11's, and 0 zones respectively. The remaining rows, numbered 1 thru 9, are called the numeric or digit rows. (The 0 row is sometimes referred to as a numeric row rather than a zone. It is treated as a zone in the following discussion.)

It is seen that an "A" is represented by a hole in the 12 zone and a hole in the 1 digit place. An "N" is represented by a hole in the 11 zone and a hole in the 5 digit place. A "Y" is represented by a hole in the "0" zone and a hole in the 8 digit place. A study of the card and the table in figure 7-6 will reveal the code for all letters, numbers, and symbols used.

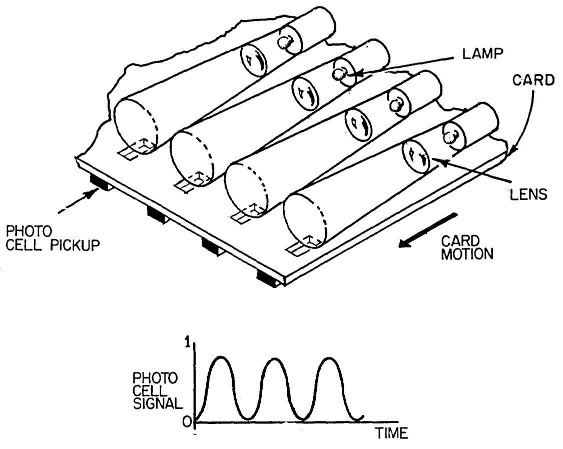
A computer which receives data directly from cards has translating matrices that convert the card-reader signals as required for use in the computer.



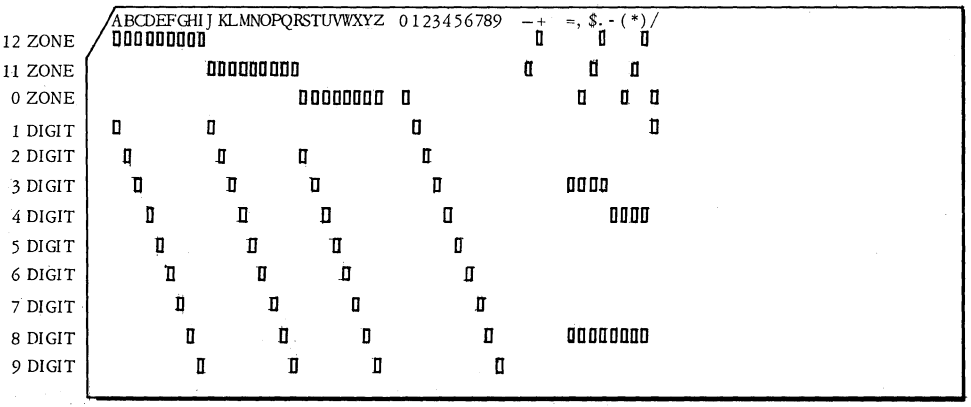
124.76
Figure 7-3.—Punched card reading.

Card Punching

Card punches are actuated by a manual keyboard, or in some instances by the computer. The keyboard buttons, shown in figure 7-7, are punched to open and close groups of switches (not shown) which apply current pulses to



124.77
Figure 7-4.—Photoelectric card reading.



124.78

Figure 7-5.—Hollerith Card Code (sample card).

solenoid-operated punches. Once the holes have been punched in one column, the card punch automatically advances the card so that the next column is located under the punching station punches.

One type of card punch machine is the IBM Type 24 Card Punch, shown in figure 7-7. The Type 24 punch consists of six main operating sections. They are: (1) the keyboard, (2) the program unit, (3) the card hopper, (4) the punching station, (5) the reading station, and (6) the card stacker.

In operation, a "program card" (which has been previously punched) is wrapped around a drum which is a part of the program unit of the card punch. A fixed reading station reads each column of the program card as the drum is rotated. Punched holes in certain zones and digits in the program card cause certain columns or number of columns to be skipped on the card being punched. The program card also causes automatic duplication of material that has already been punched in the same column of the preceding card. Notice that using a program card in this manner reduces the manual effort that is required during repetitive card punching.

The machine is also used to coordinate the timing of the various mechanical operations of the card punch. These mechanical operations include feeding the next card from the hopper (the holder for unpunched cards) to the punching station and advancing the card, column by column, under the punches. Cam-driven switches (not shown) are used to generate stepping pulses. The stepping pulses actuate ring counters or stepping switches which apply timing pulses to the various electro-mechanical devices that are used to perform the mechanical operations.

The keyboard is a typewriter-like set of manually operated keys. It is used to provide the actuating signals to the punches. The card hopper, which holds about 500 cards, feeds blank (unpunched) cards to the card bed that transports the cards past the punching station and past the reading station.

The reading station brushes are used in conjunction with the program drum brushes. If the program card is punched to cause an automatic duplication of data on each card, the manual keyboard is disconnected from the card punch solenoids until the duplicating process is completed.

Information stored in a given column of the programming punched card, as indicated by the reading station brushes, is used to control the punching station solenoids to record the data on the card being punched.

The program-card drum motion is synchronized both with the motion of the card at the reading station and with the motion of the following card as it is fed to the punching station. The three cards are advanced, column by column, at the same time.

After the card passes the reading station, a card stacker receives it and stacks it in a bin for removal. The card that had been punched previously then moves to the reading station, and the hopper feeds the next unpunched card

Character	Holes Punched*		Character	Holes Punched*		Character	Holes Punched*	
	Zone	Digit		Zone	Digit		Zone	Digit
A	12	1	V	0	5	=	NP	3,8
B	12	2	W	0	6	,	0	3,8
C	12	3	X	0	7	$	11	3,8
D	12	4	Y	0	8	.	12	3,8
E	12	5	Z	0	9	-	0	4,8
F	12	6	0	0	NP*	(	0	4,8
G	12	7	1	NP	1	*	11	4,8
H	12	8	2	NP	2	)	12	4,8
I	12	9	3	NP	3	/	0	1
J	11	1	4	NP	4			
K	11	2	5	NP	5			
L	11	3	6	NP	6			
M	11	4	7	NP	7			
N	11	5	8	NP	8			
O	11	6	9	NP	9			
P	11	7	Special Characters					
Q	11	8	-	11	NP			
R	11	9	+	12	NP			
S	0	2						
T	0	3						
U	0	4						

* NP - No Punch

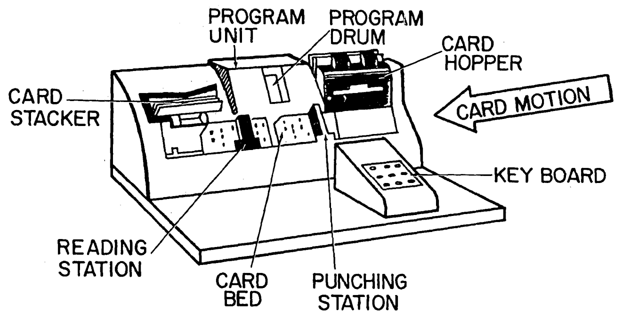
124.79

Figure 7-6.—Hollerith Card Code.

to the punching station. The punching process is then repeated.

CARD READERS

Card readers are used to translate the holes in a punched card back into the signals that represent characters. A card reader is sometimes used as a computer input device, although it is more frequently used to operate an electric typewriter or a printer in order to translate card data back into written form. Card readers are also used in conjunction with conversion devices which convert and transfer card data onto some other medium. Methods of reading



124.80
Figure 7-7.—IBM Type 24 card punch machine.

data stored on cards into a computer are explained later.

CARD VERIFIERS

A card verifier (not shown) is an electromechanical "proofreader" used as a peripheral equipment. It electrically checks the alphanumeric characters that have been punched on a card against the same data as typed by a second operator. In effect, a verification consists of a double manual-punch process which is modified to include an automatic comparison of the results. Any difference occurring during the second typing indicates an error. If the two runs agree, this fact does not provide a full guarantee that there is no error, since it is conceivable that both operators could have made the same mistake. However, the probability of this type of error is reasonably low.

CARD SORTERS

Card sorters (fig. 7-8) read portions of each card that is passed through them, (and in some cases compare certain data on the card with data programmed into the sorter) and then route the cards to separate bins or pockets in accordance with the nature of the data contained on the card. For example, if it is desired to determine the relative seniority of a group of servicemen, punched cards containing extracts of each man's service record can be fed to a card sorter. The sorter can be programmed to read those columns on the card that indicate years of service. The sorter transports the cards indicating 30 years of service into the first bin, 25 years of service into the second bin, etc.

A representative card sorter can deliver cards at a rate of 600 to 800 per minute. Some faster models can sort up to 650 cards per minute.

The card sorter in figure 7-8 uses both mechanical and electromechanical principles to perform card sorting. A column selector (not shown) is used to select the column to be read. To understand its operation, assume that it is desired to sort all cards which contain a punch (hole) in the 4's row.

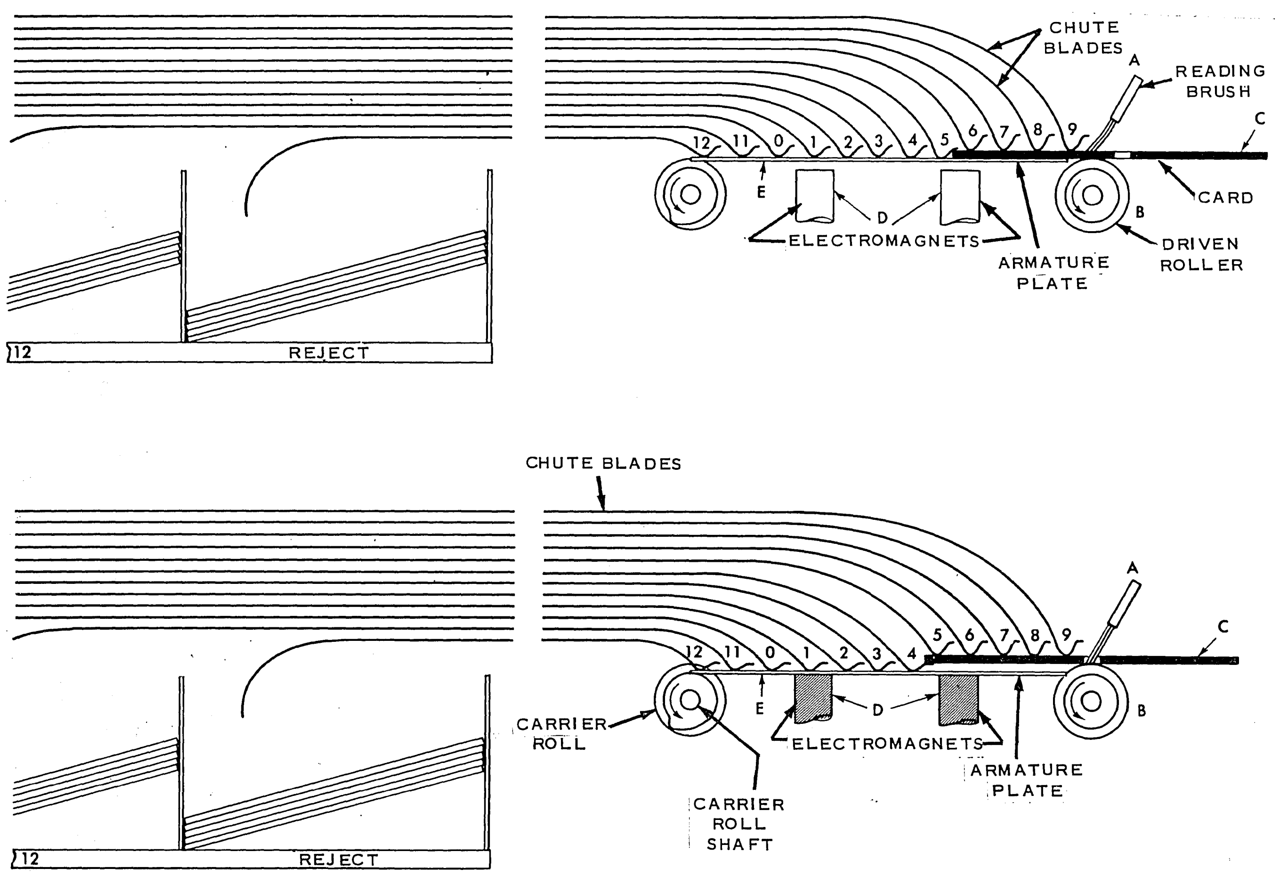
In figure 7-8A, a card (shown as a shaded line in which the punch in the 4's row is represented as an unshaded area) is advanced beneath a reading brush by a driven roller. The reading process begins at the 9's row (the bottom row of the card) and ends at the 12's row (the top row). While the card is still passing under the reading brush, the leading edge (the bottom) of that card is passing beneath the ends of chute blades. Each of the numbered chute blades terminate in an associated bin (not shown). The ends of the chute blades are supported by a moveable armature plate.

Figure 7-8B illustrates how the sorter selects the right chute blade (and therefore the proper bin) for the card. The tips of the chute blades are spaced so that the read brush (also called the "sort brush") drops through the 4-punch in the card immediately after the leading edge of the card passes under the tip of chute blade number 5. The opening in the card permits the brush to make contact with a conductive material beneath the card, thereby closing an electric circuit.

The closing of this circuit causes two electromagnets to be energized. This action exerts sufficient force against a spring tension (which normally holds the armature plate as shown in figure 7-8A) to cause the plate (fig. 7-8B) to move downward. This permits all chute blades beyond the forward end of the card to drop, thus permitting the card to pass into the proper bin.

BUFFER STORAGE

Buffers serve as intermediate storage devices to facilitate transfer of data between two mediums whose operating speeds are difficult, or impossible, to synchronize. It is frequently necessary to read data from cards, paper tapes, keyboards, etc., into the main (primary) computer memory. The speed at which input devices can supply data to storage cannot be increased sufficiently to match the ability of



124.81X

Figure 7-8.—Card sorting machine—IBM Type 82.

the computer to read-in the data at electronic speeds. The same incompatibility is encountered when reading data from memory to output devices. A buffer device is therefore designed to read-in or write-out data at speeds which are compatible with both the input/output devices and the main computer memory.

Several types of buffers are in use. The simplest type is an arrangement of flip-flop registers into which data can be slowly accumulated but can be released or read-out at electronic speeds. By design, data can be read-into and out-of the registers in either serial or parallel form. The buffer storage must be capable of reading data slowly from the input device, and, at a later time, writing this data at electronic speeds into the main memory. It must also be capable of reading-in data at electronic speeds from the main memory and writing this data slowly at the output device.

METHODS OF USING CARD READERS WITH BUFFER STORAGE

To further understand the purpose of card readers and also how they operate with buffer storage, consider the following discussion. Some card readers are designed for ''on-line'' operation with a computer. Further, the computer may be capable of selecting any one of several card readers as the input or output device.

The diagram and flow chart in figure 7-9 are used to illustrate one method of reading data from a card reader into the computer memory. Four readers are shown in figure 7-9A.

The request for read-in of data is generally in the form of a programmed instruction. When the instruction is issued, the control unit produces an input request signal on any one of the four input request lines as determined by the instruction. The same instruction (or in some cases a subsequent instruction) will select the desired function, either read or write. (A card reader is capable of performing both functions.)

Some readers read the data directly into the computer. This method is slow and does not afford best use of computer time. A more desirable arrangement and the one treated here, uses a buffer storage. The card reader passes its input over 80 transfer lines to the storage medium. (Only one of these lines from each reader is shown.) Cores are used as the storage medium in this discussion because of their simplicity.

The storage matrix contains 960 cores (only top row shown in figure 7-9), each of which can be in either the 0 or 1 state to represent any combination of holes and no holes contained on the card being read. The data are read-in (in parallel form) longitudinally across the matrix so that all 1's in the 9's row of the card are entered first. All 1's in the 8's row are entered next followed by those in rows 7, 6, 5, 4, 3, 2, 1, and 0, in that order. The 1's in the 11's and 12's row are entered (in that order) after the 0's row.

A row counter counts each row as it is entered into the matrix. After entering the last row (the 12's row) the row counter produces an output which indicates that the information is ready for transfer to the computer. If the computer is available to accept the data, a read enable signal will be developed in the computer and transmitted to the card reader to set the read enable circuits. Subsequent instructions will permit the data to be transferred (column by column) into the computer as described later.

The C register provides temporary storage of data en route to the computer. The column counter initiates the "read next card" signal after all 80 columns have been transferred to the computer at high speed. Note that the data are read into the core matrix at the speed of the input device and into the computer at electronic speeds, limited only by the ability of the computer memory to accept and store the data. Thus, the core matrix provides "buffer storage" between the card reader and the main computer memory.

The flow chart (fig. 7-9B) shows how the read-in process is executed. Upon receiving the "input request" signal from the computer, the order to "start card cycle" is issued (A). The "Read-Row" instruction causes the 9's row to be positioned at the reading station, and read. After the holes and no-holes, or 1's and 0's in the 9's row are read, the "enable row drive and store" instruction causes the drive lines in the matrix (which correspond with holes in the card) to be activated, and their selected cores are switched to the 1 state. All other cores remain in the zero state. (The memory matrix is cleared before the read-in of each card.)

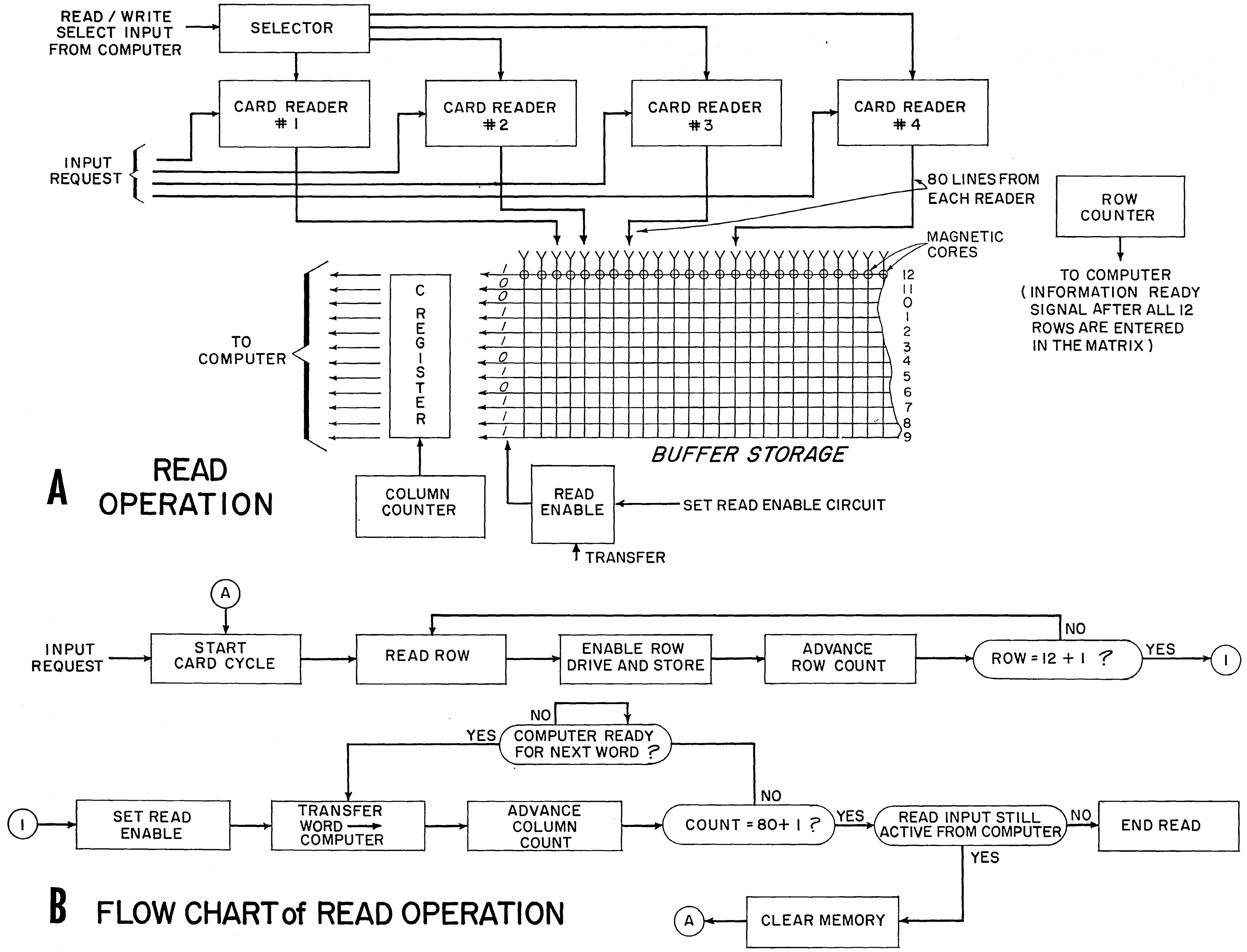
The order "advance row count" places the 8's row in the position for read-in. Because the 8's row is the next row to be read, the answer to the question "is row = 12+1?" is "no" and a signal is produced to "read-row" (row 8 in this case). The process continues, storing the data from the card row by row into the matrix, until all of the card data is transferred to the matrix. The 12's row is the last to be read-in.

After reading row 12, the answer to the question "is row = 12+1?" will be "yes." This action causes a command to be issued which sets the read enable circuits in preparation for the read-out of the data from the memory matrix. (Points ① and ① are connected.) A command line from the computer (explained under control unit in chapter 4) carries the signal to cause the read-out of one vertical word column of data (12 bits) each time this signal appears on the line. The column counter is advanced by 1 each time a 12-bit column is transferred.

If a "no" answer is produced from the "count = 80+1 circuit," and the computer is not ready to accept the next word, a method is provided for storing the word to be transferred until the computer memory is able to accept the word for storage.

After the transfer of all 80 columns, a subsequent order to "advance column count" will produce a "yes" output from the interrogation "count = 80+1?" This permits an examination of conditions to determine if the "Read input is still active from the computer." Stated differently, "is the computer still demanding the read-in of data?" A "yes" answer causes the memory matrix to be cleared and the entire operation to be repeated during the read-in of the next card (points (A) and (A) are connected). A "no" answer initiates "end read," and all card reader operations are discontinued.

To further understand buffer storage as it applies in the interest of expeditious use of computer time, consider the overall operation and use of the computer with and without the aid of buffer storage and a buffer controlled sequence.



124.82

Figure 7-9.—Card reading operation using buffer storage.

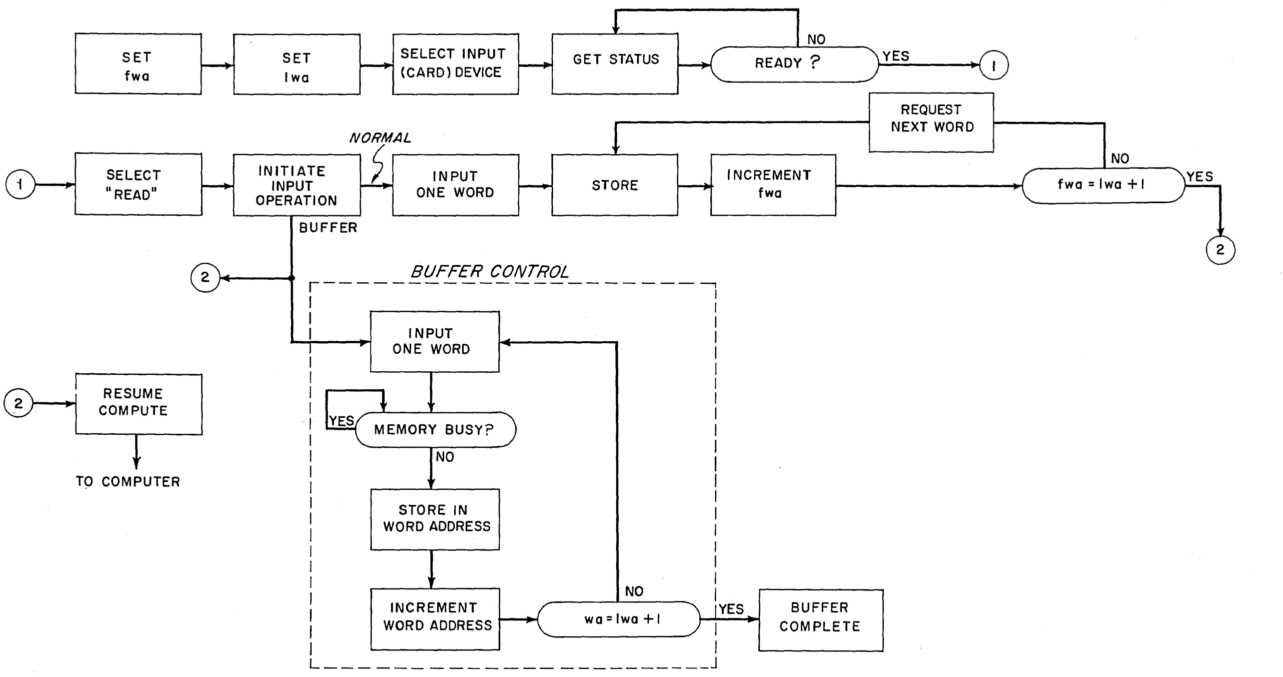
In the flow chart of figure 7-10, the instruction issued to cause read-in designates an address at which the read-in must begin (first word address, fwa) and the address of the last word (1wa). For a read-in instruction, these addresses refer to the location at which the first word and last word are to be stored in the main computer memory. For a read-out instruction, fwa and 1wa would encompass the total data to be read-out. The data are read into or out of memory in sequential order from the "fwa" to the "1wa."

A subsequent instruction causes the computer to select the "input device," which in this case is presumed to be a card reader. Next, the status of the selected device is determined. If this device is not available or other conditions are not satisfactory (such as "power switch not on"), a "no" output will result from the "ready" interrogation. In some cases, a light, alarm, or other means of alerting the operation to correct the situation is initiated. If the "ready" interrogation produces a "yes" answer ((1) and (1) are tied together), the command is issued to select "read." This action selects the "read function," meaning that data is to be read into the computer rather than written-out.

After all such preparations (as described) are completed, the "initiate input operation" is issued, and the card reading process begins. If buffer storage is not used (the method described here as NORMAL) the computer will issue the order "input one word." This is followed by a store instruction which places the word in the fwa in storage. The computer now causes a counter to increment fwa or count 1, and to examine the new address number to determine if the new word address is the same as the last word address plus 1. If the answer is "no" the next word is requested. This process repeats itself until the output from the interrogation "fwa = 1wa+1?" is "yes."

Because the numbers (2) and (2) represent the same point, a "yes" answer from the interrogation "fwa = 1wa+1" causes the command to be issued to permit the computer to resume operations under the influence of the stored program. Note that the computer is not available to perform any operations except read-in during the time that the read-in is in progress.

Now consider the action using the buffer method. All conditions and operations are the same as described earlier up to and including the "initiate input" operation. The fwa and 1wa



124.83

Figure 7-10.—Flow chart of normal and buffer read-in of card data.

are stored in a register in the buffer control section so that the buffer operation will stop when the action is completed.

As soon as the "initiate input operation" command is issued a buffer control unit begins to operate and the main computer is released to perform other operations (arithmetic, control, etc.). Buffer control operates independently of the main computer.

After buffer control has caused the read-in of the first word, an interrogation is performed to determine if memory is busy. (It is possible that memory, and all associated drive line and registers, are busy in the performance of some other operation.) If the answer to this interrogation is "yes" no attempt is made by buffer control to enter the data into the main computer storage until memory is available to accept and store the data.

After storing the data, a counter increments the word address (wa), and the new address is compared with the previously stored code which represents 1wa + 1 to determine if the last word has already been stored. If the answer is "no," the process repeats until a "yes" output is produced. This "yes" output indicates that the buffer operation is completed. Note in this case that the computer is not directly involved in the buffer operation.

The terms "buffer" and "buffer storage" are used to describe almost any operation which takes place in a computer with the aid of any form of intermediate storage. However, the term "buffer storage" is most frequently used to denote the type of operation just explained. The word "buffer" generally refers to a device or arrangement of devices or registers which perform temporary storage of data as they are being transferred in the computation process.

CARD CONVERTERS

Card converters (not shown) make it possible to change the coded data written on cards into coded data of another form. Conversely, they can receive data in one coded form, decode it, and punch the data onto cards in card code. Converters are used in conjunction with magnetic tapes, paper tapes, other card-handling machines, and radio and teletype transmission systems.

Conversion Method

One method of converting card data to another form is as follows. If an "A" is detected by the card reader, a pulse is produced on the number-12-position reading brush and simultaneously on the number-1-position brush (see figure 7-5 for card code). If "A" is to be represented by a binary "0001" in the new code to be used, the two pulses representing "A" in card code can be applied to a translating matrix. The matrix sets the proper values into four flip-flops. If the flip-flops are arranged so that the lowest order circuit is on the right followed by ascending orders to the left, the first flip-flop can be set, and the other three remain in their zero state. Thus an "A" (1, 12) in card code produces an "A" (0001) in the new code. Basically, this type of conversion, or translation, occurs for every type of converter.

Magnetic Tape Units

Although magnetic tapes are very popular as storage devices for large quantities of data, they are not considered practical for use as the main storage element of a computer due to their long access time. Magnetic tapes provide excellent secondary storage when the computer is programmed to read-in or read-out certain data to the tape in the interest of conserving main computer memory.

Other advantages of magnetic tape units are (1) their ability to retain data over a long period of time, (2) large storage capacity, (3) low cost, and (4) ease of exchanging one reel for another.

The principles involved in reading and writing on magnetic tapes are similar to those used with commercial tape recorders. A detailed explanation is treated in the chapter on Memory and Storage Units in this text.

A card-to-magnetic-tape converter translating matrix produces "1110001" when an "A" card signal is applied (see seven-bit magnetic tape code in chapter 6, figure 6-15). The magnetic tape character, 1110001, is then converted by a sequence of recording heads into seven appropriately magnetized areas on the tape. In addition to the translation circuitry, tape and card units respectively require control circuits to start and to control other internal operations.

A card-to-magnetic-tape converter is generally used when it is desired to store large quantities of data. For example, a single 2500-foot reel of magnetic tape can store the data recorded on 20,000 punched cards. In newer tape units this figure is even higher.

Tape units also provide faster read-in times than can be obtained with cards. The average "fast" card reader can only read up to a rate of about 1330 alphanumerical characters per second (1000 cards per minute, 80 characters to the card). Some card readers are slower than this. A representative magnetic tape drive can read continuously at a rate of over 100,000 alphanumerical characters per second.

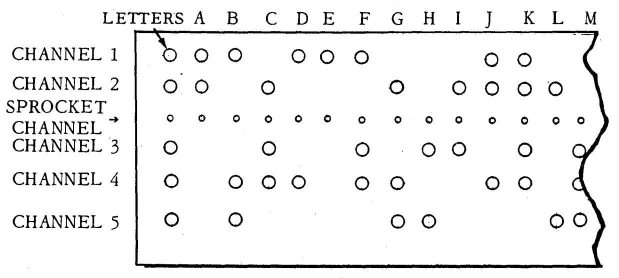
Many data-processing systems use punched paper tape (discussed later) as their data storage medium. It is often desirable to be able to convert data from punched paper tape to cards, and vice versa. Again, a translating matrix is used. This time it drives either a paper punch from a card reader or a card punch from a paper-tape reader.

PAPER-TAPE UNITS

Punched paper tapes (fig. 7-11) are often used with military and commercial data-processing systems. The advantage of this form of data recording is that most business machines can easily be modified or designed to punch paper-tape records of their operations. These paper tapes are punched in a modified teletype code that can be transmitted over wire or radio teletype lines.

Many large installations use paper-tape units as preliminary data-processing devices at the point of transaction. All pertinent data are sent by teletype to a central data-processing installation. The paper tape data can be converted and read onto magnetic tape (or some other suitable medium), then fed into a large computer.

Paper tapes are available in several widths (7/8", 1", 1-1/8", etc) and in 100-foot, 350-foot, and 700-foot rolls. A sprocket channel (a line of small circular holes that are punched on the tape at the time data is recorded), appears longitudinally along the length of the tape, as shown in figure 7-11. These holes are engaged by sprockets that drive the tape past punching and reading stations. Reels are used to store the tape.



124.84

Figure 7-11.—Five-channel paper tape code.

PAPER TAPE WRITING AND READING

The writing and reading techniques used with paper tapes are almost identical to those used with punched cards. The holes that are punched in the paper tape however, are round rather than rectangular in shape.

The speeds used with paper-tape equipment are, in general, lower than those used with punched cards. A high-speed paper-tape punch used with some commercial computers can punch at a rate of 120 characters per second. Most mechanical punches punch from 10 to 60 characters per second.

A type of reader which uses the photoelectric principle (rather than mechanical) provides the fastest means of reading punched tapes. With this method, a light-sensitive material is placed beneath each of the longitudinal channels and the sprocket channel (fig. 7-11). A light placed above the tape and over the light-sensitive material (not shown) causes an output signal to be produced from each channel in which a hole has been punched. The signals are amplified and fed to the computer as input information.

The sprocket hole outputs (which occur at each reading point on the tape) signal the read interval. Therefore, each channel output is sensed when a sprocket hole passes the station. The tape moves continuously until it is ordered to stop by a STOP instruction.

Photoelectric type readers can feed up to 1000 characters per second into the computer. This speed is comparable with that of the fastest card reader available.

PAPER TAPE CODES

The paper-tape codes that are used with data-processing systems are the five-, six-, seven-, and eight-channel codes. One widely used code (and the only one shown here) is the five-channel, modified, teletype code (fig. 7-11).

The section of tape shown illustrates how letters are represented on the tape. Figures are similarly represented, although the keyboard symbol for "figures" must be punched before figures can be recorded. The symbol for "letters" must be depressed at the keyboard before letters can be recorded. Thus, the "letters" character represented on the left of the section of tape shown indicates that all subsequent characters represent letters. Table 7-1 shows all letters and figures represented by this code.

Table 7-1.—Five-Channel Data Processing Code and Tape.

5. CHANNEL DATA PROCESSING CODE*

(MODIFIED TELETYPE CODE)

Character	Punched Channels	Character	Punched Channels	Character	Punched Channels
A	1,2	U	1,2,3	CARRIAGE RETURN	4
B	1,4,5	V	2,3,4,5		
C	2,3,4	W	1,2,5	ADVANCE PAPER	2
D	1,4	X	1,3,4,5		
E	1	Y	1,3,5		
F	1,3,4	Z	-----		
G	2,4,5	1	1,2,3,5		
H	3,5	2	1,2,5		
I	2,3	3	1		
J	1,2,4	4	2,4		
K	1,2,3,4	5	5		
L	2,5	6	1,3,5		
M	3,4,5	7	1,2,3		
N	3,4	8	2,3		
O	4,5	9	4,5		
P	2,3,5	0	2,3,5		
Q	----	FIGURES	1,2,4,5		
R	2,4	LETTERS	1,2,3,4,5		
S	1,3	SPACE	3		
T	5				

*NOTE Letters are preceded by the symbol for "Letters," and numbers are preceded by the symbol for "Figures."

HIGH-SPEED PRINTING

In many military computer applications, such as fire control systems, the computer output is used to train weapons or other similar devices (discussed in the NTDS equipment in later chapters). In many other applications, the end result is printed data. Punched cards, punched paper tape, and magnetic tape are satisfactory for providing data in forms intelligible to computers, but numbers and words are most easily interpreted by human beings. Therefore, printing units are required to translate internal computer data into words and numbers that a human operator can understand.

A computer-operated electric typewriter can print at speeds of approximately 10 letters or numbers (referred to as printed characters or digits) per second. Even though this results in printing 600 characters per minute, as compared with the 300 characters per minute that are produced by the average human typist, the electric typewriter is not fast enough for all operations.

PRINTING PRINCIPLES

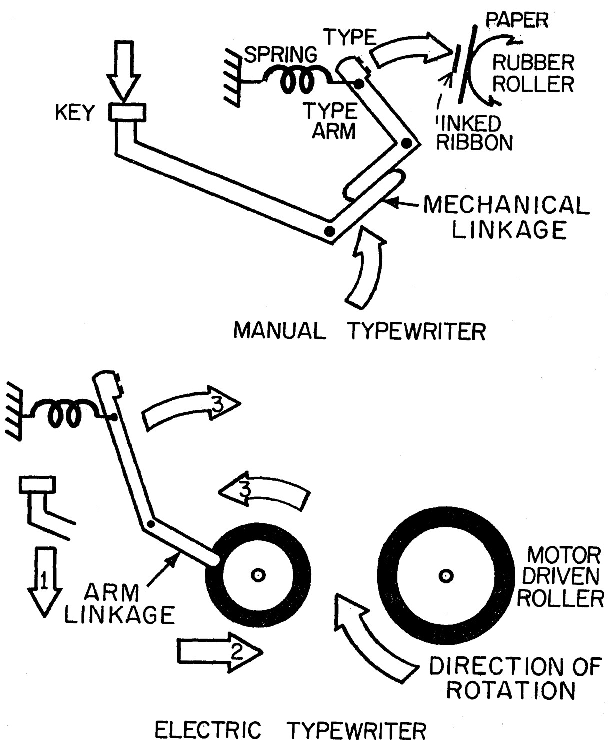
The operating principles of a mechanical typewriter (figure 7-12A) are easily understood. Downward force on a key actuates a mechanical linkage. This moves an arm that holds a single character. The type strikes an inked ribbon and presses it in contact with a sheet of paper that is held against a roller. The pressure that is placed on the ribbon by the piece of type causes the raised outline of the type to be printed on the paper.

The electric typewriter (fig. 7-12B) uses a rotating roller to drive the type arm linkage. Pressing a typewriter key causes a particular linkage to be pressed against the rotating roller. With this arrangement a character can be printed in a shorter time than is possible with mechanical typewriters.

The bar printer (fig. 7-13) is a variation of the typewriter. The type characters protrude from the edge of a bar. A character is printed by a hammer which strikes the paper and presses it against an inked ribbon and the type bar.

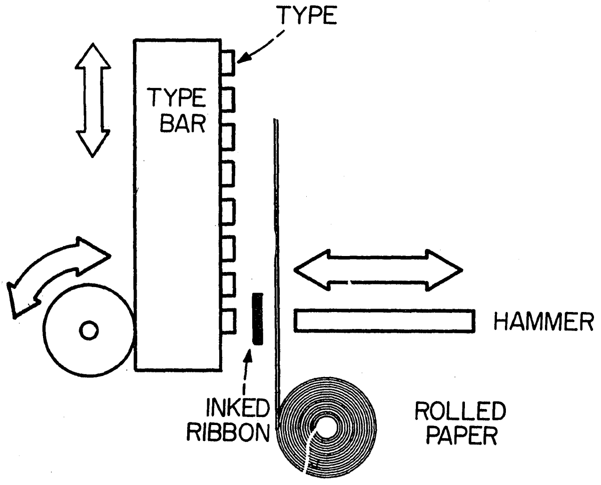
The type characters are mounted so that one appears above the other. A particular character is selected for printing by raising and lowering the type bar so that the desired character is lined up with the hammer.

Groups, or "gangs," of type bars are used to make up a multibar printer. The number of type



124.85

Figure 7-12.—Comparison of manual and electric typewriter principles.



124.86

Figure 7-13.—Bar printer.

bars that are used depends upon the number of characters that must be printed simultaneously. The bars are geared so that they are similar to toothed racks. The digitally operated pinions (not shown) that move the racks are driven by two stepping-switch mechanisms (one for each direction of motion).

Let us assume, for example, that the letter S is printed by stepping the type bar ten times. An S in computer code must be converted into ten stepping pulses which advance the type bar ten times. The printing hammer is then actuated and the letter is printed. After printing, the type bar is returned to a suitable rest position, or it is permitted to remain at the S position until the next stepping pulses are applied. These new stepping pulses are automatically corrected to allow for the fact that the bar is presently at the S position.

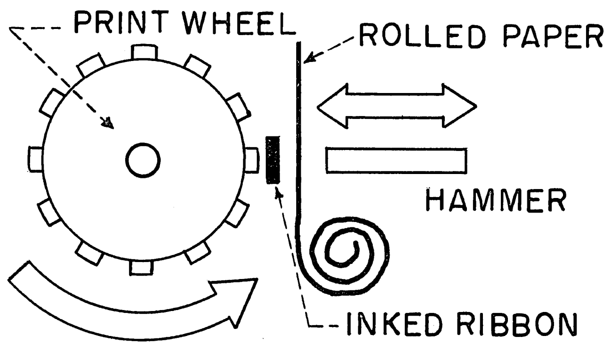
Another type of printer has the character type on the edge of a wheel (fig. 7-14). As many wheels as desired are arranged in parallel. The type wheels are continuously rotated. The striking of the hammers against the paper and ribbon is synchronized with the wheel rotation so that certain characters are selected for printing. This arrangement decreases the time required to position characters under the hammers, and provides higher printing speeds than the gang printer.

PHOTOGRAPHIC PRINTING

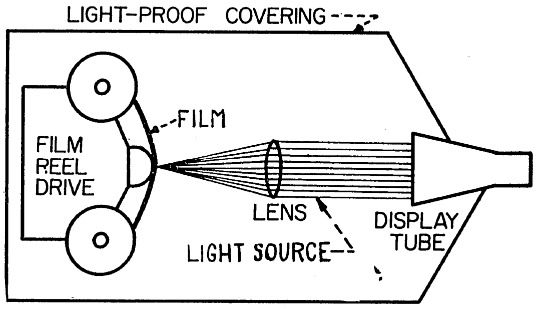
Photographic printing is accomplished by photographing the display produced by any one of several types of electronic character-writing tubes. The basic principle is illustrated in figure 7-15A. The negatives produced using this process can be used to make photographic prints or for offset printing.

The electronic character-writing tube and its associated circuitry (not shown) receive binary signals that represent alphanumeric characters. Translating circuits operate a cathode-ray (display) tube that is designed to display the alphanumeric characters in English letters and Arabic numbers.

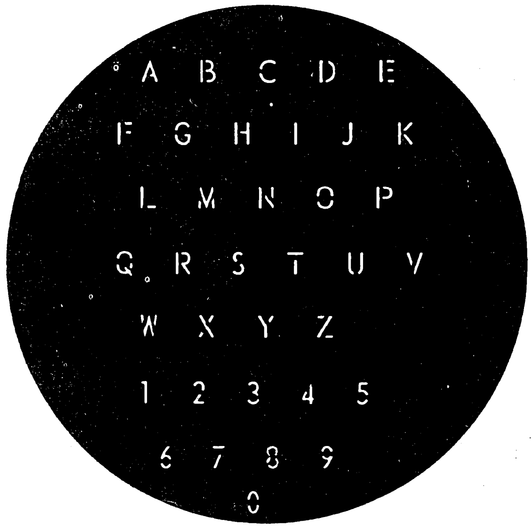
The simplest display tubes use a form of grid or matrix (fig. 7-15B) which, in effect, is an electronic stencil. The electron beam is aimed at a specific character in the metallic grid plate.



124.87
Figure 7-14.—The type wheel printer.



A BASIC PRINCIPLE



B MATRIX (GRID)

124.88
Figure 7-15.—Photographic printing.

The holes in the grid are in the shape of the characters to be printed. When the beam is deflected to the desired character the electron beam that passes through the hole has the shape of the character. A second deflection positions the character-shaped beam to its proper position on a flourescent screen. This process is continually repeated, so that the screen displays a series of characters which are photographed.

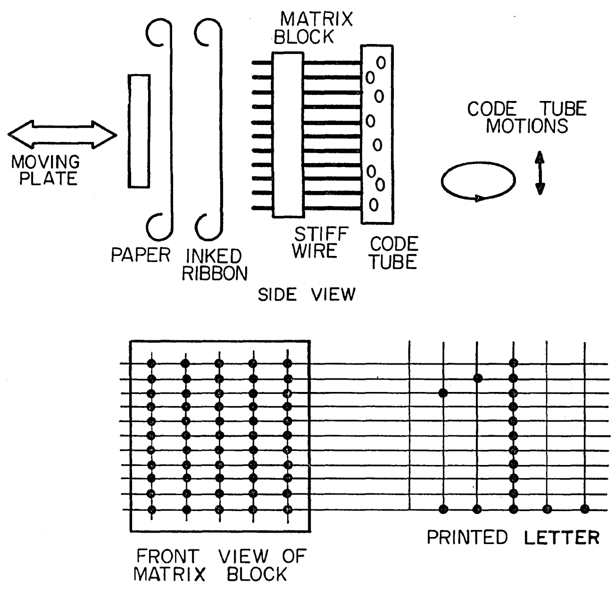
Other photographic systems use waveshaping circuits which generate voltages to approximate the component lines of each character, so that a simple cathode-ray tube can be used to display characters.

Both of the cathode-ray tube arrangements that have been discussed require elaborate beam-gating and deflection systems. However, they possess the advantages of being extremely fast and of requiring no mechanical moving parts.

The camera is usually contained in a light-proof box that surrounds the face of the tube. An optical system is used for recording the tube display on film. The film is reel-fed, and is stepped to the next frame each time that the tube message changes. A shutter is not required, since the cathode-ray tube blanking pulses perform the same function. The film-drive mechanism is the only mechanical device in this entire system.

The wire punch printer uses a different approach for forming characters. A matrix of stiff wires is used either to print letters or to punch holes (fig. 7-16). By using the proper combination of wires, it is possible to punch any character or letter. The wires of each matrix are bundled together and are run to a cylindrical code tube that has holes machined into its surface. The wires are fanned out along the longitudinal axis of the tube, with each wire resting against the surface of the cylinder. When a particular character is to be printed, the tube is rotated and moved along its axis so that the correct combination of individual wires is backed by metal, and the other wires are located over holes. At the printing end of the wire bundle, a moving plate pushes paper into contact with the matrix of wire ends. Those wires that are backed by metal will exert sufficient force on the paper to print the character. The remaining, unbacked wires will not exert this force.

If the plate is smooth, and an inked ribbon or carbon paper is placed between the wires and the paper, the backed wires will press against the ribbon and will print the appropriate character as a group of black dots. The wires are spring loaded (not shown on the diagram) so that they will return to their original positions when the plate pressure is removed.



124.89

Figure 7-16.—Wire punch printer.

CHAPTER 8

PROGRAMMING

Programming, as related to computers, involves (1) the analysis of a problem, (2) the development of a flow chart or a plan to solve the problem, and (3) the formulation of each instruction necessary to arrive at the solution to the problem. The organization of instruction obtained from this procedure is sometimes called a "list of instructions," or a "routine."

To further clarify the three distinct phases of programming, consider the procedure one would use to find the total resistance in a circuit (not shown) which contains a resistor R_s, connected in series with two parallel connected resistors R_p1 and R_{p2}. We first analyze the problem by constructing the general formula which must be used. This analysis reveals that the equivalent resistance of R_p1 and R_{p2} must be found by using one of the formulas which pertain to parallel resistances, i.e., the reciprocal formula, the "product-over-the-sum" formula or the "like method." It is known that this equivalent resistance value is series connected to R_s, and must therefore be added to R_s to obtain the correct total resistance value. This completes the analysis phase of the problem.

The second phase involves formulating the procedure to be used to accomplish the steps necessary. Using the "product-over-the-sum" method to find the equivalent parallel resistance, the general instructions would be:

Step 1. Multiply R_{p1} x R_{p2}
Step 2. Add R_p1 and R_{p2}
Step 3. Divide the results of Step 1 by the results of Step 2.
Step 4. Add result of Step 3 to R_s.

The final phase of programming the development of a list of instructions is illustrated at a later point in this chapter. For the present it is sufficient to say that each detailed step which must be accomplished by the computer must be listed, and must be capable of being executed by the particular computer.

The study of programming procedures, as presented in this chapter, will be of little value unless the reader has previously studied and understood the material presented in all previous chapters of this text. For example, the block diagram discussion presented in chapter 2 points out the main sections of the computer and how each section is involved in the operations necessary to arrive at a solution to a problem. It also points out some of the general capabilities of computers which, along with the specific capabilities of a given computer, must be known by the programmer so that any program derived will contain instructions capable of being executed by the computer.

Chapter 3 treats number systems and Boolean algebra. The basic arithmetic operations, such as addition, subtraction, multiplication and division, and the basic circuits used for logic mechanization of Boolean expressions, are vital information which must be known before the programming of arithmetic operations can become meaningful.

The basic registers and control circuits of the control unit and how the control unit operates to read, interpret, and control the execution of each instruction word is explained in chapter 4.

In the preparation of a program, the composition of the computer word must be known. The sign-bit, operation code (or functional code), operand address, and various designator portions (discussed later) of the word, must be known before detailed instructions can be written. The methods used to perform arithmetic operations and for providing main storage must also be known by the programmer. These topics are treated in chapters 5 and 6, respectively.

TERMINOLOGY

It is well at this time to consider the meaning of some of the terms used principally with regard to programming and not heretofore covered. Those terms which are not presented here are either defined at a later time in this chapter when they can be better understood, or have been treated in an earlier discussion.

SUBROUTINES

A subroutine is a portion of a routine that is complete in itself and can be isolated from the content of the larger routine. Stated another way, a subroutine is a self-contained list of instructions for executing some particular operation. If the subroutine contains the calculations necessary to compute a function such as sin X and tan X, it should be coded in such a way that it may be used in a number of routines (or in as many places as necessary in a given routine) wherever such a function is needed.

WORD MODIFIERS

Each instruction which is used in the computation process is first read from memory into the instruction register (as described in chapter 4 of this training course). The instruction is subdivided into the operation register and the address register.

It is frequently necessary to alter or modify the address portion of the word in the control unit just prior to its use to obtain data from the main memory. In such cases, one or more address modifier sections are added to the instruction word to cause an automatic change in the address to be referenced in memory.

Indexing

An index register is a counter which is generally used (1) to change the numerical value of the address portion of a computer word to obtain an effective address, or (2) to monitor subroutines (such as sin X, and tan X).

The first action (changing the numerical value of the address portion of a word to obtain an effective address) is accomplished by modifying the word address (the word in the address register) by the absolute value of a number stored in the index register. The index process does not alter the computer word or the number contained in the index register, thus making possible the use of the word as many times as necessary in its indexed or non-indexed form.

The need for indexing is explained using the following hypothetical example. Suppose it is necessary to read-in data to storage from a magnetic tape reader five times in the execution of a program and that ten instructions are required to cause the entire read-in to be executed. Without the use of indexing it is necessary to write a total of fifty instructions. If, however, the ten instructions for read-in of the tape are stored in memory (as instructions) and the index register is set five different times so that it contains the proper numbers to produce the desired addresses in memory into which the data must be read, only ten additional instructions will be necessary (five to initiate read-in at the proper time and one each time it is necessary to set the index register with the desired index number).

The index register is generally referred to as the "B-register group." The number of registers in the group vary depending on the computer. Each register is independent of the other and is separately used. The contents of any register in the group can be added to the contents of the address register.

When a B register group is used, it is necessary to add a section to the computer word for the purpose of selecting and setting the desired B register. This section is called the "b-designator" and may be 01 to designate the B1 register of the B register group, 02 to designate the B2 register, etc. A single address word may be modified as illustrated in table 8-1.

TABLE 8-1.—Sample Instruction Word Using B-Designator.

101	01	1011010
Operation Code	Index Register designator (b-designator)	Operand Address

If the B1 index register (designated by 01 in the instruction word in table 8-1) has been previously set so that it contains 0000101, the instruction in table 8-1 (when read from tha main program) will produce an effective operand address of

+1011010	Operand address in instruction word
+0000101	Contents of B1 index register
1011111	Effective address (represents address to which reference is made in memory).

The second use of indexing is illustrated using the following examples. Assume that it is desired to repeat a given operation x number of times. If the number (x) is loaded into the index register, the register will count backwards (toward zero) each time the instruction is executed until x counts have been performed.

If it is desired to alter the address portion of the instruction word by an amount n-1 for x number of instructions and before each of the instructions is executed, the action can be accomplished by setting the index register with the number by which the first instruction is to be modified and altering the index register contents as desired after each instruction to produce the desired effective address before the execution of the next instruction. After repeating this action x number of times, a programmed instruction causes an exit from this procedure.

A similar action, called a ''loop function'' is accomplished by repeating a given set of instructions (say a shift instruction) x number of times before exiting from the loop action. The exit may be caused by comparing the number of times the instruction is repeated with a number previously set in the index register (called the ''criterion''), or by a programmed instruction.

Word Arithmetic

Word arithmetic, as defined here, refers to the process of removing a word from storage, altering the word by some arithmetic process in the arithmetic unit and returning the word to the same address in memory. The process may be repeated any number of times on the same word. Thus, modification of a word using arithmetic differs from modification by indexing since, in the latter process, the word is not changed in memory.

Designators

It has been shown that a computer word contains several sections. For convenience, these various sections are assigned a given letter, called a ''designator.'' One instruction format in table 8-2 illustrates the use of word designators in a 24-bit word.

Table 8-2.—Instruction Format

Designator				
f	j	k	b	y
101101	101	100	001	111001101

Designator		Specification	Interpretation
f	=	Operation Code designator	Type of operation to be performed.
j	=	Branch condition designator	(1) jump or skip operation. (2) index (B) register specification. (3) repeat.
k	=	Operand interpretation designator	Read, store or replace instruction.
b	=	Address modification designator (index designator)	Register to be used for address modification.
y	=	Operand designator	Operand or address of operand.

Sequences

Sequences or cycles describe the procedure or steps by which a computer executes each instruction. A representative computer (and the one used as a representative computer later in this chapter) will perform its operations using four sequences A, B, C, and D. Each sequence is initiated, controlled, and terminated by the control unit.

1. The A sequence is generally the read-next instruction sequence. This sequence obtains the forthcoming instruction word (from memory or some other designated source) and controls preliminary operand modification if necessary. The repeat A subsequence which controls preliminary operand modification for the repeat instruction is also used. An interrupt A subsequence provides entrance into an interrupt subroutine in the event of an internal or external interrupt (discussed in a later chapter).
2. The B sequence performs operand modification and obtains the appropriate operand from storage.
3. The C sequence controls primarily the arithmetic operations as prescribed by the current instruction word.

4. The D sequence controls the storage of the appropriate operand in the location specified by the instruction word.

5. The read as modified by k subsequence controls the operation of obtaining the operand as specified by the instruction and the k designator.

6. The store as modified by k subsequence controls the storage of the operand as specified by the instruction and the k designator.

7. In general, it can be stated that the control section directs computer operations necessary to execute any given instruction or series of instructions.

REAL-TIME CLOCK

The real-time clock (RTC) is the means by which the actual elapsed time is measured in seconds or fractions thereof. The RTC is thus used for timing in/out operations, keeping record of the time required for any arithmetic operation, or the time required to complete a given program.

PROGRAMMING FUNDAMENTALS

In order to gain a full understanding of programming fundamentals, it is necessary to use an actual computer for which simple programs can be derived. To this end, the Digital Data Computer CP-642A/USQ-20(V) (presently being used as a component part of the Naval Tactical Data System) is used as a vehicle in this presentation. The actual discussion of programming procedures for this computer is preceded by a brief general description of the computer.

The computer functions are divided among four sections: control, arithmetic, input/output, and storage. These sections (fig. 8-1) are integrated in such a way that their functions are dependent on operations of the others. The division into sections is for the purpose of explanation and description of parts having allied functions only and not to describe any physical separation.

CONTROL SECTION

The control section contains registers, modifier, and designator interpretation circuity and timing sequence circuits. The timing sequences are used to control computer functions and to execute instructions in accordance with the stored program. Basically, the control section consists of two parts—the operational registers (and modifiers) and the sequence control circuits.

Operational Registers

The operational registers are used for storage of the instruction word (with its modifiers, designators, and operand) throughout the execution cycle. The computer uses a fixed word length of 30 bits. The modifier circuits provide the control section with the capability of modifying data under certain conditions.

(Some registers contain only 15 bit positions since it is never necessary to store longer words in these registers. They store either the upper 15-bits of a register, say α_U, or the lower 15-bits, α_L.)

1. The U register is the principal control section register. It holds the instruction word during the execution of the instruction; and in certain cases, the lower 15 bits are used as the operand. The instruction word designators (f, j, k, b, and y) stored in the U register are applied to translator circuits that control certain computer functions.

2. There are seven B registers (15 bits) used as indexing-incrementing registers. They are used to modify U_L (the lower 15 bits of U register) but may serve other functions. An example of this is B_7 which is used as storage for the repeat count for a repeat instruction. Each B register can provide a B=0 test to determine if a given action has been counted down to zero. The use of the B registers is determined by the instruction word. The particular B register used is determined by the b or j designators.

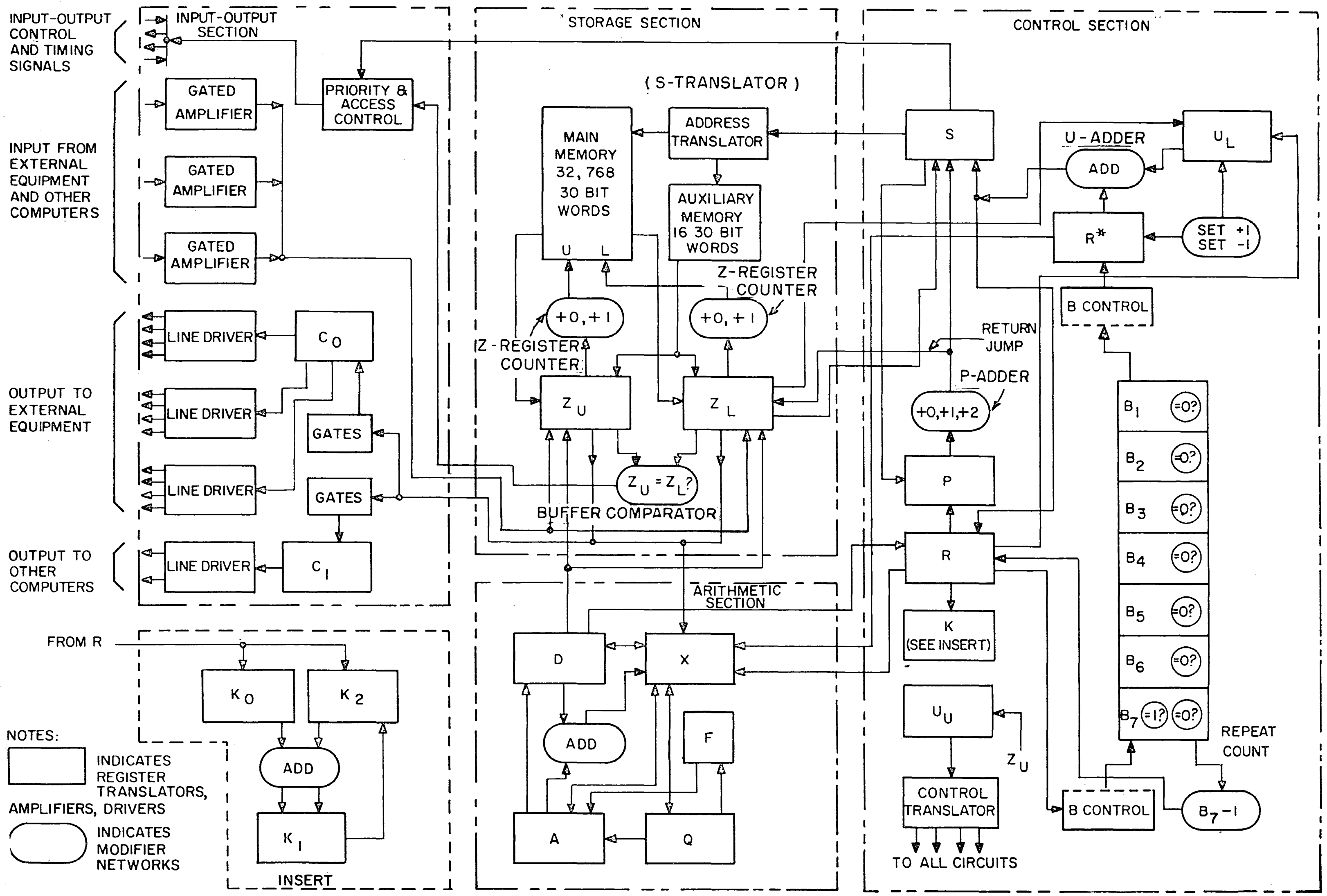
3. The R register (15 bits) is used as a communications register between the control and arithmetic sections and for communications within the control section.

4. The R* register (15 bits) is used primarily as an indexing register.

5. The P register (15 bits) is the program address register. It holds the address of the instruction being executed.

6. The K register functions as a shift counter for all arithmetic operations that involve shifts.

7. The U_L + R* adder provides the means for modifying U_L as specified by the instruction word.



124.90

Figure 8-1.—Digital Data Computer, CP-642A/USQ-20(V) block diagram.

ARITHMETIC SECTION

The arithmetic section is composed of registers and special circuits that perform the arithmetic operations of the computer. These arithmetic operations include addition, subtraction, multiplication, division, shifting, special applications, and/or combinations of these operations. These operations are performed by using the adder and special data manipulations between registers.

The A, D, Q, and X registers (30 bits each) are the primary registers in the arithmetic section. There is also a special purpose F register used as a secondary shift register. The A and Q registers are addressable (they can be loaded from the console), and the others are not. The normal word length is 30 bits but there are provisions for a 60-bit product (multiplication) and a 60-bit dividend (division).

The Add instruction (designated by function code f = 20) is accomplished by placing the augend in the A register and the addend in the D register (see table 8-3). Decimal numbers are used in the example for ease of explanation, although binary numbers are used in the computer. The adder combines these and produces a sum which is entered in the X register and then transmitted to the A register.

The Subtraction instruction (f = 21) also uses the A, D, and X registers. However, for subtraction, the minuend is placed in the A register and the complement of the subtrahend is placed in D. The adder now produces the difference which is transmitted to X and then to A.

The Multiplication operation (f = 22) is performed by placing the multiplicand in the D register and the multiplier in Q. Then a subsequence, called the multiply step is initiated. This subsequence controls the shifts and additions necessary to produce the product. The product will be stored in Q if it is single-length (30 bits) and in AQ if it is double-length (60 bits) with A holding the most significant bits.

The Division process (f = 23) is performed by placing the divisor in the D register and the dividend in AQ and then initiating the divide sequence. The dividend is always considered to be 60 bits and A holds the most significant digits if the dividend is double-length. The divide sequence controls the subtractions and shifts necessary to produce the quotient (which is stored in Q), and the remainder, if any, is stored in A.

Any of these arithmetic operations can be performed using numbers of positive and/or negative values. Suitable steps are taken so that the final result has the appropriate algebraic sign.

Table 8-3.—Basic Arithmetic Operations.

Operation code				
f = 20	10	AUGEND	A Register	
	+ 5	ADDEND	D Register	
	15	SUM	X Register then to A	
f = 21	10	MINUEND	A Register	
	- 5	SUBTRAHEND	D Register (contains complement of subtrahend)	
	5	DIFFERENCE	X Register then to A	
f = 22	10	MULTIPLICAND	D Register	
	x 5	MULTIPLIER	Q Register	
	50	PRODUCT Q(single length)	AQ Register (double length)	
		2	QUOTIENT	Q Register
f = 23	DIVISOR D Register	$5\overline{)10}$	DIVIDEND	AQ Register
	REMAINDER (if any)		A Register	

The Shift instructions (f = 01, 02, 03, 05, 06, 07) are performed by referencing a particular register and specifying the direction and amount of shift. The shifts are performed by the shift setup, AQ shift step control, and A and/or Q shift step subsequences. During right shifts, the lower order bits are lost and the sign of the number is extended by entering a 1 bit in the highest order bit position during each shift (sign extension). For left shifts, the lower order bits are replaced by the higher order bits as the shift progresses (circular).

The adder (30 bits) is a subtractive device with end-around borrow capabilities which uses the principle of one's complement binary arithmetic.

INPUT/OUTPUT SECTION

The input/output section contains gated amplifier circuits to supply inputs from external equipment and other computers. The amplifier inputs are fed to the Z register and then into the storage section. External equipment used with the computer may include auxiliary memory units such as magnetic tape or magnetic drum units, teletypewriter units, high-speed printer units, paper-tape units, and other related equipment.

The input/output section supplies output to the external equipment and other computers through the line driver circuits. Output is supplied through the C_0 register to the external equipment line drivers, and through the C_1 register for intercomputer line drivers. The priority and access control serves a coordinating function, regulating input signals (using the S register) according to memory availability; and output signals (from the Z register) according to control section and external equipment requirements.

Priority

Since there are 14 input/output channels, there is the possibility that more than one channel would require computer attention at any one time. The priority networks determine which channel is to be considered first, if more than one channel has data on the lines. Channel 13 has the highest priority and channel 0 has the lowest priority (channels 0 and 1 are used for intercomputer transfers).

Control Sequences

Input/Output operations are partially controlled by several timing sequences.

1. The Scan sequence interrogates all input/output requests and processes the data from the highest-order priority channel on the basis of channel number and type of request.
2. The I/O_1 sequence updates the real-time clock control address and updates the contents of the buffer control address.
3. The I/O_2 sequence updates the upper half of the real-time clock control address (if necessary) and gains access to memory for data transfers. This sequence also generates the appropriate acknowledge signal for data transfer.

STORAGE SECTION

The storage section consists of the main magnetic core memory; wired auxiliary memory, and associated address; transfer; and control circuits.

The main memory, constructed of modular arrays of ferrite cores, has a capacity of 32,768 words of 30 bits each with provisions for handling each as two, 15-bit words. A word is referenced by loading the S register with the proper address, and the associated translator selects the proper cores.

Current pulses are then applied to the cores and the data at the selected address is read out to the Z register. This is a destructive readout, but the data is written back into memory as a part of the memory cycle. The time required for this read/write cycle is eight microseconds, during which time the memory lockout circuits are active. The lockout prevents other memory references from occurring at the same time.

Each memory core is a bistable element capable of storing a "1" or a "0" depending upon the direction of magnetization. The time required to drive a core from one state to the other is approximately 1.2 microseconds. Each core surrounds the intersection of four wires by which its state can be changed and sampled.

All data exchange with memory is accomplished through the Z register.

The auxiliary memory is a 16-word permanent (wired) memory. It is made up of plugs and jumper wires attached to the five memory chassis, and provides permanent storage for important instructions such as loading routines.

FLOW DIAGRAMS

Any problem that can be solved by mathematical procedures can be solved by the computer. If it is determined that the problem can be solved best by the computer, it is then necessary to formulate the problem in the language of the computer. A program of instructions and the data needed for the solution of the problem must be devised.

A flow diagram or flow chart indicates the flow, or steps, in the computation that leads to the solution of a particular problem. A basic flow diagram usually lists the series of simple arithmetic steps that are to be performed by the computer. It is imperative that the coder familiarize himself with the overall operations and the peculiarities of each particular computer so that he can construct the flow chart and subsequent instructions with regard to the capabilities of that computer.

Usually, more than one flow diagram is formed for the more complicated problems. The first flow outline may be equations in mathematical language written in the sequence in which they will be computed, together with brief explanations of the steps involved.

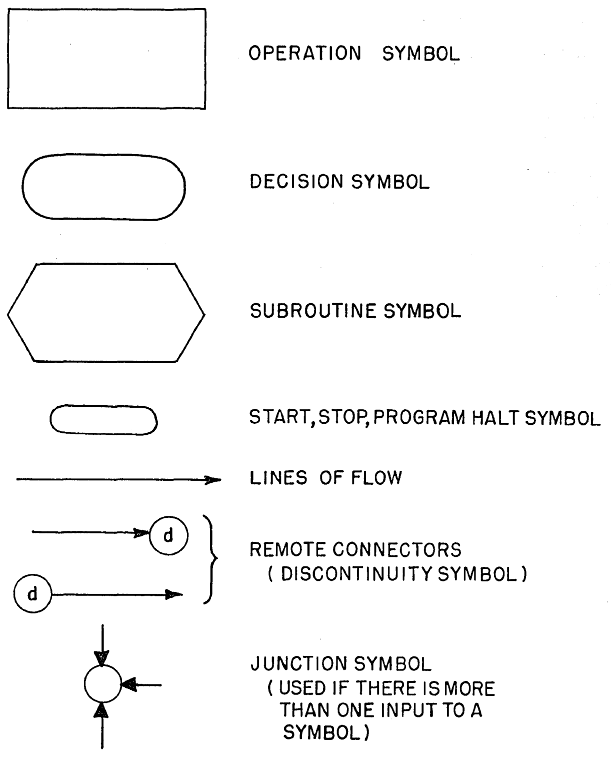
The second flow chart usually formulates the flow of computations between registers as the problem will be computed in the computer. This chart usually contains the instructions necessary for the data input, the instructions that operate on the input data to obtain the solutions, and the necessary instructions for the output of the results. Many problems are such that the second type of flow diagram will consist of many charts and/or diagrams, each a more detailed presentation of the preceding charts.

FLOW CHART SYMBOLS

To make it easier to formulate and understand flow diagrams, certain symbols have been accepted and are used throughout this discussion. The following list of symbols (fig. 8-2) gives the coder or programmer an example of the basic symbols used in constructing flow charts.

Lines of Flow

A solid line with an arrow touching the next element of the flow diagram usually is used to indicate the path to be followed by the computer; or, more precisely, the path to be followed by the coder (person who is formulating the computer instructions from the flow diagrams).


OPERATION SYMBOL
DECISION SYMBOL
SUBROUTINE SYMBOL
START, STOP, PROGRAM HALT SYMBOL
LINES OF FLOW
d
d
REMOTE CONNECTORS (DISCONTINUITY SYMBOL)
JUNCTION SYMBOL (USED IF THERE IS MORE THAN ONE INPUT TO A SYMBOL)

124.91

Figure 8-2.—Flow chart symbols.

Operation Symbol

The rectangular box usually contains a statement about a computer or mathematical operation. The content of the box may be a simple statement or a mathematical expression.

Decision Symbol

An oval is used to indicate a two-way decision. This symbol sometimes contains the operation code to indicate the type of decision to be made. For example, the operation code 04 written within the symbol designates the use of the Compare instruction to make an evaluation in the computer.

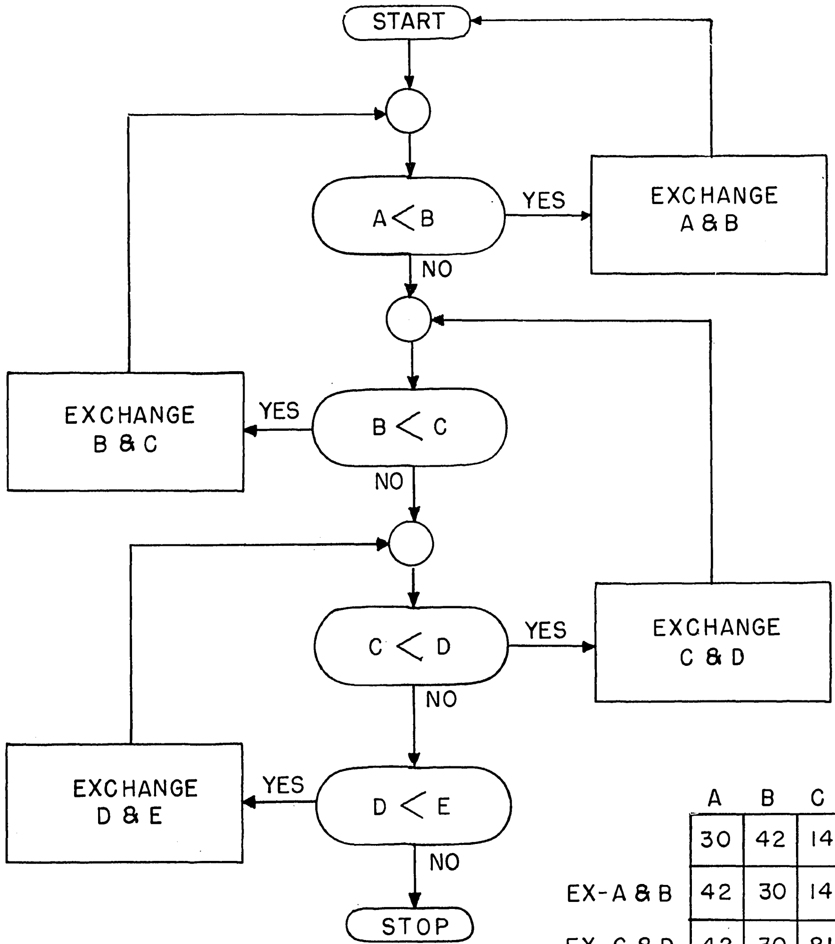
Connectors and Remote Connectors

To eliminate as much as possible the crossing of flow lines on a diagram, remote connectors are used to indicate a destination not easily reached in the diagram. Thus, the flow can be broken at a convenient point by terminating it in an arbitrary symbol which can be used to initiate the flow in another region of the diagram.

SAMPLE FLOW CHARTS

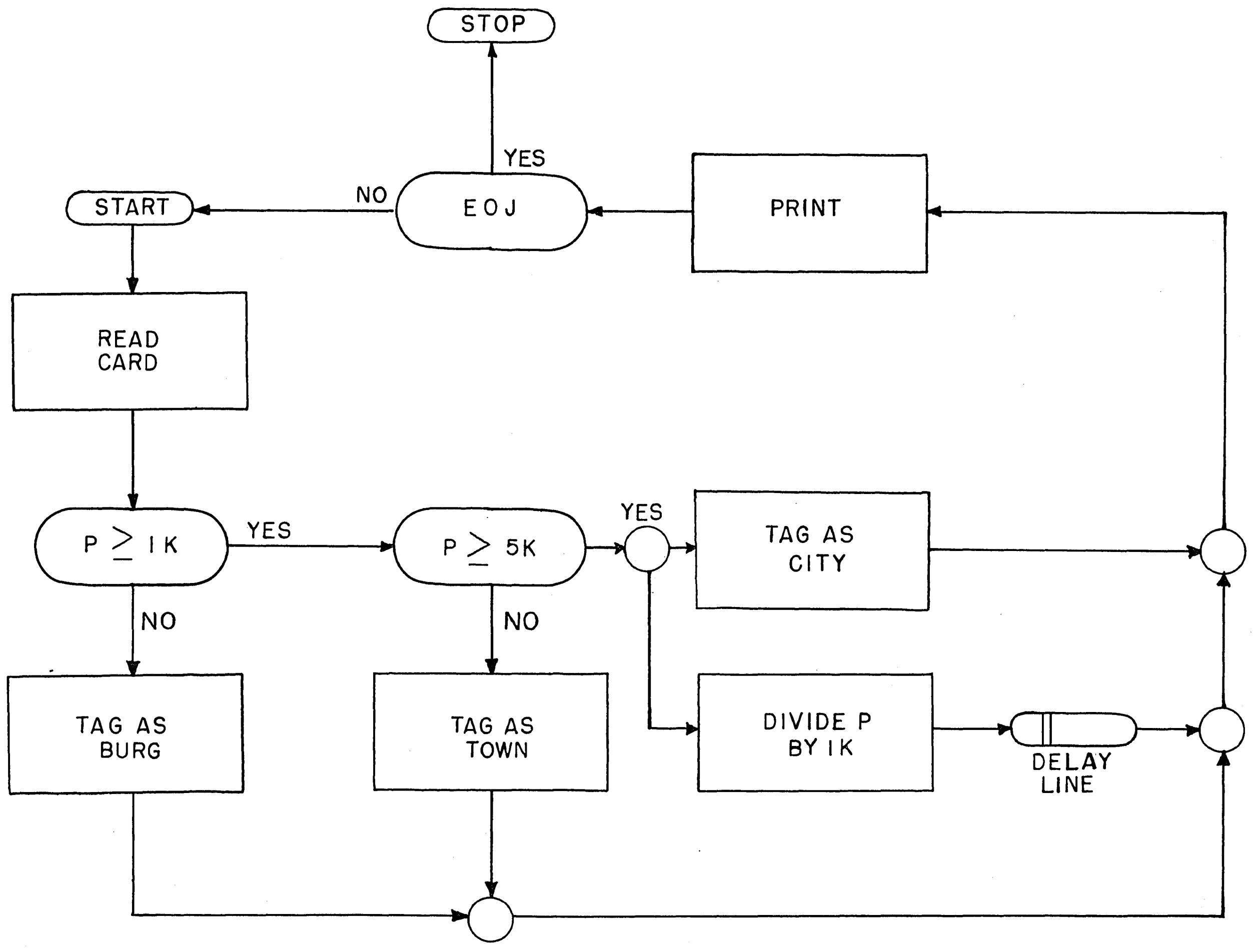
A simple example of the use of flow charts is shown in figure 8-3. The problem is to arrange five numbers in descending order regardless of their present arrangement. The numbers used in this example 30, 42, 14, 81, and 12, are arbitrarily selected and arbitrarily arranged. The table shows the results after making an exchange in the order of two numbers if and when an exchange is requested.

Another example of the use of flow charts is shown in figure 8-4. The problem is as follows: Determine the classification of a locality (town, burg, or city) based on the population of that locality. If the population is greater than or equal to 1,000 but less than or equal to 5,000 classify the locale as a town. If the population is less than 1,000, class as a burg. If the population is greater than 5,000, class as a city and determine



	A	B	C	D	E
	30	42	14	81	12
EX-A & B	42	30	14	81	12
EX-C & D	42	30	81	14	12
EX-B & C	42	81	30	14	12
EX-A & B	81	42	30	14	12

124.92
Figure 8-3.—Flow chart showing procedure for arranging numbers in descending order.



124.93
Figure 8-4.—Flow chart showing procedure for classifying cities according to population.

the relationship of the locality as compared to a town of 1,000 population. It is understood that the machine must be instructed to start, and to stop at the end of job (EOJ).

MACHINE CODING

The CP-642A/USQ-20 (V) computer is a self-modifying, single-address machine. Although this means that one reference or address is provided for the execution of an instruction, this reference can be modified automatically during a programmed sequence.

The references are modified by using the B-registers, 1 through 7, that contain any previously stored constants. In order to modify the address, the content of a selected B-register is added to the operand address designator.

A programmed address is coded using octal notation with each octal digit denoting three binary digits (a representative word is shown later). The instructions are read sequentially from magnetic core storage except after jump or skip instructions. Every instruction executed by the computer is transmitted from memory to the Z-register (see figure 8-1) and to the U-register. While the instruction is in the U-register, its components are translated to determine the exact method of executing the instruction.

SYMBOL CONVENTIONS

The following symbols are used in computer discussions:

α	:	any register or memory location
(α)	:	content of α
$(\alpha)_i$	:	initial content of α
$(\alpha)_f$	:	final content of α
α_n	:	the nth bit of a α
$(\alpha)_n$	:	the nth bit of the content of α
α_u	:	the upper 15 bits of α
α_L	:	the lower 15 bits of α
Y	:	the operand designator U_L
$\underline{Y}$	:	y + B_b (memory address of operand)
($\underline{Y}$)	:	contents of memory address $\underline{Y}$
y	:	the operand regardless of source
L(Y) (Q)	:	bit-by-bit multiplication; logical multiplication of Y_n and $(Q)_n$

SEQUENTIAL BREAKDOWN OF INSTRUCTIONS

Table 8-4 is a list of the instructions which can be performed by the computer. The two-digit numbers in the code column represent the function code and are written in binary coded octal form in the instruction (shown later).

Each of the instructions indicated by an asterisk in table 8-4 is used in a later discussion of simple programs and is analyzed briefly following the table. This analysis of instructions is intended for reference only.

The actions performed by each of the four sequences A, B, C, and D (not to be confused with the A, B, C, and D registers) are stated in a general sense earlier in this chapter.

The A sequence normally involves the same data transfer in the execution of each instruction. Normally, the A sequence operations consist of the following (fig. 8-1):

A Sequence

P adder $\longrightarrow$ S

S $\longrightarrow$ P

Read instruction $\longrightarrow$ U_u (from Zu)

Modify U_L according to b index designator

The data flow given for each instruction is not complete, but is concerned with the flow to and from the major registers (B, R, Uu, etc.).

INTERNAL PROGRAMMING CONCEPTS

Once the program is written and coded in an acceptable form, it is entered into the storage section of the computer. From this point, the computer, upon proper initiation, executes the instructions of the program. The instructions of the program are stored in the memory (storage) section of the computer in a sequential manner, such that the computer will first execute the instruction located at the lowest address of the program and proceed to the highest address. From its address location, the instruction is moved to the control section, where the computer analyzes the instruction to determine the method of execution. (The instruction is rewritten in memory in its original form and is therefore not altered in memory due to this process.) Normally the next instruction to be executed is located at the address that is, in value, one greater than the address of the previous instruction.

All operations in the CP-642A/USQ-20(V) computer are controlled by a 30-bit instruction word. The 30 bits of the word are divided into the sections shown below. Different bits of the word perform different functions and have been assigned symbols or designators for ease of reference. The normal instruction word bit positions are illustrated as follows:

f	j	k	b	y
29, 28, 27, 26, 25, 24;	23, 22, 21;	20, 19, 18;	17, 16, 15;	14,.... 00

A special instruction word used for input/output (I/O) operations is as follows:

f	$\hat{j}$	$\hat{k}$	b	y
29, 28, 27, 26, 25, 24;	23, 22, 21, 20;	19, 18;	17, 16, 15;	14,.... 00

TABLE 8-4.—REPERTOIRE OF INSTRUCTIONS.

CODE	FUNCTION		CODE	FUNCTION
00	(Fault Interrupt)		42	SUBTRACT LOGICAL PRODUCT
01	RIGHT SHIFT Q		43	COMPARE MASKED
02	RIGHT SHIFT A		44	REPLACE LOGICAL PRODUCT
*03	RIGHT SHIFT AQ		45	REPLACE A+ LOGICAL PRODUCT
*04	COMPARE A, Q, AQ		46	REPLACE A- LOGICAL PRODUCT
05	LEFT SHIFT Q		47	STORE LOGICAL PRODUCT
06	LEFT SHIFT A			
07	LEFT SHIFT AQ			
			50	SELECTIVE SET
*10	ENTER Q		51	SELECTIVE COMPLEMENT
*11	ENTER A		52	SELECTIVE CLEAR
*12	ENTER Bj		53	SELECTIVE SUBSTITUTE
13	EXTERNAL FUNCTION ON $C\hat{j}$		54	REPLACE SELECTIVE SET
*14	STORE Q		55	REPLACE SELECTIVE COMPLEMENT
*15	STORE A		56	REPLACE SELECTIVE CLEAR
16	STORE Bj		57	REPLACE SELECTIVE SUBSTITUTE
17	STORE $C\hat{j}$			
			60	JUMP (Arithmetic)
* 20	ADD A		*61	JUMP (Manual)
21	SUBTRACT A		62	JUMP ON $C\hat{j}$ ACTIVE INPUT BUFFER
22	MULTIPLY		63	JUMP ON $C\hat{j}$ ACTIVE OUTPUT BUFFER
* 23	DIVIDE		64	RETURN JUMP (Arithmetic)
24	REPLACE A+Y		*65	RETURN JUMP (Manual)
25	REPLACE A-Y		66	TERMINATE $C\hat{j}$ INPUT BUFFER
26	ADD Q		67	TERMINATE $C\hat{j}$ OUTPUT BUFFER
27	SUBTRACT Q			
			70	REPEAT
30	ENTER Y+Q		71	B SKIP ON $B\hat{j}$
31	ENTER Y-Q		*72	B JUMP ON $B\hat{j}$
32	STORE A+Q		73	INPUT BUFFER ON $C\hat{j}$
33	STORE A-Q			(without Monitor mode)
34	REPLACE Y+Q		74	OUTPUT BUFFER ON $C\hat{j}$
35	REPLACE Y-Q			(without Monitor mode)
*36	REPLACE Y+1		75	INPUT BUFFER $C\hat{j}$
37	REPLACE Y-1			(with Monitor mode)
			76	OUTPUT BUFFER ON $C\hat{j}$
40	ENTER LOGICAL PRODUCT			(with Monitor mode)
41	ADD LOGICAL PRODUCT		77	(Fault Interrupt)

* Denotes instructions to be used later.

TABLE 8-4.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

03 RIGHT SHIFT AQ [SHIFT (AQ) RIGHT BY Y]

This instruction shifts (A) and (Q) as one 60-bit register. The shift is to the right Y bit positions, with the lower bits of (A) shifting into the higher bit positions of (Q). The higher-order bits of (A) are replaced with the original sign bit as the word is shifted. Only the lower-order six bits of Y are recognized for this instruction. The higher-order bits are ignored.

B SEQUENCE	C SEQUENCE
Clear D	k $\neq$ 0, 4; Clear R, D$\rightarrow$R
k $\neq$ 0, 4; Shift Count$\rightarrow$D	Initiate Shift Sequence
k = 0, 4; U Adder$\rightarrow$D	

04 COMPARE: SENSE j: [$(A)_i = (A)_f$]

This instruction compares the signed value of Y with the signed value of (A) and/or (Q), but does not alter either (A) or (Q). Branch condition designator j is interpreted in a special way for this instruction, as follows:

j = 0:	Do not skip the next instruction.
j = 1:	Skip the next instruction.
j = 2:	Skip the next instruction if Y is less than or equal to (Q). ($Y \leq Q$)
j = 3:	Skip the next instruction if Y is greater than (Q). ($Y > Q$).
j = 4:	Skip the next instruction if (Q) is greater than or equal to Y and Y is greater than (A). ($Q \geq Y$ and $Y > A$)
j = 5:	Skip the next instruction if Y is greater than (Q) or if Y is less than or equal to (A). ($Y > Q$ or $Y \leq A$)
j = 6:	Skip the next instruction if Y is less than or equal to (A). ($Y \leq A$)
j = 7:	Skip the next instruction if Y is greater than (A). ($Y > A$)

B SEQUENCE	C SEQUENCE
Operand$^L\rightarrow$D	Sense j (for A)
	Clear X, A$\rightarrow$X
	Clear A, Q$\rightarrow$A
	Sense j (for Q)
	Clear A, X$\rightarrow$A

TABLE 8-4.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

10 ENTER Q $[Y \rightarrow Q]$

Clear the Q register, then transmit Y to Q.

B SEQUENCE	C SEQUENCE
Operand $\rightarrow$ D	Clear X, D $\rightarrow$ X
	Clear Q, X $\rightarrow$ Q
	Sense j

11 ENTER A $[Y \rightarrow A]$

Clear A, then transmit Y to A.

B SEQUENCE	C SEQUENCE
Operand $\rightarrow$ D	Clear X, D $\rightarrow$ X
	Clear A, X $\rightarrow$ A
	Sense j

12 ENTER B^j $[Y \rightarrow B_j]$

Clear the contents of B register j. Then transmit 15 bits of Y to B register j. Branch condition designator j is used to specify the selected B register for this instruction and is not available for its normal function.

B SEQUENCE	C SEQUENCE
Operand $\rightarrow$ D	$k \neq 0, 4$; Clear R, D $\rightarrow$ R
	Clear B_j, R $\rightarrow B_j$

14 STORE Q $[(Q) \rightarrow \underline{Y}]$

Store (Q) at storage address $\underline{Y}$ as directed by operand interpretation designator k. If k = 0, complement (Q). If k = 4, store in A.

B SEQUENCE	C SEQUENCE	D SEQUENCE
	Clear X, Q $\rightarrow$ X	Store as modified by k
		Sense j

15 STORE A $[(A) \rightarrow \underline{Y}]$

Store (A) at storage address $\underline{Y}$ as directed by operand interpretation designator k. If k = 4, complement (A). If k = 0, store in Q.

B SEQUENCE	C SEQUENCE	D SEQUENCE
	Clear X, A $\rightarrow$ X	Store as modified by k
		Sense j

TABLE 8-4.—REPERTOIRE OF INSTRUCTIONS— CONTINUED.

20 ADD A [(A)+Y $\rightarrow$ A]

Add Y to the previous content of the A register. D receives the exact number to be added to A.

B SEQUENCE	**C SEQUENCE**
Operand $\rightarrow$ D	**Clear X, Adder $\rightarrow$ X**
	Sense j from Adder
	Clear A, X $\rightarrow$ A

23 DIVIDE [(AQ/Y$\rightarrow$Q]

Divide (AG) by Y, leaving the quotient in Q register and the remainder in the A register. The remainder bears the same sign as the quotient. k = 7 should not be used in this instruction. J = 3 skip the next instruction if Q is negative.

B SEQUENCE	**C SEQUENCE**
If A is negative:	**If D is negative:**
Set Complement 1 flip-flop and initiate Complement AQ sequence	**Set Complement 2 flip-flop**
	Clear X, D $\rightarrow$ X
	Clear D, X $\rightarrow$ D
Operand$\rightarrow$D	**If D is positive:**
	Clear X, D $\rightarrow$ X
	Clear D, X' $\rightarrow$ D
	Clear K, Set K15
	Initiate Divide

31 ENTER Y - Q [Y - (Q)$\rightarrow$A]

Clear A. Then transmit (Q) to (A). Then subtract Y from (A). Finally, complement (A) if Y - (Q) $\neq$ +0.

B SEQUENCE	**C SEQUENCE**	
Clear A, Q$\rightarrow$A	**Clear X, Adder$\rightarrow$X**	
Operand'$\rightarrow$D	**Clear D, X'$\rightarrow$D**	**if A $\neq$ D**
	Clear X, D$\rightarrow$X	
	Clear A, X$\rightarrow$A	
	Sense j	

TABLE 8-4.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

36 REPLACE Y + 1 [$Y + 1 \longrightarrow \underline{Y}$ & A]

Clear A. Then set (A) = +1. Then add Y to (A). Then store $(A)_f$ at storage address $\underline{Y}$.

B SEQUENCE	C SEQUENCE	D SEQUENCE
Clear A, +1 ⟶ A	Clear X, Adder ⟶ X	Store as modified by k
Operand ⟶ D	Clear A, X ⟶ A	
	Sense j	

61 JUMP (Manual) [JUMP TO ADDRESS $\underline{Y}$ IF j SATISFIED]

This instruction clears program address register P and enters a new program address in P for certain conditions for manual JUMP selections. Branch condition designator j is interpreted in a special way for this instruction and determines the condition under which a jump in program address occurs. If the jump condition is not satisfied, the next sequential instruction in the current sequence is executed in a normal manner. If the jump condition is satisfied, $\underline{Y}$ becomes the address of the next instruction and the beginning of a new program sequence, as listed below.

Program execute may be stopped by certain STOP selections upon execution of this instruction. Branch condition designator j specifies which selections are effective.

j = 0:	Execute jump (unconditional).
j = 1:	Execute jump if JUMP 1 is selected.
j = 2:	Execute jump if JUMP 2 is selected.
j = 3:	Execute jump if JUMP 3 is selected.
j = 4:	Execute jump. Stop computation.
j = 5:	Execute jump. Stop computation if STOP 5 is selected.
j = 6:	Execute jump. Stop computation if STOP 6 is selected.
j = 7:	Execute jump. Stop computation if STOP 7 is selected.

B SEQUENCE	C SEQUENCE	
Operand ⟶ D	k ≠ 0, 4;	
	Clear R, D ⟶ R	
	Clear P, R ⟶ P	j = 1, 2, 3 and
	Clear p (0 ⟶ p)	JUMP selected

TABLE 8-4.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

Clear P, R ⟶ p Clear p (0 ⟶ p) Stop at beginning of next B sequence.	j = 4
Clear P, R ⟶ P Clear p, 0 ⟶ P Stop at beginning of next B sequence if option j set.	j = 5, 6, 7 and STOP selected

65 RETURN JUMP (Manual) [JUMP TO $\underline{Y}$ + 1 & (P) + p ⟶ $\underline{Y}_L$ IF j SATISFIED]

This instruction executes a return jump sequence for certain conditions of manual JUMP or STOP options. Branch condition designator j is interpreted in a special way for this instruction, and determines the conditions under which the return jump sequence is executed. If the return jump condition is not satisfied, the next sequential instruction in the current sequence is executed in a normal manner. If the return jump condition is satisfied, as listed below, the following sequence is performed: Store (P) (the number in the program address counter which denotes the instructions being executed) + p (the amount by which (P) is increased after each instruction is executed) in the lower half of memory address Y (the operand which is the first address in the subroutine.) Then jump to Y + 1.

j = 0:	Execute return jump (unconditional).
j = 1:	Execute return jump if JUMP 1 is selected.
j = 2:	Execute return jump if JUMP 2 is selected.
j = 3:	Execute return jump if JUMP 3 is selected.
j = 4:	Execute return jump, then stop computation.
j = 5:	Execute return jump. Stop computation if STOP 5 is selected.
j = 6:	Execute return jump. Stop computation if STOP 6 is selected.
j = 7:	Execute return jump. Stop computation if STOP 7 is selected.

B SEQUENCE	C SEQUENCE	D SEQUENCE
Operand ⟶ D	k = 0, 4	If j satisfied:
	Clear R, D ⟶ R	Clear S, U Adder ⟶ S
	Clear R* & U_L	(P) + p ⟶ Z_L ⟶ M_L
	R ⟶ U_L	Clear P, S ⟶ P
		+1 ⟶ p

TABLE 8-4.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

Note

If a return jump immediately follows recognition of an interrupt by the control section (that is, if the return jump is the instruction stored at the Interrupt Entrance register), it is described as follows: Store P + 0, 1, or 2 in the lower half of memory address $\underline{Y}$. Then jump to $\underline{Y}$ +1. The p designator controls the modification of (P) and is set up by the instruction preceding the return jump caused by the interrupt. Therefore, the return jump causes the storage of the address of the next instruction that would have been executed if the interrupt had not occurred. The general description of the return jump is the jump to $\underline{Y}$ +1, with the understanding that in non-interrupt cases, p is set to "one", which causes the storage of P + 1 in $\underline{Y}$.

72 B JUMP ON B^j $[(B)_j = 0$, EXECUTE NI: $(B)_j \neq 0$, $(B)_j - 1 \longrightarrow B_j$, JUMP TO $\underline{Y}]$

If the content of B register j is nonzero, execute a jump to program address $\underline{Y}$. Reduce the content of B register j by one. If the content of B register j is zero, proceed to next instruction in a normal manner. Branch condition designator j is used to designate the selected B register in this instruction and is not available for its normal function. If the jump condition is satisfied, the lower-order 15 bits of Y become the address of the next instruction and the beginning of a new program sequence. The higher-order 15 bits of Y are not used in this instruction.

B SEQUENCE	C SEQUENCE
Operand $\longrightarrow$ D	$k \neq 0, 4$
	Clear R, D $\longrightarrow$ R
	Clear R*, $B_j \longrightarrow$ R*
	$-1 \longrightarrow U_L$
	$B_j \neq 0$
	Clear P, R $\longrightarrow$ P
	Clear p ($0 \longrightarrow$ p)
	Clear R, U Adder $\longrightarrow$ R
	Clear B_j, R $\longrightarrow B_j$

An instruction word is composed of the following components:

f — Function Code designator (29 — 24)
J — Branch Condition designator (23 — 21)
k — Operand Interpretation designator (20 — 18)
b — Address Modification designator (17 — 15)
y — Operand Address designator (14 — 00)

The instruction word is placed in the U register and its outputs are applied to translator circuits which interpret the bits. Further simplification of the word is attained by using the octal numbering system because of the ease of conversion. The binary notation of an instruction could be as follows:

```
 f      j    k    b            y
001 100 010 011 101 000 010 001 111 110
```

Octal notation of the same instructions would be: 14 2 3 5 02176. The designators can be stated as: f = 14, j = 2, k =3, b = 5, and y = 02176. The translation of these designators determines the exact method of executing the instruction.

The designators used later in developing simple programs are defined in table 8-5.

Table 8-5.—Interpretation of Word Modifiers (Designators).

j = 2	Skip the next instruction if (Q) is positive.
j = 3	Skip the next instruction if (Q) is negative.
j = 5	Skip the next instruction if (A) is nonzero.
j = 6	Skip the next instruction if (A) is positive.
k = 1	Store in M_L leaving M_u undisturbed.
k = 1	Read: Y/u = 0's; $Y_L = (Y)_L$ (Operand in M_L).
k = 3	Store in M (Store instruction).
k = 3	Y = (Y). (Operand in M) (Read instruction).
k = 7	Y = (A). (Operand in A register) (Read instruction).
b = 1	Add (B^1) to y.

PROGRAM CONSTRUCTION

Two examples illustrating the method of constructing simple programs are considered in the following discussion. Frequent reference to tables and illustrations presented in this section and in earlier portions of this chapter will be necessary to facilitate a full understanding.

MANUAL BLOCK TRANSFER

The first example of programming is considered by assuming the following problems:

Problem: TRANSFER THE WORDS IN ADDRESSES 00020-00025 in MEMORY TO ADDRESSES 00030-00035.

We first analyze the problem, noting that data from one section of memory is to be transferred to another section of memory. The use of an index register (B-register) will be helpful in counting the number of transfers and in incrementing the various addresses involved. A knowledge of the actions involved for a 61 function code with a j = 4 designator (see table 8-4 and item 61) is helpful to the programmer since he knows that after transfer of the required data the j = 4 designator will cause the action to stop after a return jump to the initial address.

The next step is to develop a flow chart (fig. 8-5). The first step after starting is to enter a number in the index register (B1 in this case) which is one less than the number of transfers desired. (This is necessary since the count from 00000 to 00001 causes one transfer.)

This action is followed by entering the data in address X (the actual address numbers are not significant at this point) into the Q register and then storing this data (from Q) into address Y. This completes the transfer of one data word.

The number in B1 is checked to determine if all transfers have been completed (B1 = 0?). A "no" answer indicates that all transfers have not been completed. This initiates the step to reduce the number in B1 by 1 (B1 - 1) and the next address is read into the Q register.

The steps just explained are repeated until a "yes" answer is produced from the interrogation (B1 = 0?), whereupon a jump and stop command is executed.

It is now possible to write the program as shown on following page.

PROGRAM
(BLOCK TRANSFER SUBROUTINE)

Relative Address	Instruction	Function
m1	12100 N	Enter B1 with number of words to be transferred less one.
m2	10031 y1	(y1 equals lowest address of block to be transferred.) Enters word from highest address to be transferred.
m3	14031 y2	(y2 equals lowest address of new storage location.) Stores word at highest address of new location.
m4	72100 m2	If B1 = 0, read NI; if B1 ≠ 0, B1-1 and jump to m2. (This operation stops transfer when all words have been transferred.)
m5	61400 m1	Jump to beginning of routine and STOP 4.

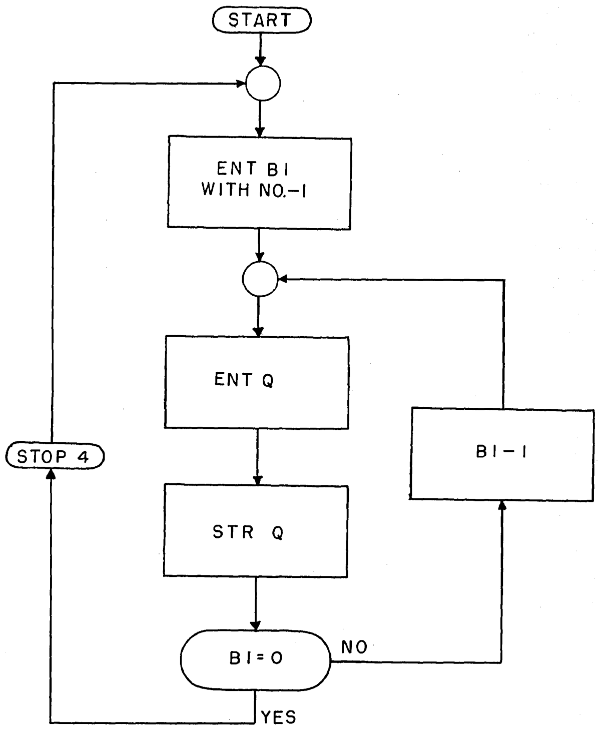
The relative address column does not show actual addresses in memory but merely represents the order in which the instructions are to be executed. The relative address method of programming is discussed presently.

The first instruction in the program (12100 N) is shown below in binary and octal numbers.

Designator

f	j	k	b	y
Binary Notation				
001 010	001	000	000	000000000000101
Octal Notation				
1 2	1	0	0	N

The instructions are represented in binary numbers within the computer but are written in octal form in this discussion. The numerical value (in binary) of each designator in the 12100 N instruction is shown. The "f" designator always contains 6 bits; the j, k, and b designators each contain 3 bits; and the y designator (the operand or address of the operand) contains 15 bits. The interpretation of the designators in the instruction is found in table 8-5 or in the detailed description of the instructions in table 8-4.



124.94
Figure 8-5.—Block transfer routine.

The instruction (12100 N) supplies the computer with a function code, 12, ENTER B^J instruction (see table 8-4 and item 12) where B^J is interpreted as the B register (fig. 8-1) specified by the j designator (001) in the instruction. The letter N represents the number of transfers requested, less one. Because this subroutine can be used for the transfer of any number of addresses at any number of points in the program, the value of N (15 bits) is assigned for each use.

The second instruction (10031 y1) at relative address m2 has a 10 function code (see table 8-4 and item 10) which would normally cause the data located in address y1 in memory to be read from memory into the Q register. The data at address y1 (00020 in this case) however, (as illustrated in figure 8-6), is increased by the amount of the number N in the B register (5) in accordance with the b-designator of 1 in the instruction (see table 8-5) before it is transferred to the Q register. Likewise, the instruction at relative address m3 (14031y2) would normally cause the Q register content to be stored in memory address y2 (00030). The b-designator of 1 in this instruction, however, causes y2 to be incremented by the content of the B register (5) in this case. Thus, the data from the lowest memory address (00020) is transferred to the lowest memory address in the highest of the addresses to which the block of data is to be transferred (00030).

The instruction at address m4 (72100 m2) check (B1 = 0) yields a "no" result and the number in the B1 index register is reduced by 1 to become (N = 4).

This completes the transfer of one cycle or loop. The actions can be briefly stated as follows:

1st Cycle—TRANSFER WORD FROM y1 (00020) + N(5) TO y2 (00030) + N(5) AND B1-1 (N = 4).

The 72100 m2 instruction (see detailed operations for a 72 function code, table 8-4) causes a jump to relative address m2 after the execution of each cycle until the B register content is 0. The actions in the 2nd, 3rd, 4th, 5th, and 6th cycles are explained below:

2nd Cycle—TRANSFER WORD FROM y1 (00020) + N (4) TO y2 (00030) + N(4) AND B1-1 (N = 3)

3rd Cycle—TRANSFER WORD FROM y1 (00020) + N(3) TO y2 (00030) +N(3) AND B1-1 (N = 2)

4th Cycle—TRANSFER WORD FROM y1 (00020) + N(2) TO y2 (00030) + N(2) AND B1-1 (N = 1)

5th Cycle—TRANSFER WORD FROM y1 (00020) + N(1) TO y2 (00030) + N(1) AND B1-1 (N = 0)

6th Cycle—TRANSFER WORD FROM y1 (00020) + N(0) TO y2 (00030) + N(0) AND JUMP (B1 = 0) AND STOP 4.

RELATIVE ADDRESSING

In a coding routine it is often preferable to assign relative address locations to quantities referred to in the program. For example, it may be convenient to postpone the assignment of absolute addresses to instruction data, constants, or temporary storage until coding of the entire problem has been completed. Similarly, it may not be desirable to assign locations to subroutines until coding is completed and it is known what the storage requirements are for each of the sections of the program.

It is possible to postpone the assignment of absolute addresses by coding with relative addresses as was done in the program for the block transfer routine just described. In this case, the addresses are relative to the address m.

The alphabetic character m can denote any address allowable in the computer. For example, it can be assigned the address of 01000, where 010 denotes m and 00 the relative address. The numerals following an alphabetic character in a relative addressing scheme are usually interpreted as being additive; e.g., if m denotes 01000, then m12 denotes 01012, as follows:

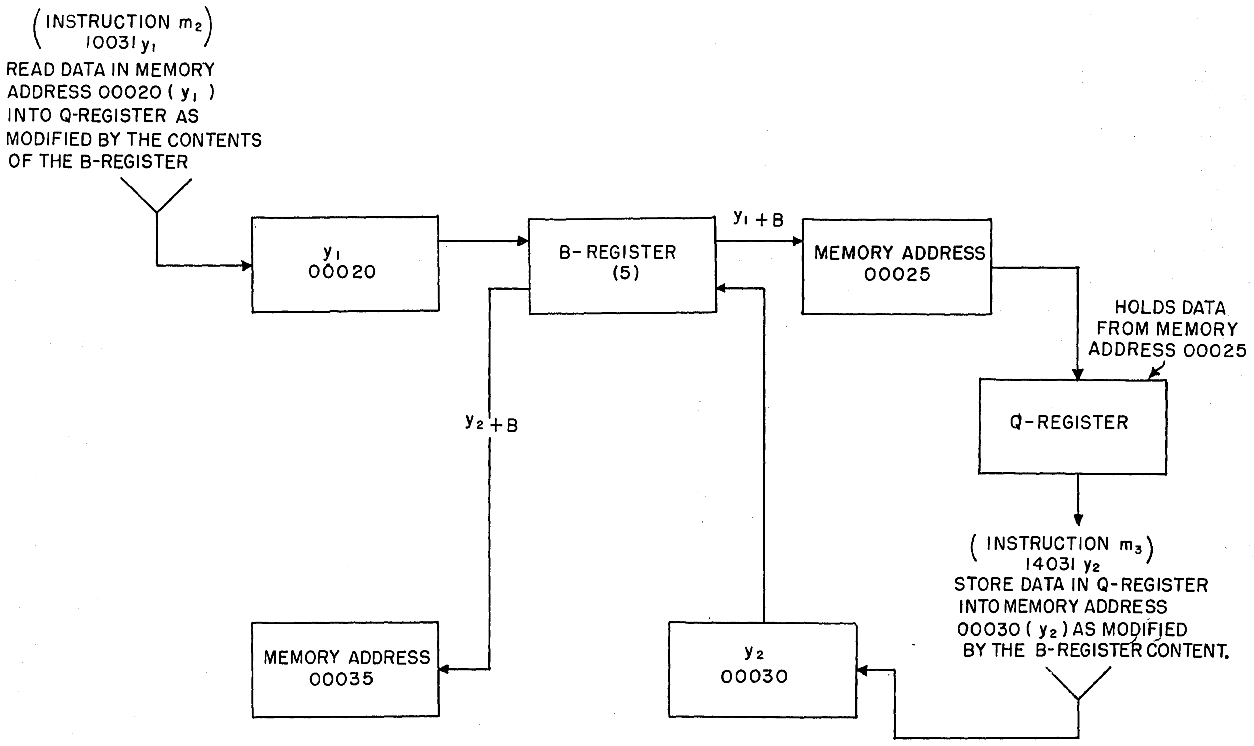
$$\begin{array}{rl} m = & 01000 \\ \underline{+} & \underline{12} \\ m12 = & 01012 \end{array}$$

Nearly all alphabetic characters are usually allowable in relative addressing schemes. A few exceptions are the reservations of the letter A, denoting the Accumulator, and the letter Q, denoting the Q register. Each new alphabetic character used in a program may indicate a new region in the storage of the computer. By using relative addresses it becomes comparatively easy to assign segments of a problem to various regions, with the routine assigned to each region performing a separate calculation or function.

QUOTIENT ROUNDOFF SUBROUTINE

The flow chart for a quotient roundoff subroutine resulting from the division of two positive numbers is shown in figure 8-7. A subroutine, as described earlier in this chapter, can be used as many times as necessary in a given program to produce the solution to a specific type of arithmetic operation.

To make use of a subroutine, the main routine must be interrupted and a return jump provided to the correct entry point of the subroutine. The subroutine, in return, must provide an exit back to the main routine. If in a subroutine the address of the first instruction to be executed is, for example, at 30001; the y (address) portion of the subroutine exit jump at address 30000 is set to the address of the instruction following the exit in the main routine.


(INSTRUCTION m2) 10031 y1
READ DATA IN MEMORY ADDRESS 00020 (y1) INTO Q-REGISTER AS MODIFIED BY THE CONTENTS OF THE B-REGISTER
y1 00020
B-REGISTER (5)
y1 + B
MEMORY ADDRESS 00025
HOLDS DATA FROM MEMORY ADDRESS 00025
Q-REGISTER
y2 + B
(INSTRUCTION m3) 14031 y2
STORE DATA IN Q-REGISTER INTO MEMORY ADDRESS 00030 (y2) AS MODIFIED BY THE B-REGISTER CONTENT.
MEMORY ADDRESS 00035
y2 00030

124.95

Figure 8-6.—Memory data transfer using B-register (1st cycle).

For example, assume that address 502 in the main routine initiates a return jump to a subroutine at address 30001. The instruction at address 502 must be coded so that it will cause the number of the next instruction in the main routine (503 in this case) to be placed in the address portion of the instruction at 30000 in the subroutine. The instruction at address 30000 is coded so that it becomes a jump instruction to address 503.

At the completion of the subroutine, or at any point in the subroutine which signals an exit is needed in order to re-enter the main routine, the computer will be instructed to select the instruction at address 30000 of the subroutine as the 'next instruction. This, in turn, causes a jump to address 503 of the main routine. In this way the subroutine may be entered from any point in the main program by programming a return jump.

As stated earlier, the problem is to round off the quotient resulting from the division of two positive numbers. The divisor, the dividend, the quotient, and the remainder are all variables, but are known for a given operation.

By dividing the remainder R (obtained from a given division) into the divisor D, and looking at this quotient to see if it is less than or equal to two, we can round off to the next highest integer. If the new quotient is greater than two, then leave the quotient of the original division as it is. (Note that the process of examining $\frac{D}{R}$ is the inverse of making an examination of the fraction $\frac{R}{D}$ (as is more common), where R is the remainder and D the divisor. The larger the quotient obtained as a result of the $\frac{D}{R}$ procedure, the less significant is the remainder in the roundoff of the main quotient.)

The first step in writing a subroutine to make the roundoff is to lay out the flow diagram. It may be necessary to make several flow diagrams before proceeding with the next step. Figure 8-7 shows the basic operations necessary to accomplish quotient roundoff. This chart can be further amplified as shown in figure 8-8. It is now possible to go directly into coding and programming. In some cases, it is desirable to have a more detailed flow chart as shown in figure 8-9.

The following list of instructions derived from table 8-4 is the program to execute the steps in the roundoff subroutine as they are set up in the detailed flow chart in figure 8-9. The instruction which initiates the divide operation (23 330 c shown below), the instruction which provides the exit to the subroutine (65 000 z), and the instruction 65 000 -) for the divide overflow routine, represent the section of the main routine or program which is concerned with the execution of the roundoff subroutine. Relative addressing is used.

MAIN PROGRAM ROUTINE

ADDRESS	INSTRUCTIONS OCTAL NOTATION	NOTES
00501	23 330 c(an address in memory)	Divide, skip if overflow
00502	65 000 z(relative address)	Return jump to roundoff routine
00503	65 000 -	Return jump to overflow routine

SUBROUTINE

RELATIVE ADDRESS	INSTRUCTIONS	NOTES
z + 0	61 000 00503	61 function code means JUMP (MANUAL). The word "manual" used with the JUMP instruction is to be interpreted as an unconditional jump. The j, k, and b designators are not needed in this instruction. The last five digits of this instruction (as well as all instructions) carry the y designator. As was stated earlier, the address portion of the instruction at address 00000 (first address) of the subroutine must contain the address to which the computer operations are to be returned after the completion of the subroutine, or after it is determined that certain conditions exist which necessitate the return to the main routine. Thus, address 503 is indicated in this instruction as the address to which the operations are to be resumed in the main routine.
z + 1	20 500 00000	20 function code means ADD A (see table 8-4 and item 20 of detailed instructions). This instruction adds Y (00000) to the previous content of the A register. (The A register holds the remainder (block 1) derived from a divide operation (fig. 8-9). The j=5 designator causes a skip of the next instruction if (A) is nonzero (A $\neq$ 0). It follows that if (A) is equal to zero, execute the next instruction. Stated another way, if A = 0, there is no remainder and the roundoff subroutine is not necessary. Thus execute the next instruction (which is an exit instruction back to the main routine). If A $\neq$ 0, skip the next instruction (the exit instruction) and continue with the roundoff subroutine.

SUBROUTINE Cont'd

RELATIVE ADDRESS	INSTRUCTIONS		NOTES
z + 2	61 010	z + 0	61 function code, JUMP (MANUAL). This instruction is interpreted as jump to address specified by operand Y. The operand, in this case, is shown as z + 0 (the first address of the subroutine). The k designator (1) table 8-5 causes the address portion of the instruction to be read from memory. When this instruction is executed, the next instruction to be executed will be at address z + 0 followed by the instruction at address 503 of the main routine.
z + 3	14 030	e	STORE Q instruction (see block 2). This instruction stores (Q) the quotient at storage address Y as directed by the operand interpretation designator k (table 8-5). (See tables 8-4 and 8-5 and item 14 of detailed instructions.) The operand refers to an address e in storage into which (Q) is stored.
z + 4	15 030	v	STORE A instruction. Store (A), the remainder (block 2), at address specified in the operand (v in this case).
z + 5	11 000 00000		ENTER A instruction. (See table 8-4 and item 11 detailed instructions.) The ENTER A (see block 3) instruction represents a CLEAR A instruction since the process of entering A is accomplished by clearing the A register and entering the operand, Y. If the operand is 00000 (as is true in this case) the A register will not contain a number, a condition which may be described as cleared.
z + 6	10 030	c	ENTER Q (block 4). This instruction enters the operand , Y (represented in this instruction as the divisor c) into the Q register.
z + 7	23 030	v	DIVIDE AQ/Y (block 5). This instruction causes the content of the A and Q registers (combined as one 60-bit register for divide operation) to be divided by Y, the operand. This step executes Divide AQ/A_i as indicated in the flow chart (fig. 8-9) where A_i is the operand, Y.
z + 10	04 300 00003		COMPARE, SENSE j. instruction. The j designator, 3, is interpreted by the control unit as follows: skip the next instruction if Y is greater than or equal to (Q), the quotient (block 6). It follows then that if Y is less than (Q) the next instruction must be executed.

SUBROUTINE Cont'd

RELATIVE ADDRESS	INSTRUCTIONS		NOTES
z + 11	61 010	z + 0	Same as instruction at relative address z + 2 of this subroutine.
z + 12	04 200 00002		COMPARE SENSE j instruction (block 7). The j designator, 2, is interpreted by the control unit as follows: Skip the next instruction if Y is less than or equal to (Q). Stated in the terms of the flow chart, is Q $>$ 2, (Y). A "no" answer indicates that Q (the quotient) is less than 2 and that roundoff is necessary.
z + 13	61 000	z + 20	JUMP (MANUAL) instruction. This instruction executes the "add 1" procedure after it has been determined in the previous instruction that a quotient roundoff is necessary. The jump to the operand indicated in this instruction (z + 20) causes a 1 to be added to the initial quotient which was stored at address e in memory during the execution of the instruction at address z + 3.
z + 14	20 570 00000		ADD A instruction (block 8). This step is used to determine if there was a REMAINDER obtained from the divide step AQ/A_i in the flow chart. The A register is cleared, the remainder from the AQ/A_i divide is transferred to A, and a check is made to determine if (A) is 0. If A = 0, it is known that the dividend of the AQ/A_i divide is the same as the divisor and that no remainder exists from this division. The value of Q is greater than 2 but less than 3. In the case where A = 0, the remainder is the same value. Thus, AQ/A_i = 1 and it is necessary to increase the original quotient by 1. The j designator (5) is interpreted as follows: Skip the next instruction if (A) is nonzero, (A) $\neq$ 0. If (A) is 0, the next instruction is executed.
z + 15	61 000	z + 20	JUMP (MANUAL) instruction. This instruction is the same as the instruction z + 13 in this subroutine. The action is to add 1 to the quotient as was commanded by the previous instruction at z + 14.
z + 16	03 000 00036		RIGHT SHIFT instruction (block 9). This instruction shifts (A) and (Q) as one 60-bit register. The shift is to the right Y bit positions (36 in this case) with the lower bits of (A) shifting into the higher bit positions of (Q). The higher order bits of (A) are replaced with the original sign bit as the word is shifted. Only the lower order six bits of Y are recognized for this instruction. The right shift prepares the AQ registers for a subtract operation to be accomplished in the next instruction.

SUBROUTINE Cont'D

RELATIVE ADDRESS	INSTRUCTIONS		NOTES
z + 17	31 630	v	ENTER Y - Q instruction (block 10). Note from instruction z + 4 that Y at address v in memory stores the original remainder. Also recall that the previous instruction right-shifted the remainder from the second division (AQ/A_i) into the Q register. Thus a check is made in the execution of this instruction to determine if the second divisor (Ai, the original remainder) is greater than (Q), which is the second remainder. If the divisor is greater than the remainder, it is not necessary to round off the original quotient. A larger remainder than quotient will produce a negative result, "yes". A "Yes" answer (a positive answer) indicates that the divisor will go into the remainder and it is thus necessary to increase the original quotient by 1. This action is satisfied in the next instruction. The j designator (6) is interpreted as follows: Skip the next instruction if (A) is positive. The k designator (3) is interpreted as follows: Read the operand Y from memory address v.
z + 20	36 030	e	REPLACE Y + 1 instruction (block 11). This instruction clears the A register, after which it sets the A register contents to +1. The operand e, the original quotient, (see instruction at address z + 3) is then entered (added) to (A). The k designator (3) causes the final content of A to be placed in memory address e.
z + 21	61 010	z + 0	JUMP (MANUAL) instruction. This instruction signals the end of the subroutine, and the next instruction is taken at address z. The operand of this instruction (as explained) contains the address of the next instruction in the main routine thereby causing the exit from the subroutine.

The next step in the process is to convert these relative addresses to actual memory locations as shown below:

MAIN PROGRAM

Address	Instruction
01501	23 330 00100
01502	65 000 05000
01503	65 000 -----

SUBROUTINE

05000	00 000 00000
05001	20 500 00000
05002	61 010 05000
05003	14 030 05050
05004	15 030 05051
05005	11 000 00000
05006	10 030 01100
05007	23 030 05051
05010	04 300 00003
05011	61 010 05000

SUBROUTINE Cont'D

```
05012        04 200 00002
05013        61 000 05020
05014        20 570 00000
05015        61 000 05020
05016        03 000 00036
05017        31 630 05051
05020        36 030 05050
05021        61 101 05000
```

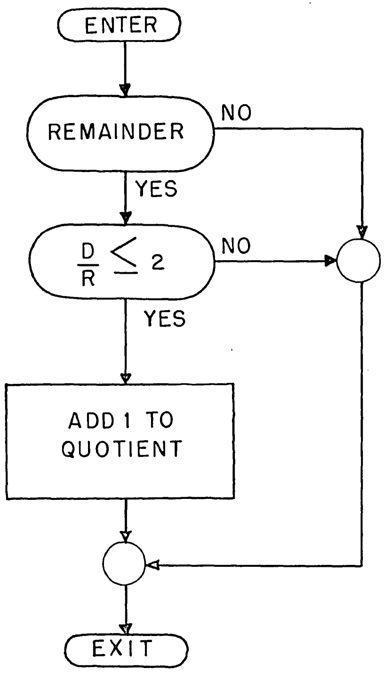
COMPILERS

In order to write a program for a digital computer, the programmer must be able to communicate with the machine. This communication must be through numbers—the only language the machine can understand. Since the programmer has probably seldom used numbers as a means of communication, a serious handicap exists between man and the machine.

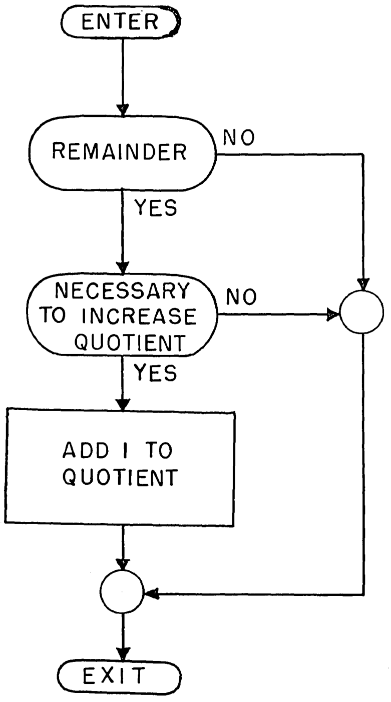
Compilers are devices which accept the language of the program and translate it into numbers comprehensible to the machine. NELIAC is one such compiler. This compiler is used with NTDS and is discussed in a later training course.

The name NELIAC is derived from the much longer terms "Navy Electronic Laboratory International Algol Compiler." The algol (fixed) language concept was conceived in recent years. Its primary purpose is to standardize the coding of scientific problems on computers and thereby eliminate all existing languages and dialects that are common to only a few computers. In its ultimate state, it is to be a language (oriented to the solutions of problems by numerical algorithms) that will be acceptable to any and all scientific computers.

Thus, the compiler is, in itself, a program of statement translators. For example, finding the square root, of AxBxC, is generally a lengthy process in the computer program requiring several machine coded programmed instructions. If the compiler program contains a "SQUARE ROOT" routine the "ROOT" statement, when requested, will take previously defined operands and extract the square root. This then requires only a single compiler statement.

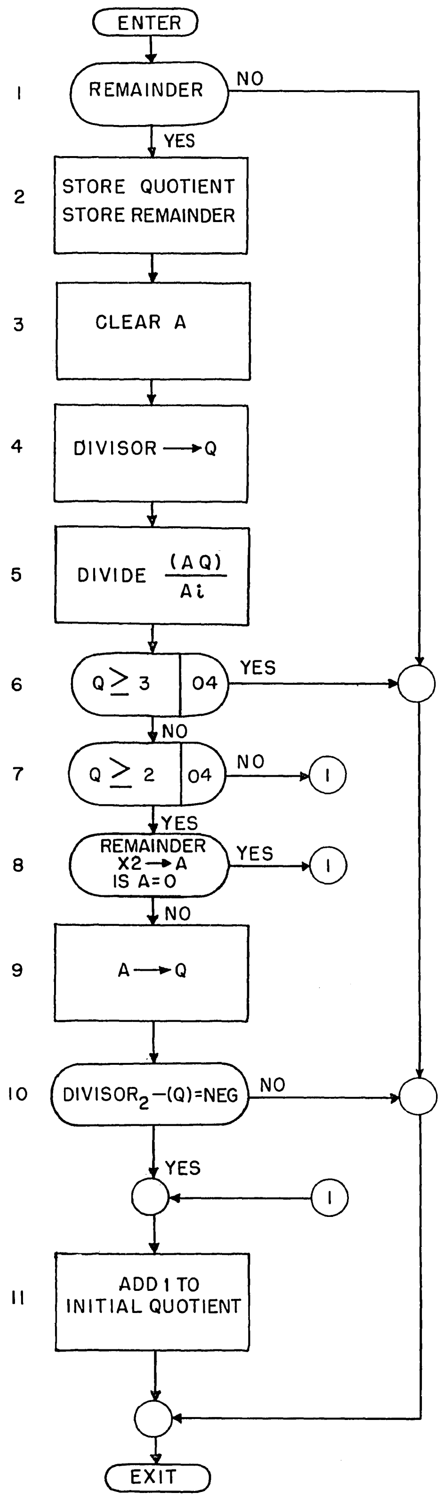


124.97
Figure 8-8.—Quotient roundoff subroutine (amplified diagram).



124.96
Figure 8-7.—Quotient roundoff subroutine (basic diagram).

Another feature of the compiler is its use of symbolic or relative addresses for assigning specific memory addresses. Future references to these symbolic addresses will cause the data in each selected memory address to be used as an operand. Thus, if symbolic addresses A, B, and C represent values of 3535, 2525, and 1515, respectively, these values will be stored at specific address in memory. Now assume that an AxB/C statement is read in by the compiler program. The compiler will generate the machine code to locate the operands, perform the designated operation, and display the final result.



124.98

Figure 8-9.—Quotient roundoff subroutine (detailed diagram).

CHAPTER 9
ANALOG-DIGITAL AND DIGITAL-ANALOG CONVERSIONS

There are three general types of computers, as pointed out in Chapter 2 of this course. These are: (1) digital, (2) analog, and (3) hybrid. The selection of a particular type of computer to perform a certain type of operation is dependent upon several factors.

One factor involved in this selection is the method and source of input information. For example, received from a radar set, an airspeed probe, or a shaft position are readily accepted and worked upon by an analog computer, since this type of data is analog in nature. Conversely, the type of data obtained from a ballistic table, a census tabulation, or a logistics manual are most readily accepted and worked upon by a digital computer, since this type of data is digital in nature.

A second factor which determines the selection of a specific type of computer is the methods used for computations. Analog computers cannot function as rapidly as digital computers, consequently, it is generally more desirable to use a digital computer where large amounts of data are to be processed, provided all other conditions can be satisfied.

Digital computers are generally more versatile than analog computers. Analog computers are designed to solve a specific problem. In practice then, it would be necessary to have as many types of analog computers as there are analogous problems to be solved. This limitation of analog computers gives rise to the use of analog-digital converters.

ANALOG-TO-DIGITAL CONVERSION

An analog-to-digital converter is defined as a device that receives an analog input and supplies a digital output. A converter of this type is capable of accepting instantaneous values of a constantly changing variable and expressing these values in incremental form. The converter may or may not perform some computation in addition to its primary function of data conversion.

The process of analog to digital conversions is 'not uncommon. Our minds make hundreds of analog-digital conversions each day. For example, if you ask a young child what time it is, he will probably look at a clock and tell you that the long hand is near the six and the short hand is near the four. The child is performing the function of data transmission, not data conversion. He is accepting the instantaneous value of the constantly moving hands of the clock, but he is transmitting the information in analog, not digital, form. The data which the child transmitted is converted into numerical values, in this case, four thirty. You, the inquirer, are therefore performing the analog-to-digital conversion, the child is not.

Care must be taken to avoid confusing truly digital read-outs with others that merely appear to be digital. The needle of a voltmeter or the pointer on an automobile speedometer provides analog data by indicating the instantaneous value of volts or miles per hour in a constantly changing manner. It is only by reading these meters and assigning numerical values that a truly digital read-out is obtained. Just as in the example of the clock, the human observer is serving as the analog-to-digital converter. Similarly, the type of data that is obtained from a micrometer scale or from the mercury column of a thermometer is analog in nature, until converted by the reader. The accuracy of each of these devices is, in large measure, a function of the ability of the reader to convert the information accurately from analog to digital form.

On the other hand, mechanical counters such as those commonly used to count the number of people entering a building provide a true digital read-out. No great amount of skill is required for reading such a device because its accuracy does not depend on the reader, but upon its design. If each click of the counter corresponds to a count of one, no amount of skill on the part of the reader will increase its inherent precision.

MECHANICAL TECHNIQUES FOR ANALOG TO DIGITAL CONVERSION

Probably the simplest method of converting analog information, such as shaft rotation, into digital form is the obvious method of connecting the shaft through a gear drive to a decade counter. This process is sometimes called shaft rotation digitation, since each rotation of the shaft is explained in digital form. One common application of the gear drive and counter method of conversion is used by the mileage counter (odometer) of an automobile (described in Math vol III NP 10073).

In addition to providing a digital read-out from analog data, the odometer also performs some simple analog computing. The basic information is the rotation of a shaft. A gear train multiplies this by the correct constant to produce miles per revolution. The data are still in analog form. After multiplication, the data are summed by the counter and displayed in digital form to represent the total miles traveled. The original input is analog, the computing is performed by means of analog techniques, but the read-out is digital.

This technique is also used in various other applications. Many airborne and ship-based analog computers used in bombing, navigation, or fire control applications, use this form of read-out both for output indications and for an indication of input information that has been cranked in manually. Although this technique is simple and widely used, it is of rather limited value. It can be used for read-out or read-in purposes only. Since the digital output must be visually observed on the counter wheels, it cannot be used to feed another computer except through a human operator. In fact, this technique is not suitable for automatic operation.

Shaft-Position Digitation By Geared Counter

Shaft position digitation is defined as the process of converting and expressing the instantaneous position of a shaft in digital form. The simplest method of shaft-position digitation is similar to the technique that has been described in connection with shaft-rotation digitation. In the shaft rotation method, where the total revolutions or a function of total revolutions is to be read-out, a decade-type counter is used in order to obtain relatively high readings. For position indication, only a relatively small number of positions are required. It is therefore possible to use only one counter wheel which contains as many digits as may be desired, each digit representing a definite rotary position of the shaft.

If the number of positions is very large, a decade type counter would be helpful. The use of seven counter wheels could theoretically produce extremely fine readings, with a resolution as fine as one second of arc. Two of the wheels, for example the two right-hand wheels on an odometer type counter (not shown), could be used to produce the readings from 00 to 60 seconds; the next two to the left could read from 00 to 60 minutes; and the last three from 000 to 360 degrees.

Unfortunately, the similarities between this technique and the preceding one also reveal the limitations involved, since this method also provides only a read-out or read-in device.

Shaft-Position Digitation by Mechanical Switching

A device that is more useful than the previous devices in at least one respect can be devised by replacing the counter with a mechanical switching arrangement. Either cam-operated sensitive switches or a multi-position rotary switch can be used. With this method, the output is electrical, although the device is basically mechanical. If each switch or tap feeds separate lines, the various output signals will correspond to each separate shaft position. The signals could be used either as an input to some type of computing device, or to feed numbered output lamps.

A disadvantage of this device is the physical limitation associated with the number of positions that can be read out. Each additional position requires the addition of another sensitive switch, or the use of a larger rotary switch, plus another lamp. Although such a device might be practical for applications such as calling out sixteen main points of a compass, any greater number of points would require the use of either an additional coding device or an entirely different technique.

Shaft-Position Digitation by Coding Disks

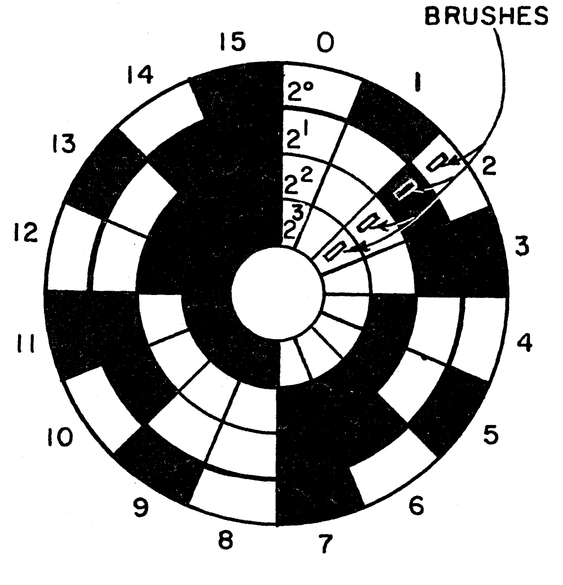
Until now, none of the devices discussed provide any type of numerical code, although most devices designed to receive digital inputs from a converter require the use of some special code. The most frequently used types of codes for these applications generally involve some form of binary coding.

If lamp read-out is desired, the binary system will reduce the number of lamps required. For example, the binary-coded decimal (BCD) system of counting reduces the number of lamps required for reading out the numbers between 0 and 9 from ten lamps to four lamps (since 1 lamp is required in each of 4 columns, to be capable of representing any decimal number from 0 to 9 in the binary system).

When you recall that this ratio increases by powers of 2, the tremendous advantage of the binary system becomes readily apparent. Twenty-two lamps can, in this manner, represent over a million-and-a-half numbers. The lamps can be replaced by tubes, transistors, relays, diode networks, etc., to yield over 1.5 million bits of information in the form of electrical signals.

The coded disk (fig. 9-1) is a device used for adapting a binary code for shaft position digitation. If it is assumed that the shaded areas represent conductors, the light areas represent insulators, and the small rectangles represent brushes, the operation of the device can be described as follows.

The disk is attached to the associated shaft so that as the disk rotates in a counterclockwise direction under the brushes, an ON ("1") signal is generated each time a brush contacts a dark area, and an OFF ("0") signal is generated each time a brush contacts a light (unshaded) area. The circles which produce values of 2^0, 2^1, 2^2, and 2^3 are as shown at the "0" position (segment) of the wheel. Thus, the outermost circle produces the least significant digit (LSD) in the code, followed by increasing orders to the most significant digit (MSD) produced by the innermost circle.



124.99
Figure 9-1.—The binary coded disk.

The binary number represented by each segment of the disk is read by interpreting the shaded and light areas in the segment, containing the brushes, reading from the innermost to the outermost circle. Read an "0" for each light area and a "1" for each dark area. The areas in segment 1 are read 0001, representing the number 1 in the decimal system. The areas in segment 12 are read 1100, (BCD) representing the number 12 in the decimal system. The brushes are shown in a position which reads 0010_2, or 2_{10}.

Because only four zones are used (2^0, 2^1, 2^2 and 2^3), the resolution of the disk is 1 part in 16, (counting zero as 1 part). If a circle which contains 32 alternating ON and OFF areas (each of which is approximately half the size of the present outer circle) were added, a resolution of 1 part in 32 could be obtained. The resolution can thus be increased by a factor of two for each additional circle that is added.

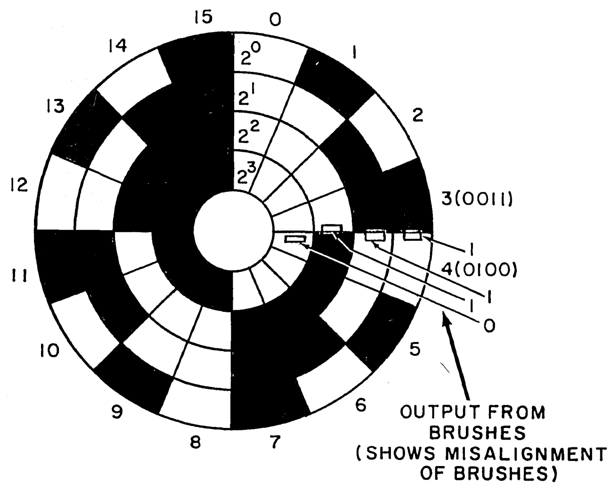
A circuit can be completed through the disk by connecting each conducting segment to a conductor on the rear of the disk, which would serve as a slipring.

Another form of coding disk (not shown) uses the photoelectric principle of reading out digital information. The disk consists of opaque and transparent segments. A light source is placed on one side of the disk, and an arrangement of photocells is placed on the other side. An aperture through which the light must pass in order to reach the disk replaces the brushes as discussed in the previous example of coding disk.

Ambiguity With Natural Binary Code

The major difficulty with the coded disk described above is illustrated by considering the disk to be in a position where the output number is changing from 0011 to 0100 (fig. 9-2).

Because the brushes used for read-out from the disk cannot be perfectly aligned, it is probable that all bits will not change simultaneously. If



124.100
Figure 9-2.—Coded disk showing erroneous output due to misalignment of brushes.

Table 9-1.—Natural Binary Code Showing Sequentially Changing Bits Underlined.

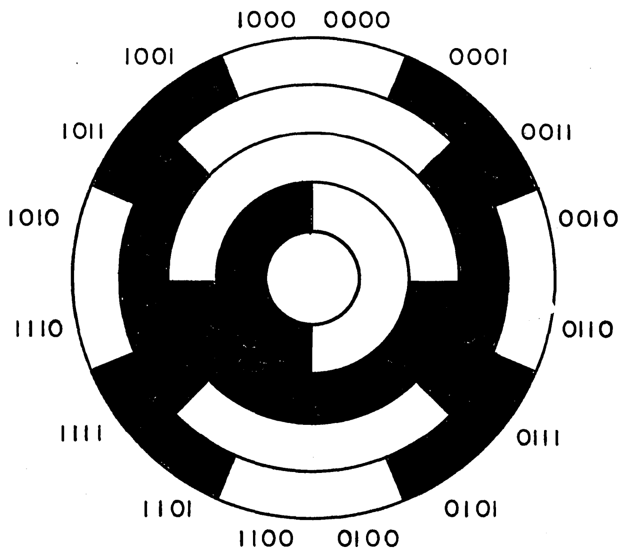
Decimal	Natural Binary Code
0	0000
1	000<u>1</u>
2	00<u>10</u>
3	001<u>1</u>
4	0<u>100</u>
5	010<u>1</u>
6	01<u>10</u>
7	011<u>1</u>
8	<u>1000</u>

the shaft stops at a position where the 2^2 brush is in the conductive (shaded) area and the 2^1 and 2^0 brushes are not completely in the insulated (unshaded) areas, the output will be 0111. This represents an appreciable error. If, on the other hand, the 2^2 brush is not in the shaded area and the 2^1 and 2^0 brushes are in the unshaded areas, the output will read 0000. This again represents a large error.

The possibility of large error is most severe when more than one bit changes value simultaneously. Table 9-1, in which the sequentially changing bits are underlined, reveals how frequently this condition arises. Note that from 0000 to 0011 (from zero to three) only one bit changes in sequence. Starting with the 0000 in the natural binary code or binary coded decimal, alternate numbers require a change in more than one bit. Thus, the possibility or error is great and this code is not satisfactory for use in shaft position digitation.

The most popular code used to circumvent this problem is the "Gray" code (fig. 9-3) sometimes called the "unit distance" code. The Gray Code representation of numbers from zero to fifteen is shown. Note that only one bit changes value from one position to the next higher or lower position. In this way, the magnitude of the error introduced by imperfect alignment of the read-out brushes or by reading out data from a brush (or brushes) which spans a shaded and unshaded section cannot exceed 1. Thus, a 4 may be read for a 5 or a 7 may be read for an 8. Larger errors cannot be made.

Most computers are not designed to perform computations using Gray Code. When this code is used in the analog to digital conversion process,



124.101
Figure 9-3.—Coded disk (Gray code).

it is necessary to convert the data from Gray code to binary coded decimal — a form more suitable for computer use. Decoding matrices of the type discussed in chapter 5 of this text may be used for this purpose.

ELECTRICAL TECHNIQUES

The important differences in the electrical and mechanical techniques for analog-to-digital conversion are now noted. They both convert or "encode" analog inputs into digital outputs. Both types usually make use of a particular zero value setting as a reference point.

The mechanical encoder (explained earlier) generally uses some technique for the ON and OFF switching of a number of electrical contacts on a coded drum (not discussed) or disk, thereby generating coded digital outputs. The digital read-out is sometimes accomplished by causing the mechanical switch to advance a counter.

The analog input to an analog-digital (AD) converter which uses the electrical technique for encoding must first be converted to a voltage. This conversion is achieved by using some type of transducer (a device which converts energy from one form to another, as from mechanical energy to electrical energy). Some common types of transducers are microphones, loudspeakers, phonograph pickup heads, and synchro systems which use a transmitter as the input device and a control transformer as the output device. The type of transducer used for a specific application depends upon the type of analog information to be converted.

In many computer applications it is necessary to transform an analog input voltage into a form more suitable for decoding. A common method of doing this is to express the magnitude of the analog input with respect to time. Once a period to time versus voltage is available to provide a measure of the magnitude of the analog signal, it then becomes a relatively simple process to convert this voltage-with-respect-to-time data into a digital representation. A given time interval can be summed and fed into the computer as a digital input, or displayed on a counter.

Two commonly used methods of performing analog-to-digital conversion by electrical means are explained below. They differ mainly in the method used to determine the magnitude of the analog input. The first method (called the comparison method) used a locally generated sawtooth (ramp) voltage as a reference voltage with which the analog voltage is compared. The second method determines the magnitude of the analog input by means of a coding tube.

Comparison Method

The block diagram of figure 9-4 is used to illustrate the comparison method of determining the magnitude of the analog input.

Generally the arm of the potentiometer is mechanically linked to the device which produces the analog signal, i.e., the float in a tank, the temperature gauge, the pressure gauge, or the servomechanism. Used in this way, the potentiometer and the regulated source voltage serve as a transducer to convert the mechanical input into an electrical output.

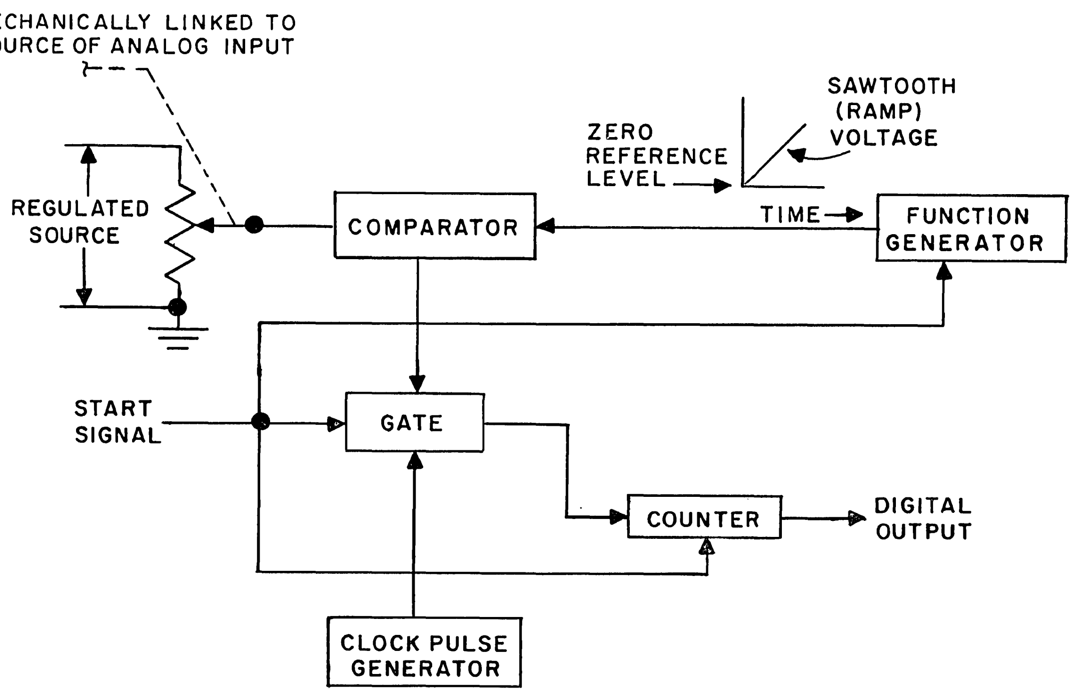
Because each position of the potentiometer arm produces a different value of voltage between its arm and ground, each position is unique in its content.

Thus, each position is considered as a specific address called the "analog address." It should be noted that in theory a total possible number of addresses greater than any preassigned number can be obtained from the arm of the potentiometer. Thus, no attempt is made to assign specific address numbers to the various positions of the potentiometer arm when it is used in this manner.

A specific address number can be assigned to each position of the potentiometer arm if the movement of the arm across the "pot" is made in a specified number of steps rather than by continuous movement. This, of course, is accomplished only by sacrificing accuracy. If specific steps are used the accuracy of the input data increases as the number of evenly spaced taps or steps across the "pot" increases.

Two inputs are applied to the comparator in figure 9-4. These are: (1) the analog input from the potentiometer, and (2) a linear sawtooth voltage from the function generator. The function generator output is initiated each time a start signal is applied. The start signal also enables the gate circuit.

As long as the analog and function generator inputs to the comparator differ in magnitude, the clock pulse generator will be permitted to transmit pulses at a constant repetition rate through the gate into the digital counting circuits in the counter. When the two inputs to the comparator become equal (as a result of the linearly rising sawtooth) the comparator will generate a stop signal which disables the gate circuit and ends the



124.102
Figure 9-4.—Determining the magnitude of an analog input using the comparison method.

comparison time interval. The disabled gate circuit blocks the flow of pulses from the clock pulse generator to the counter.

The number of pulses accumulated in the counter during the comparison time interval is proportional to the amplitude of the analog input voltage. The counter indication is the desired digital representation.

Sawtooth generators of the ''Bootstrap'' type produce output voltages with sufficient linearity to be used as the basic circuit of the function generator. However, linearity is not the only consideration. The slope of the sawtooth voltage must also be controlled since the counter sums the fixed frequency pulses throughout the time interval required for the sawtooth voltage to equal the voltage of the analog sample. Hence, even if the sawtooth were perfectly linear, any variation in its slope would cause the total number of pulses accumulated in a given period to vary accordingly.

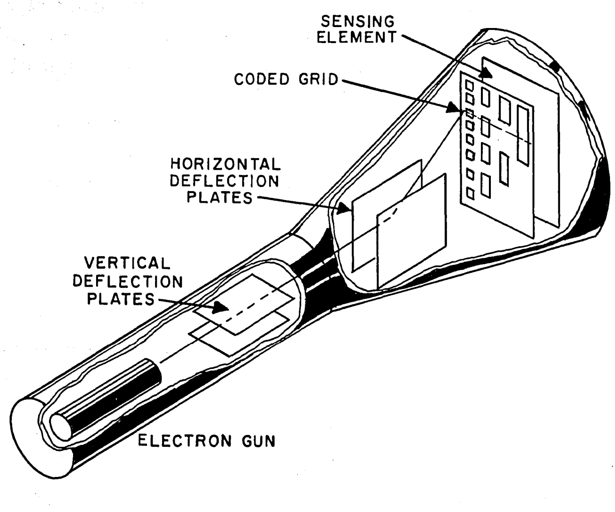
The clock pulse generator is a stable pulse oscillator. It may be a form of stabilized multivibrator. The counter is usually an arrangement of flip-flops of the type discussed in chapter 4.

Coding Tube Method

The coding tube method uses a specially designed cathode-ray tube (fig. 9-5). The special features of the tube are in the form of a grid, which contains rows of coded slots, and a special sensing element.

The coding tube method of performing analog-to-digital conversion consists of the following operations. The voltage that is to be converted is connected (after amplification) to the vertical deflection plates. A linear sawtooth sweep voltage is connected to the horizontal deflection plates. The connections to the vertical and horizontal deflection plates are not shown. The stream of electrons, which is emitted, focused, and accelerated by the electron gun, is positioned vertically on the grid so that it strikes a particular line of slots that correspond to the amplitude of the voltage to be converted.

The linear sawtooth voltage on the horizontal deflection plates causes the beam to be scanned across the line of slots. This action causes a coded image to be developed on the sensing element, which serves as the digital read-out device.



20.320
Figure 9-5.—The cathode-ray coding tube.

High speed is the main advantage of the coding tube method of conversion. The disadvantages are that its accuracy is limited by the linearity of the scanning beam, and by the beam diameter or spot size. The greater the spot size the less reliable is the result.

DIGITAL-TO-ANALOG CONVERSION

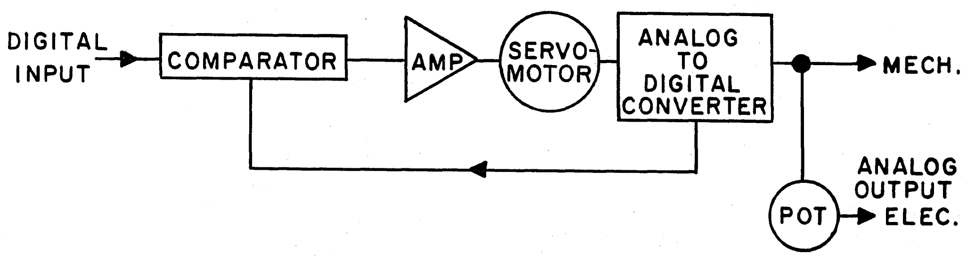
Digital-to-analog conversion, like analog-to-digital conversion, can be accomplished by both mechanical and electrical methods. The purpose of mechanical digital-to-analog conversions is to convert digital information into a mechanical movement or shaft position. Some of the basic principles involved in both methods are explained in the following paragraphs.

MECHANICAL TECHNIQUES

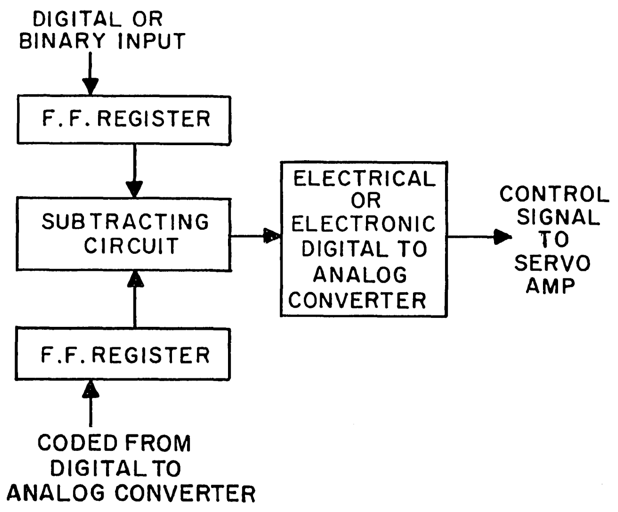
Mechanical digital-to-analog conversion is almost always based upon feedback and comparison principles. A representative example of this action is illustrated in figure 9-6. The digital input is fed into a comparator which produces an input to a servoamplifier and servomotor combination. The servomotor is used to drive any of the many types of mechanical analog-to-digital converters. The digital output of the converter is fed back to the comparator, which operates as the error-detection circuit. As long as the digital output does not equal the input, an error signal will persist. This signal, in turn, causes the servomotor to continue to rotate until the error is reduced to zero.

The shaft or other mechanically driven device at the output of the analog-to-digital converter moves through a certain angle which is the analog representation of the digital input. If a transducer arrangement which uses a potentiometer for read-out purposes is connected to the output shaft, an electrical signal will be produced which represents the same analog quantity.

One form of the comparator which will perform the desired error detection function is illustrated in figure 9-7. Using this arrangement, the difference between the digital or binary input and the present actual instantaneous position of the servo shaft is constantly being converted into an analog signal which is applied to the servoamplifier. The flip-flop (FF) registers store their respective inputs. The subtracting circuit computes the difference and interprets this difference as the existing error. When the analog shaft position corresponds to the digital input (as indicated by a zero difference in the subtracting circuit), the signal to the servoamplifier will be zero and no further shaft rotation will occur.



124.104
Figure 9-6.—Block diagram of a mechanical digital-to-analog (D-A) converter.


DIGITAL OR BINARY INPUT
F.F. REGISTER
SUBTRACTING CIRCUIT
ELECTRICAL OR ELECTRONIC DIGITAL TO ANALOG CONVERTER
CONTROL SIGNAL TO SERVO AMP
F.F. REGISTER
CODED FROM DIGITAL TO ANALOG CONVERTER

124.105
Figure 9-7.—Representative comparator, block diagram.

ELECTRICAL TECHNIQUES

Many electrical digital-to-analog converters operate on the principle of voltage division using selective relays and a series-parallel network of precision resistors. One of the simplest circuit arrangements based on this principle is shown in figure 9-8.

The binary (digital) input is fed into the bank of flip-flops. The 1 bits in the word stored in the flip-flops cause their respective relays to energize, which results in the closing of the associated contacts. Each of the resistors R_1 through R_4 are precision resistors, i.e., they have very low tolerances. Assuming that each of the parallel branches conducts a given current value when its associated circuit switch is closed, it follows that the value of current through R_L is a function of the number of branches which are placed in the conducting state. If high accuracy is to be maintained, R_L must be kept small in comparison with resistors R_1, R_2, R_3, and R_4, even when these resistors are all connected in parallel. Because R_L is necessarily of low value, the output voltage (C_o across R_L) is low compared to the value of the input voltage (usually of the order of one volt). When higher output voltage levels or higher power output are required, the output voltage, e_o, can be amplified. (See Basic Electronics, NP 10087—revised.)

A similar method of performing digital-to-analog conversion involves the use of a series arrangement of resistors connected in a feedback loop of an operational amplifier circuit. In order to understand this method it is first necessary to discuss the basic principles of the operational amplifier, and also the principle of negative feedback amplifiers as they are used in operational amplifiers.

NEGATIVE FEEDBACK AMPLIFIER

A schematic diagram of an amplifier which uses an electron tube and degenerative (negative) feedback is shown in figure 9-9. The characteristics of this amplifier are largely determined by the amount and type of feedback developed across R_f as will be shown presently.

In conventional amplifiers, any attempt to calculate the amount of amplifier voltage gain along a given parameter, is complicated by changes in input and output impedances, polarity and amount of feedback, increases or decreases in frequency, and several other factors. It is shown in the following paragraphs that the voltage gain in a negative feedback amplifier is determined by the ratio of the feedback resistance (R_f) to the grid resistance (R_g, or R_{in}).

In the schematic diagram of figure 9-9, it is known the e_{in} represents the input signal voltage which, in this case, is developed across R_{in} between grid and cathode of the tube. Simultaneously, an amplified voltage, which appears between grid and plate of the tube, is developed from plate to ground (across R_L and the B supply in one path and across R_f and R_{in} in the other). Because of the inherent 180° triode phase shift between grid and plate, the feedback voltage developed across R_{in} is 180° out of phase with the input signal voltage e_{in}. The opposing voltages across this resistor produce the degenerative feedback.

That the gain A of the amplifier is equal to R_f/R_{in} can be shown as follows:

(1) $iR_{in} + iR_f = Ae_{in}$

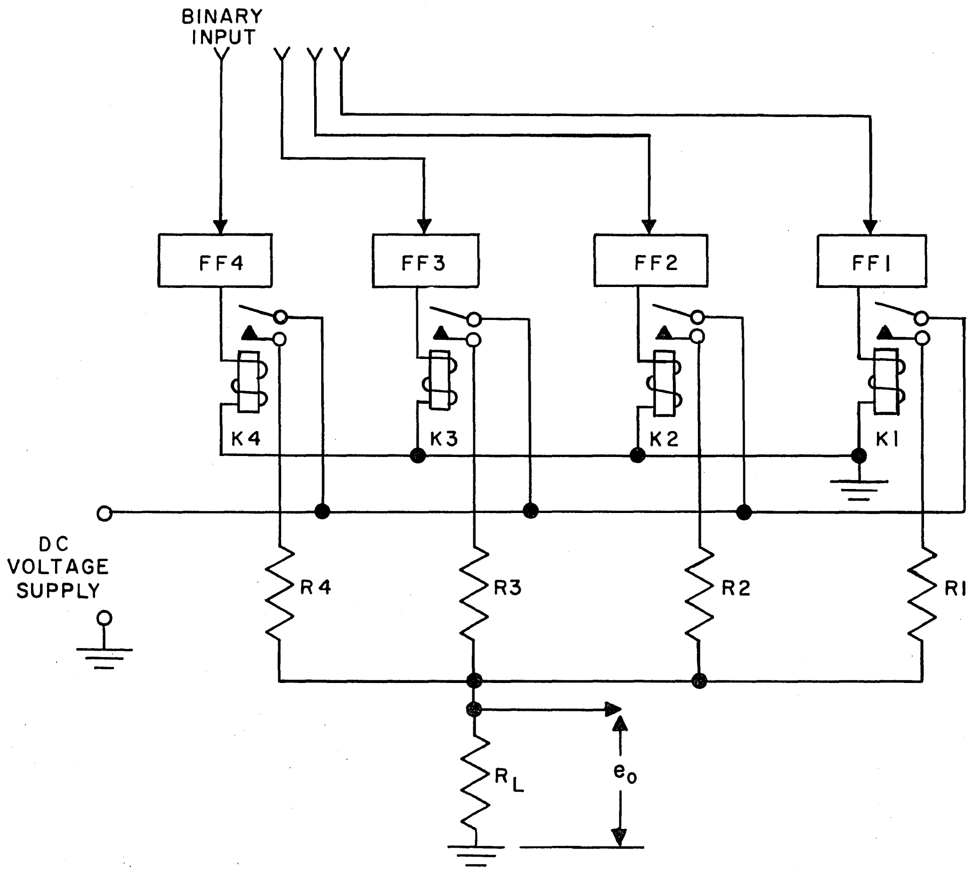
(2) $\cancel{i}R_{in} + \cancel{i}R_f = A\,\cancel{i}R_{in}$

(3) $\dfrac{R_{in}}{R_{in}} + \dfrac{R_f}{+R_{in}} = A\dfrac{R_{in}}{R_{in}}$

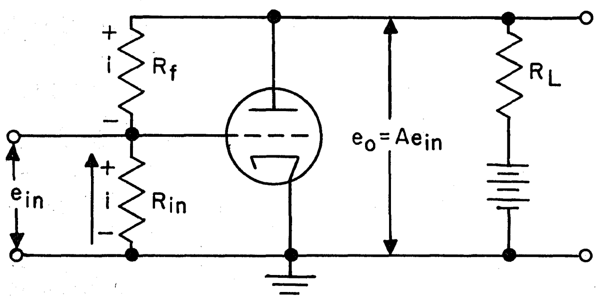
(4) $1 + \dfrac{R_f}{R_{in}} = A$

(5) $\dfrac{R_f}{R_{in}} = A$

where A = stages gain
and $e_g = iR_{in}$



124.106
Figure 9-8.—Digital-to-analog conversion using selective relays and voltage dividers.



124.107
Figure 9-9.—Schematic diagram of a negative feedback amplifier using an electron tube.

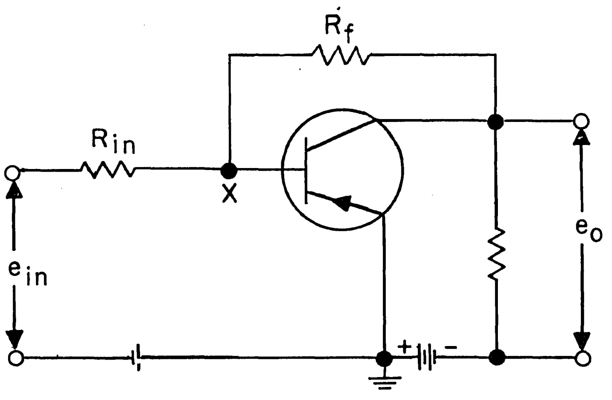
Because the Ratio of R_f to R_{in} is usually high, the 1 in equation (4) is negligible.

The stage gain of the transistorized degenerative amplifier in figure 9-10 is calculated in the same way. The series connection of R_{in} with respect to the emitter-base junction and the input signal source permits the emitter-base current to vary only slightly when relatively large signal voltage changes occur at the input. Likewise, the effect of feedback current changes through R_{in} (which are very large compared to the forward emitter-base resistance) produce a large controlling effect on the amplifier output.

Also note that the large ohmic value of R_{in} with respect to the emitter base junction permits point X to maintain only a slight voltage value (negative) to ground.

Feedback and emitter base currents flowing at point X are relatively high. These factors make point X a point of low impedance, i.e.,

$$Z_x = \frac{e_x \text{ (low)}}{i_x \text{ (high)}}$$



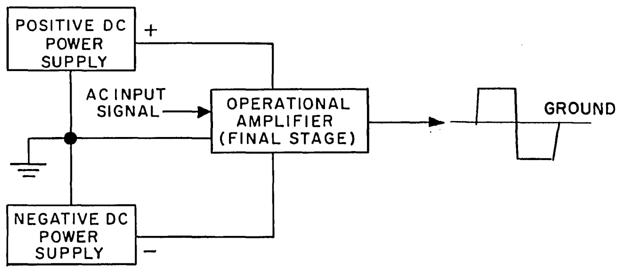
124.108
Figure 9-10.—Schematic diagram of a negative feedback amplifier using a PNP transistor.

Because of the large value of Rin with respect to the emitter-base resistance, the input current does not change appreciably over wide variations of the input signal amplitude. Thus the input source (as seen from point X) is, in effect, a constant current source. The input impedance is essentially the value of R_{in} and point X is very near ground potential. The circuit will not produce a useful output if point X is maintained at zero volts, since this would mean that the amplifier input is zero. Rather, point X must be maintained at a slight voltage value to ground to permit accurate amplification. The type amplifier just described is frequently used in computers to multiply the d-c input by a constant (A), the stage gain.

OPERATIONAL AMPLIFIERS

An operational amplifier is a very high gain direct-coupled feedback amplifier. It may consist of several stages, which, combined, produce output signals several million times the amplitude of the input.

An operational amplifier circuit is arranged so that the input signal is applied with respect to ground and the output signal is taken from a point on the final amplifier which is at zero volts d-c with respect to ground. The zero d-c level at the output is established by connecting the final amplifier stage across two power sources (fig. 9-11) whose outputs are of opposite polarity to ground. The output terminal is connected to that point on the amplifier which is at the zero volt d-c ground potential. It follows that a-c output signals alternate above and below the ground reference level.

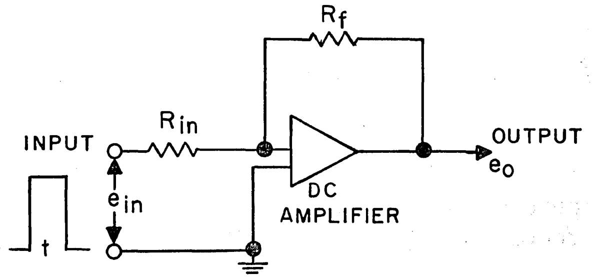


124.109
Figure 9-11.—Basic block diagram of operational amplifier showing method used to obtain d-c ground at output.

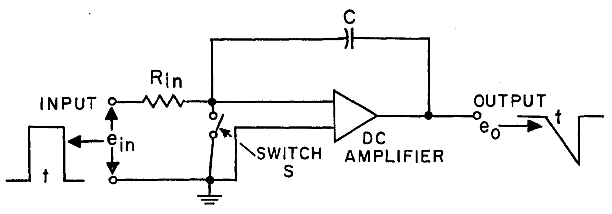
The operation of an operational amplifier circuit (fig. 9-12) is as follows. Assume that an input signal, e_{in} (which tends to increase the d-c amplifier output current), is applied across the input terminals. The current output from the circuit flows through R_f and the output impedance in parallel. Because R_f is (in general usage) several hundred times the size of R_{in}, and because the current through R_{in} is smaller than the current through R_f, it follows that the voltage across R_f will be many times the voltage across R_{in}. This ratio is a measure of the amplification.

Since the voltage across R_{in} is the input voltage (minus the negligible drop across the emitter-base junction), the ratio of R_f/R_{in} represents the stage gain (A).

An operational amplifier can be used as a Miller sweep generator, or an integrator by replacing the feedback resistor R_f with a capacitor, C, as shown in figure 9-13. Switch, S, is added so that the integrating process can



124.110
Figure 9-12.—Operational amplifier using resistive feedback circuit.



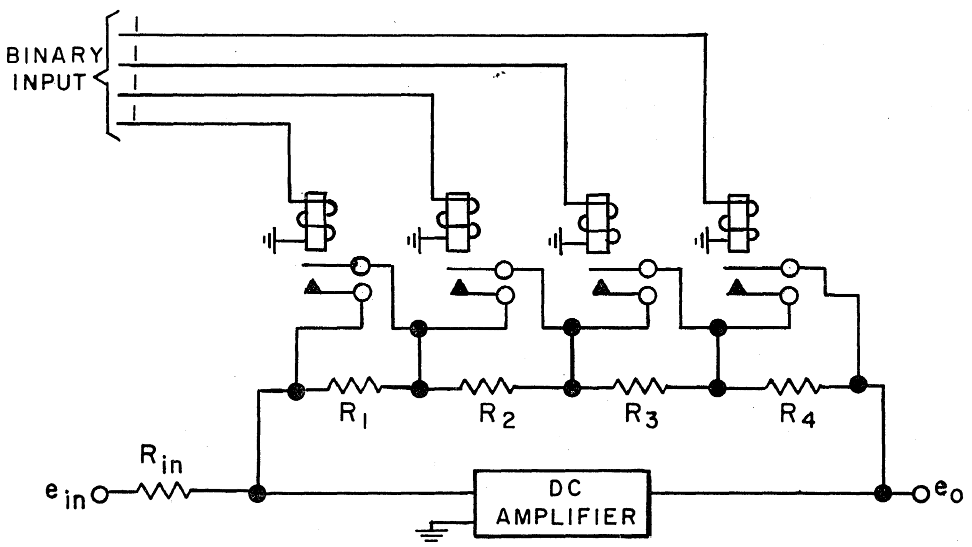
124.111
Figure 9-13.—Operational amplifier using capacitive feedback circuit.

be started and stopped at a desired rate. In practice, switch, S, is replaced by an electronic switching device such as an electron tube or a transistor. The voltage amplifier is a high-gain high-impedance circuit.

The degenerative feedback current through capacitor, C, prevents the amplifier input voltage from changing instantaneously with e_{in}. The combined effects of the input voltage and the negative feedback maintain a constant charging current through C, which, in turn, produces a linear voltage change across this capacitor. The output voltage is proportioned to the integrated input voltage. The slope of the output waveform is dependent upon the times constant R_{in} C_{FB}. Where R_{in} = 1 megohm and C_{FB} = 1 microfarad the time constant is 1 and the scope is linear.

DIGITAL-TO-ANALOG CONVERSION USING THE SERIES RESISTOR METHOD

The circuit in figure 9-14 contains a series array of resistors (R_1 through R_4, of selected sizes) in the feedback loop of an operational amplifier. Each resistor is bypassed by a pair of relay contacts which represent zero (on OFF) in the closed position and one, (or ON) in the open position. When all of the contacts are closed there is zero resistance in the feedback path, and the amplifier will not produce a voltage gain. When a 1 binary input is applied across any or all of the relay solenoids, the associated contacts open (as shown) causing resistance to be added to the feedback loop. The amount of increase in the amplifier gain is in direct proportion to the amount of resistance placed in the feedback loop by the opening of the relay contacts. The voltage, e_o, therefore represents the analog value of a digital input.



124.112
Figure 9-14.—Digital-to-analog conversion using operational amplifier and a series array of feedback resistor.

THE DIGILOG CIRCUIT

A digilog circuit (sometimes called a decoder) is used to convert digital signals to analog voltages. In effect, it is a translator. It translates the language of digital devices, ("1's" and "0's") into the language of analog devices. These analog voltages are then used to position needles on meters, oscilloscopic sweeps, and/or dots on cathode ray tubes, turn motor shafts, drive analog circuits, etc.

The digilog is composed of a bank of transistor switches and a resistive network known as a "ladder-adder". Figure 9-15 shows a block diagram of one transistor switch and the ladder-adder network. In an actual circuit where several digital inputs are received, a transistor switch is needed for each binary input digit. The transistor switches are opened and closed by the digital signal, allowing either a voltage or ground to be presented to the ladder-adder resistive network. The output of the ladder-adder is the required analog voltage.

DIGILOG TRANSISTOR SWITCH

The purpose of the transistor switches is to supply a voltage to the ladder-adder as determined by the binary value of the digit which controls each switch. There are, of course, many transistor circuit configurations which may be used to accomplish the switching. Generally, the digilog switching circuit will be similar to that shown in fig. 9-16. The circuit of fig. 9-16 uses a positive reference voltage V_{Ref}, as indicated at the emitter of Q1. By replacing Q1 with an NPN transistor and Q2 with a PNP transistor and properly biasing the transistors, a negative reference voltage can be switched into the ladder-adder.

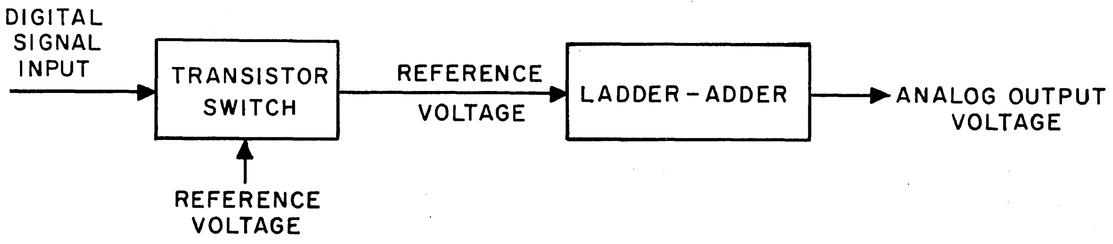
As stated earlier, a transistor switch is needed for each binary input digit. For example: A four digit word requires four transistor switches and a ten digit word requires ten transistor switches. In fig. 9-16 it is assumed that the true or "1" state is represented at the input and that the false or "0" input state is represented as 0V. With a true (-V) at the digital input terminal, Q1 conducts heavily and Q2 is cut-off. Hence the + V_{Ref} is supplied to the ladder-adder via the low conducting resistance of Q1 in series with R6. When the input is false (0V), Q1 is cutoff and Q2 conducts, applying ground to the ladder-adder via the low conducting resistance of Q2 in series with R7.

THE LADDER-ADDER RESISTIVE NETWORK

For purposes of explanation, four transistor switches and corresponding ladder-adder network is illustrated in figure 9-17.

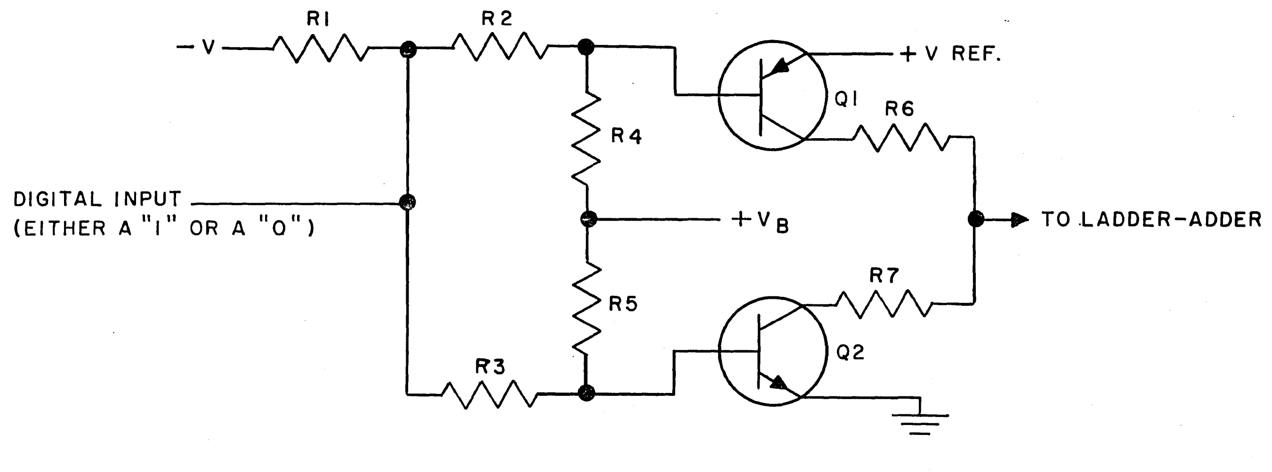
It is important to remember that in a practical application, the digilog can have as many transistor switches as desired and the associated ladder-adder network. A four switch digilog has sixteen possible voltage levels as an output. This is so because the input can be either true or false (-V or 0) and there are four input switches. Hence there are 2^4, or sixteen, possible input and output combinations. A ten switch digilog has 2^{10} or 1024 possible output voltage levels. The greater the number of switches in the digilog, the smaller the incremental difference in the output voltage levels as will be shown later. For simplification, the transistor switches have been replaced by toggle switches.

Certain basic requirements must be met with regard to relative resistance values and supply voltages in order to produce satisfactory results from the ladder-adder network. These



124.113

Figure 9-15.—Block diagram of digilog action.



124.114

Figure 9-16.—Digilog transistor switch.

are as follows: Resistors R3 = R5 = R7 and R1 = R2 = R4 = R6 = 2R3. A + V_{Ref} and $-V_{Ref}$ of equal absolute value must be used. To aid in understanding the digilog, the following values are assigned:

A. R3, R5, R7 = 5K ohms
B. R1, R2, R4, R6 = 10 k ohms
C. $+V_{REF} = +10$ volts
D. $-V_{REF} = -10$ volts

Table 9-2 shows the possible input conditions to the digilog switches, the corresponding position of each switch (either ground or a reference voltage), and the analog output voltage. A positive reference voltage is utilized at switches A, B, and C, and a negative reference voltage for switch D, so that the output voltage, (V_0) can swing from a maximum positive voltage value through zero volts to a maximum negative voltage. In practice though, the digilog can be constructed to vary between any two voltage levels desired.

A few of the output voltages produced at the ladder-adder output of figure 9-17 will be calculated using figure 9-18A, B, C, D, and E. When points A, B, and C are grounded and D is connected to the - 10 volts reference, the circuit equivalent is as illustrated in figure 9-18A. Note that resistors R1 and R2, are in parallel to ground. Hence, R1 and R2 combine into a single 5K resistance in series with R3 (5K) to ground. The resistance from point M to ground is now 10 K as illustrated in the equivalent circuit of figure 9-18B.

R_{eq1} $(R3 + \frac{R1 \times R2}{R1 + R2})$ is in parallel with R4 to ground and the combination is in series with R5. Disregarding R_6, these three resistors (R4, R5, and R_{eq1}) combine to form a new equivalent resistance to ground from point N (fig. 9-18B) which is 10K. This result in R_{eq2} $(R5+\frac{R4 \times R_{eq1}}{R4 + R_{eq1}})$ as shown in figure 9-18C.

Resistor R6 and $R_{eq}2$ constitute a parallel combination which results in R_{eq3} (fig. 9-18D) from point N to ground of 5K. This represents the final equivalent of the circuit in figure 9-17 with points A, B, and C grounded and point D connected to a -10v supply. In figure 9-18D, V_0 is -5V which is the maximum negative voltage obtainable from the digilog in figure 9-17.

Now consider another condition which can exist in the circuit of figure 9-17. Leaving point D connected to the -10V reference and points A and B connected to ground, the V_0 will be calculated when point C is connected to the + 10 V reference.

From the previous discussion the equivalent circuit will be as shown in figure 9-18E (batteries added for continuity). The output voltage is calculated using simultaneous equations. The current through R6 is arbitrarily referred to as i_1, the current through R_{eq3} is called i_2, and the current through R7 is $i_1 + i_2$ by Kirchoff's law; the sum of the voltage drops around a closed circuit is equal to the source voltage.

Table 9-2.—Input and Output Levels for Four Digit Digilog.

Possible Switch Controlling Input Conditions				Switch Position Due To Input Conditions				Analog Output Voltage
Q4	Q3	Q2	Q1	D	C	B	A	V_0 (volts)
0	0	0	0	gnd	gnd	gnd	gnd	0
0	0	0	1	gnd	gnd	gnd	+ 10v	+ 0.625
0	0	1	0	gnd	gnd	+ 10v	gnd	+ 1.250
0	0	1	1	gnd	gnd	+ 10v	+ 10v	+ 1.875
0	1	0	0	gnd	+ 10v	gnd	gnd	+ 2.500
0	1	0	1	gnd	+ 10v	gnd	+ 10v	+ 3.125
0	1	1	0	gnd	+ 10v	+ 10v	gnd	+ 3.750
0	1	1	1	gnd	+ 10v	+ 10v	+ 10v	+ 4.375
1	0	0	0	-10v	gnd	gnd	gnd	- 5.000
1	0	0	1	-10v	gnd	gnd	+ 10v	- 4.375
1	0	1	0	-10v	gnd	+ 10v	gnd	- 3.750
1	0	1	1	-10v	gnd	+ 10v	+ 10v	- 3.125
1	1	0	0	-10v	+ 10v	gnd	gnd	- 2.500
1	1	0	1	-10v	+ 10v	gnd	+ 10v	- 1.875
1	1	1	0	-10v	+ 10v	+ 10v	gnd	- 1.250
1	1	1	1	-10v	+ 10v	+ 10v	+ 10v	- 0.625

(The source voltage applied in a circuit is sometimes the combined voltage of two or more supplies.) Thus (1) $E_{R7} + E_{R6} = E_1 + E_2$ (2) $E_{R7} + E_{Req3} = E_2$ and (3) $E_{R6} + E_{Req3} = E_1$.

Substituting IXR in equation (1)

$$5K (i_1 + i_2) + 10K (i1) = 20$$

Substituting IXR in equation (2)

$$5K (i_1 + i_2) + 10K (i_2) = 10$$

Substituting IXR in equation (3)

$10K (i_1) - 10K(i_2) = 10$ (Note that the two voltage supplies tend to produce currents through R_{eq3} in opposite directions. Thus the voltage developed across R_{eq3} is the resultant voltage established by the opposing supplies.)

Simplifying the equations

$$5K (i_1 + i_2) + 10K (i_1) = 20$$

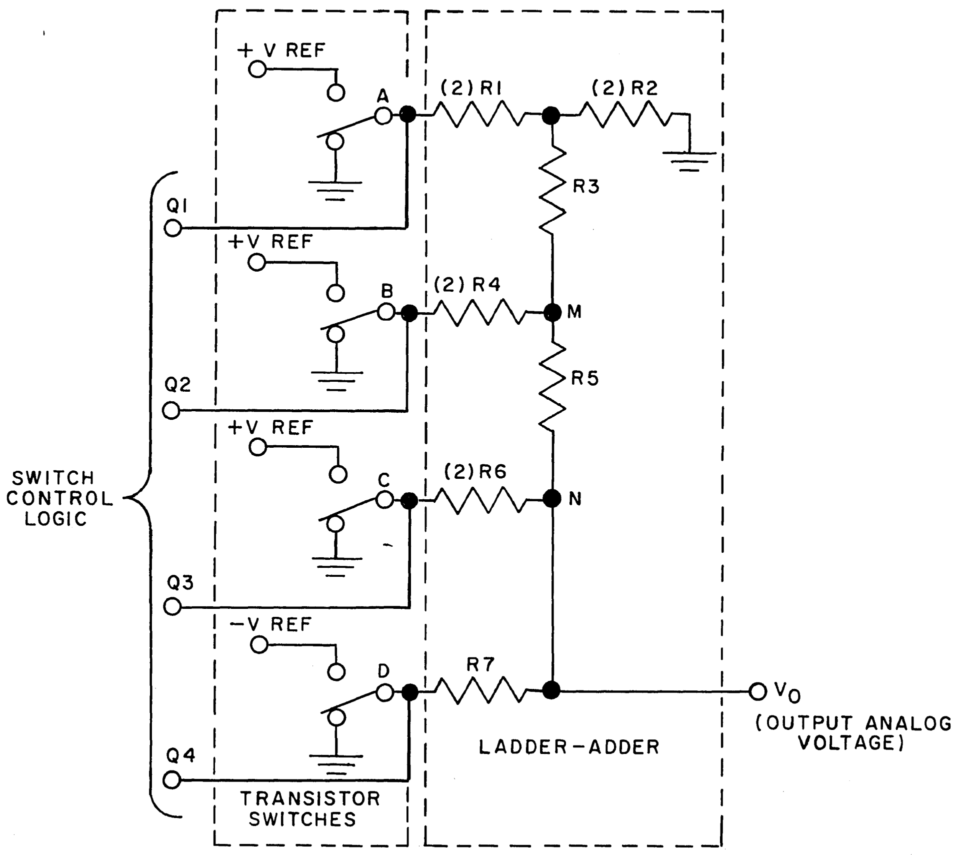
$$5K (i_1) + 5K (i_2) + 10K (i_1) = 20$$

(a) $$15K (i_1) + 5K (i_2) = 20$$

$$5K (i_1 + i_2) + 10K (i_2) = 10$$

$$5K (i_1) + 5K (i_2) + 10K (i_2) = 10$$

(b) $$5K (i_1) + 15K (i_2) = 10$$



124.115

Figure 9-17.—A four digit digilog.

From equations (a) and (b); where equation (b) is multiplied by 3,

$$\cancel{15K(i_1)} + 5K(i_2) = 20$$
$$\cancel{15K(i_1)} + 45K(i_2) = 30$$
$$-40K(i_2) = -10$$

$$i_2 = \frac{-10}{-40K}$$

$$i_2 = .00025 \text{ ampere}$$

Solving for current (i_1)

$$15K(i_1) + 5K(1/4ma) = 20v$$
$$15K(i_1) = 20v - 1.25v$$
$$i_1 = \frac{18.75v}{15K}$$
$$i_1 = .00125 \text{ ampere}$$

Using Ohms law and substituting in equation (2)

$$E_{R7} + E_{Req3} = E_2$$

$$5K(i_1 + i_2) + 10K(i_2) = 10v$$

$$5\times10^3(.00125+.00025)+10\times10^3(.00025)=10v$$

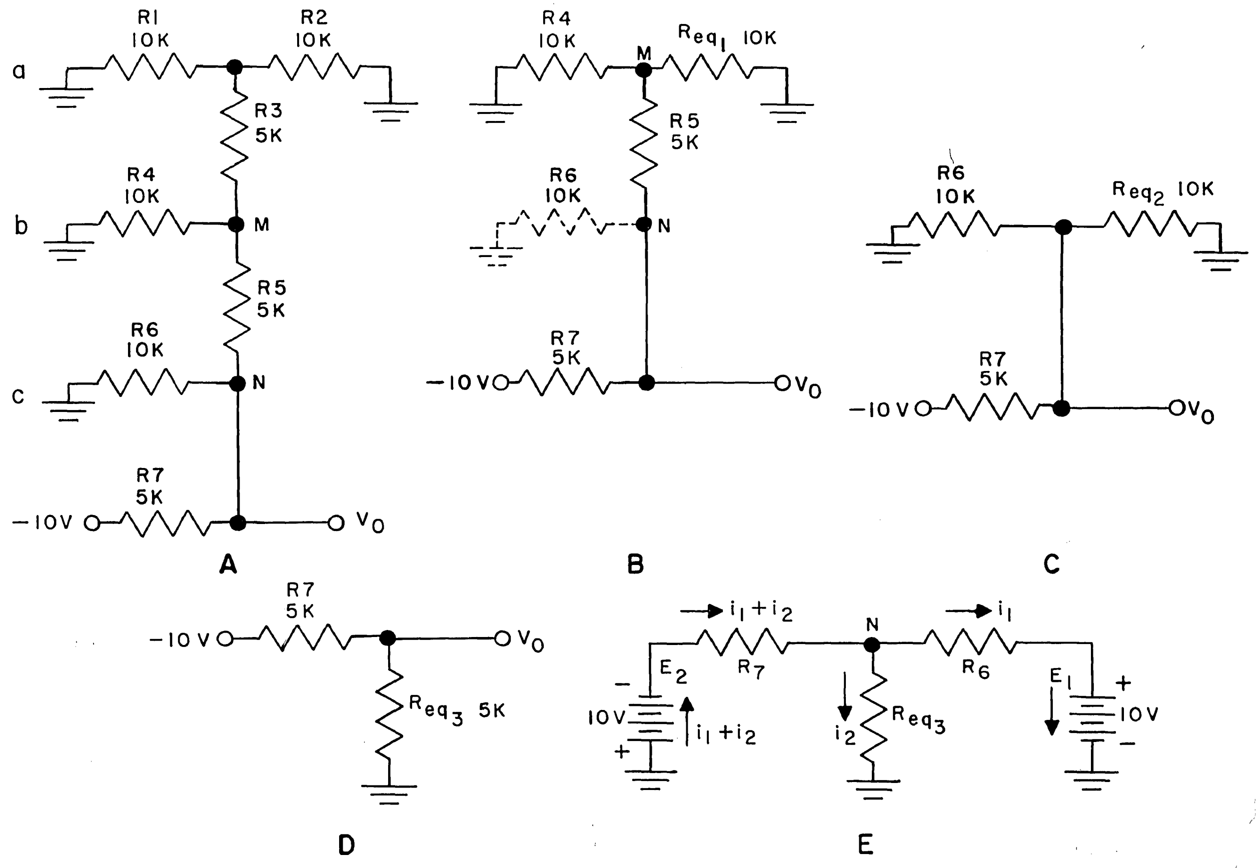
$$5\times10^3(.00150) + 10\times10^3(.00025)=10v$$

$$5\times10^3(150\times10^{-5})+10\times10^3(25\times10^{-5})=10v$$

$$750v \times 10^{-2} + 250v \times 10^{-2} = 10v$$

$$7.5v + 2.5v = 10v$$

$$7.5v - 10v = -2.5v$$



124.116

Figure 9-18.—Analysis of ladder-adder circuit.

Thus, $E_{Req3'} = 2.5v$ when point D (fig. 9-17) is connected to the -10v reference, points A and B are grounded, and point C is connected to the +10v reference.

Leaving point D (fig. 9-17) connected to the -10V reference, C to the +10V reference, and A to ground, the V_0 will be -1.25V when point B is connected to the +10V reference (calculation not shown). With all points (A, B, C, and D) connected to reference voltages, the V_0 is -.625 V, which is the minimum negative voltage output available. If all inputs to the ladder-adder are at ground, V_0 is zero volts. Minimum positive voltage output is obtained when point A is connected to the +10 volt reference and points B, C, and D are grounded. Maximum positive output voltage is obtained when points A, B, and C are connected to the +10V reference, and point D is grounded.

CHAPTER 10

NTDS COMPUTER, CONTROL SECTION

The treatment of the NTDS computer (CP 642A/USQ-20v) as presented in this and other chapters in this text is intended to disclose the functional operation of the various computer sections and circuits. The coverage is not all-inclusive and the reader must therefore refer to the instruction manual for more detailed information.

Chapter 10 describes the computer control section. The explanations include the purpose and the operations of the various registers, translators, and other special circuits. A description of data flow, information regarding the interdependence of sections, and an explanation of the sequence of operations are also included.

DESCRIPTION

The CP-642A/USQ-20 (v) computer is made up of four sections: control, arithmetic, input/output, and memory. (The input/output section is sometimes taken as two separate sections.) These sections contain registers, modifying circuits, designator circuits, timing sequences, and other control operations as necessary to successfully execute instructions. A functional block diagram of the computer is shown in figure 8-1 (chapter 8).

REGISTERS

The following registers (fig. 10-1) are located in the control section.

U register. .a 30-bit program-control register, which contains a 15-bit upper section (U_u) and a 15-bit lower section (U_L). The register is shown in two sections in the diagram for reasons described later.

P register. . .a 15-bit program-address register

K register. .a shift, multiply, and divide control register. The K register is subdivided into K0, K1, and K2 registers for simplification of the K register operations.

R register. .a 15-bit communications control register

R* register. .A 15-bit indexing and incrementing register

B registers. .(1 through 7) 15-bit index registers

DESIGNATOR CIRCUITS

Four designator circuits (not shown) are located in the control section. These circuits are used to interpret and control the execution of certain instructions.

Recall that the U register (instruction register) holds the instruction word during its execution. The designators interpreted in the designator circuits are in addition to the instruction word designations f, j, k, b, and y (discussed in chapter 8) which are stored in the U register. They are:

The g-designator circuit. .a 2-bit branch designator circuit

The p-designator circuit. .a 2-bit program address counter control designator circuit

The h-designator circuit. .a-6-bit branch evaluation designator circuit

MODIFYING CIRCUITS

The following modifying circuits (not shown) are located in the control section:

The $U_L + \dot{R}^*$ adder

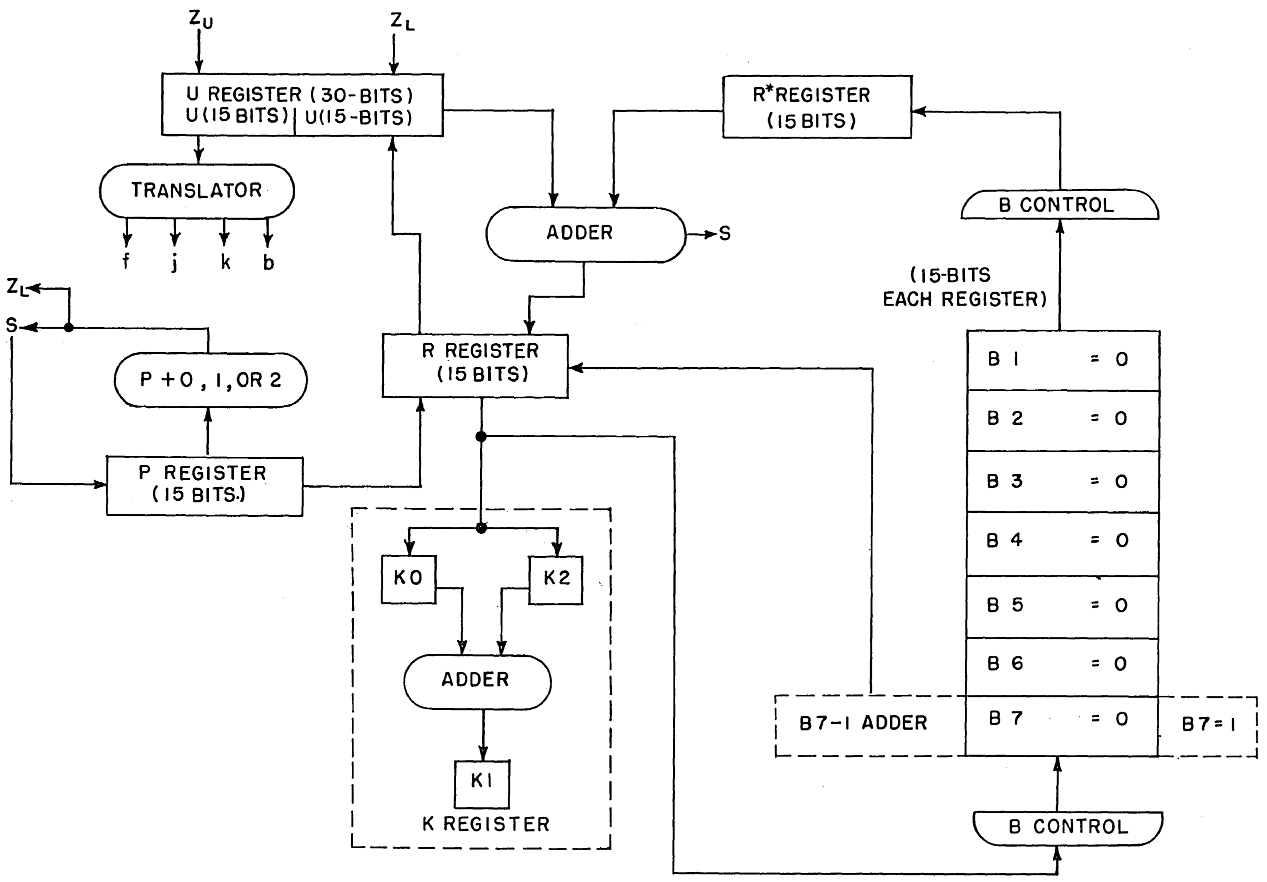
The P adder, adds (P) + p designator (P + 0, 1, 2 etc.)

The K adder (K0 + K2)

The $B_j = 0$ circuits

The $B_7 = 1$ circuit

The B_7 -1 adder



124.117

Figure 10-1.—Control section, block diagram.

MEMORY SECTION

The following information regarding the memory section of the computer (treated in a later chapter of this training course) is of particular importance to an understanding of the control section.

The computer memory section is a large-capacity, magnetic-core storage system. It is a high-speed, random access, nonvolatile storage system, i.e., its speed of operation is compatible with the computer; its data words may be referenced in a non-sequential manner; and it retains data words through normal on-and-off conditions of power.

The memory section contains 32, 768 locations (memory registers) for storage of 30-bit words. Each 30-bit word can be divided into two 15-bit words, the upper 15 bits and the lower 15 bits. By proper programming (using the k-designator) each of these 15-bit words may be handled separately.

Each of the 32-768 memory registers is a unique address. The K-designator (discussed in chapter 8) is used to define the word length, i.e., k = 2 when M_U (the upper 15 bits of the memory Address) is to be used as a 15-bit word, and K = 3 when the 30 bits of the memory address are to be used as one 30-bit word.

The total time needed for the memory reference, or the basic memory cycle time, is eight microseconds. The readout time, or the time from the moment when a given function assumes control of the memory sequence to the delivery of the data from the memory, is approximately 2.0 microseconds.

There are two control registers, the S-register (see figure 8-1) and the Z-register, associated with the memory. During a memory reference, the S-register (address selection register) contains the 15-bit address word that specifies one of the 32,768 memory registers. Data transmission into, or out of, the selected memory register is channeled through the Z-register. The arithmetic, control, and input/output sections of the computer have independent access to the memory registers by using the S and Z registers.

BASIC OPERATION

Data are transmitted from external equipment into the computer and from the computer to the external equipment through the input/output section. All data entered into the computer are loaded into memory, and data transmitted from the computer through the input/output section must come from memory. Data are transmitted from memory to the appropriate output channel or to the arithmetic section according to the requirements of the computer program and accessibility of output channels or the arithmetic section.

All basic processes are performed by the arithmetic section. The control section regulates transmission into and out of memory, and regulates input and output according to the program of instructions. The control section also performs other special functions.

Because of the interdependence of one section upon the other, no one section of the computer can be explained completely without introducing some material which logically belongs in the discussion of the other sections. In the following discussion of the control section, frequent reference is made to parts contained in, or functions performed in other computer sections. Complete explanation of the parts or functions are, in general, not contained at that point, but are left to the discussion of the particular section.

CIRCUIT DESCRIPTION AND SYMBOLS

The basic circuits and symbols used in the CP-642A/USQ-20 (v) are discussed in the following paragraphs.

FLIP-FLOPS (SYMBOLS AND NOMENCLATURE)

The majority of flip-flops in computers are (1) in timing chains, and (2) as components in storage registers. Timing chains are used to accurately time pulses which are required by the various gate circuits in the computer in order to execute particular steps involved in performing a given operation in a specific time period. Timing chains are generally constructed of flip-flops and/or delay lines in an arrangement which produces the timed pulses.

A basic example of a computer timing chain is shown in figure 4-12 (chapter 4). The timing chain consists of the crystal oscillator, the multivibrator (flip-flop), and the delay line components D0 thru D5.

In the following discussion, flip-flops are identified (fig. 10-2A) by the bold line with arrows between the two major elements indicated by the two circles. The one side is distinguished from the zero side by noting the 00 left superscript of T (indicating the zero side) and the 01 left superscript of T indicating the one side. The zero side of the flip-flops in timing chains is always the top circuit, i.e., the ${}^{00}T^{05}$ element is always above the ${}^{01}T^{05}$ element. Flip-flops in timing chains are always drawn in the vertical position.

The other information within each circle indicates the physical location of the card containing that element (see figure 10-2B). This information is not important in this discussion and is deleted from subsequent diagrams.

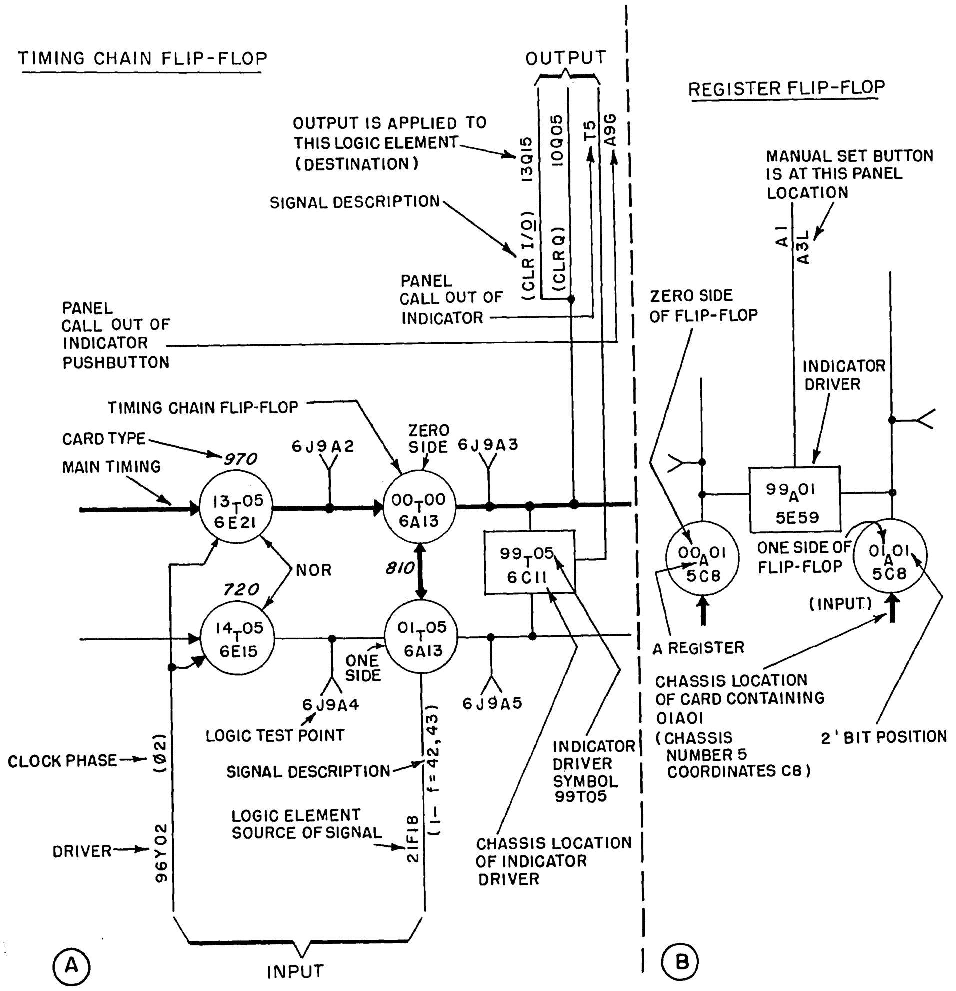
The flip-flops of a register (fig. 10-2B) are drawn in the horizontal plane as opposed to the vertical flip-flops of timing chains.

An indicator driver (by which the flip-flops can be manually set as described later) is centered between the two sides of the flip-flop.

The letters and numbers by which the flip-flop is identified convey a special meaning.

For example in figure 10-2B the "A" indicates the "A" register. The left side is the zero side as shown by the left superscript "00" in the designation ${}^{00}A^{01}$ inside the left circle. The right hand side is the "one" side as shown by the left superscript "01" in the designation ${}^{01}A^{01}$ in the right circle; the right superscript "01" indicates the bit position of the flip-flop within the register. The 5C8 inside the circles shows the chassis location of the card containing ${}^{01}A^{01}$ which means chassis number 5 and coordinates C8.

Flip-flops, although composed of two inverters, are sometimes referenced by using only the letter associated with the flip-flop and the digits representing the bit position of the flip-flop. Thus, the ${}^{00}A^{00}$ and ${}^{01}A^{00}$ flip-flop



124.118

Figure 10-2.—Flip-flop (logic symbology).

(the zero and one sides respectively of the lowest order flip-flop in the A-register) becomes A^{00}. The $^{00}T^{05}$ flip-flop is referred to as the T^{05} flip-flop.

INPUT AND OUTPUT LINES

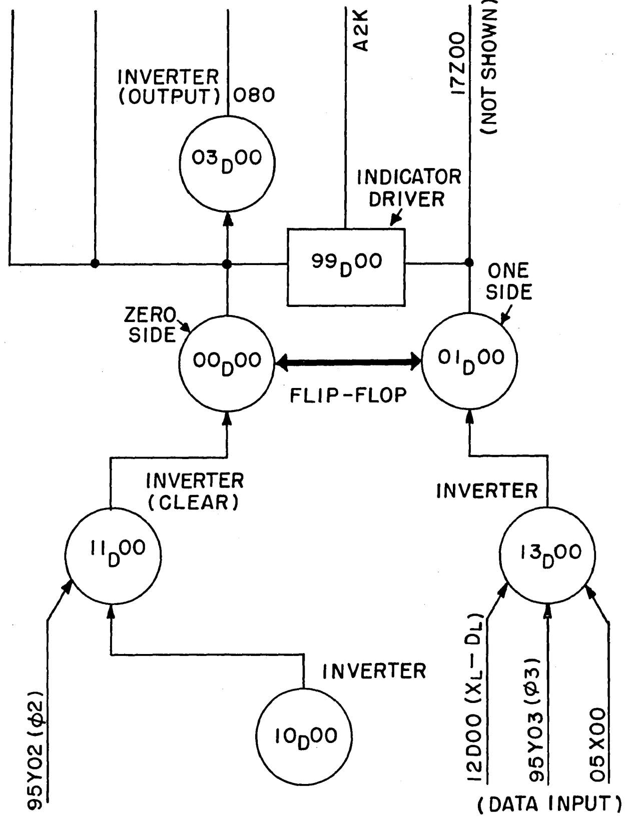
With few exceptions, inputs to all illustrations enter from the bottom of the illustration. In this chapter the information shown at the input line indicates the source of the input and the signal description.

Output lines are at the top of the page. The information shown on these lines indicates the destination of the signal and the signal description.

The output from the indicator driver circuits indicate the panel callout of the indicator lamp associated with a particular flip-flop. The physical location of the indicator-pushbutton on the console (shown later) is also given.

BASIC LOGIC ANALYSIS

The following analysis of data transfer in a representative register (fig. 10-3) in the computer will serve as a basis for an electrical and logical explanation of inverters, flip-flops, and indicator drivers. The first flip-flop stage of the D-register, (D^{00}) its associated indicator driver, an output inverter, the gate used for clearing the stage, and an gate used to enter data into the stage have been arbitrarily chosen.



124.119
Figure 10-3.—Partial function of the schematic of the first stage of the D-register.

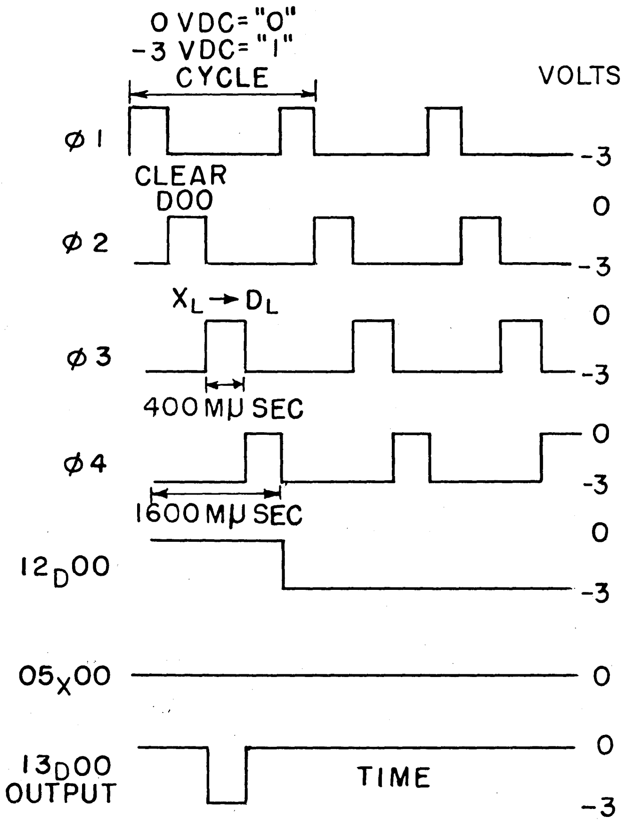
LOGIC

The flip-flop shown (00_D00 and 01_D00) is distinguished from other circuits by the two circles connected by a bold line with arrowheads at each end as explained. Flip-flops are two inverters connected so that the output of one inverter serves as an input for the other inverter. When one circuit is conducting the other is cutoff. The state of the flip-flop indicates the data that is stored. The output available from the zero side is the bit contained in the flip-flop. When the zero side has a "1" output, the stage is SET and is storing a 1; when it has a "0" output, the stage is CLEARED.

MASTER CLOCK

Before proceeding with the discussion of the functions executed by D^{00} (fig. 10-3) consider the following brief description of the master clock. One of the functions of the master clock is to ensure orderly processing of binary information through static circuits (registers). The transfer of data through a register (such as that shown in figure 10-3) is accomplished by pulses generated by the master clock. The master clock system produces outputs in four phases (fig. 10-4) equally spaced in time, which are distributed to all chassis and are available for completing certain functions to permit transfers to take place.

A delay-line oscillator (not shown) which normally operates at 1.25 megacycles generates the basic clock signals. They are applied to shaping circuits which produce accurately spaced and shaped clock pulses. A clock pulse is a "0" (zero volt) for approximately 400 millimicroseconds (0.4 μ· sec) out of each 1600 millimicroseconds (1.6 μ sec). The clock is used in combination with commands (discussed later) to gate binary information into the register.



124.120
Figure 10-4.—Computer clock phases.

DATA TRANSMISSION

The transmission of data to a register is accomplished by clearing the register before the actual transmission takes place. The clearing operation is accomplished by $^{11}D^{00}$ (fig. 10-3). (The number 11 is used to identify the particular inverter stage. The D^{00} is interpreted the same as described earlier.) Stage $^{11}D^{00}$, has a "0" input from $^{10}D^{00}$ at the proper time during the execution of the instruction. The output from $^{11}D^{00}$ is then a "1" when the input from $^{95}Y^{02}$ is a "0" (phase 2 time). This "1" is applied to $^{00}D^{00}$ forcing a "0" output and the cleared state of the flip-flop is established.

The transmission of data to D^{00} is normally through a gate circuit such as $^{13}D^{00}$. (In this case, $^{13}D^{00}$ is used for the $X_L \longrightarrow D_L$ transmission which is executed during an arithmetic operation.) The inputs to $^{13}D^{00}$ are from $^{95}Y^{03}$ ("0" at phase 3), $^{05}X^{00}$ (in the X-register) and $^{12}D^{00}$ (in the D-register).

The phase 3 input is a "0" for 0.4 microsecond of each cycle of the master clock. (Refer to figure 10-4.) Thus, $^{13}D^{00}$ is phase-enabled at phase 3 time. The input from $^{12}D^{00}$ is a "0" when a time in the instruction execution is reached when this transmission must take place. The remaining input (from $^{05}X^{00}$) now determines the bit that is entered into D^{00}. If this input is a "0", the output from $^{13}D^{00}$ is a "1" and D^{00} is set. If the $^{05}X^{00}$ input is a "1", the output from $^{13}D^{00}$ is a "0" and D^{00} remains cleared. The only time $^{13}D^{00}$ has a "1" output is when all inputs are "0's". Note that the flip-flop was cleared on phase 2 and the transmission of data into the register occurred on phase 3. This same procedure is generally used in all registers; i.e., clear on one clock phase and receive incoming data on one of the next clock phases.

The output of $^{00}D^{00}$ is applied directly to other circuits and also $^{03}D^{00}$. The output of $^{03}D^{00}$ is the inverted output of $^{00}D^{00}$; thus it is the same as the output from $^{01}D^{00}$.

INDICATOR DRIVER

The Indicator Driver circuit, $^{99}D^{00}$, is connected to the output of both sides of the flip-flop and has an output to the computer console (shown later). Located on the console is a neon indicator-pushbutton. The indicator is ionized when the flip-flop is set. Pushing the indicator pushbutton clamps the output of the 1 side ($^{01}D^{00}$) and the input to the 0 side of the flip-flop ($^{00}D^{00}$) at ground, "0". This causes a "1" output from $^{00}D^{00}$. In this condition, the flip-flop is set, and the neon indicator lights.

The statements describing $^{00}D^{00}$ and $^{01}D^{00}$ apply to most registers in the computer. Variations of this general operation are discussed where these peculiarities exist.

CIRCUIT ANALYSIS

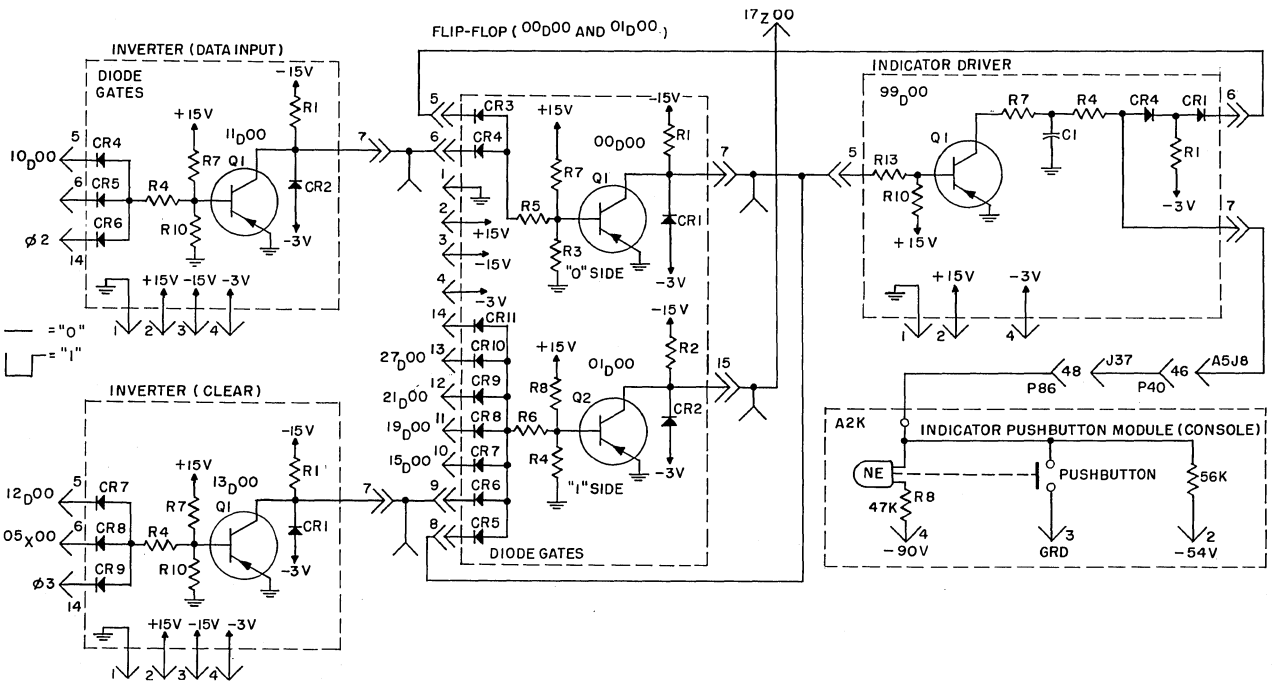
The basic circuits of the computer are the diode gates, flip-flops, and transistor inverters. Figure 10-5 is a representative arrangement of these circuits.

INVERTER

The schematic of a diode gate and transistor inverter is shown in a simpler circuit arrangement in figure 10-6A. This particular circuit has three inputs and one output but other circuits may have up to 10 inputs and five outputs. The transistor, Q1, is a PNP type and for conduction the base must be negative with respect to the emitter. The base of Q1 is connected to a voltage divider consisting of R7 and R10 between +15 VDC and ground. This places a positive potential on the base of Q1 and the transistor is at cutoff unless the inputs are such that conduction is allowed.

Assuming the transistor is at cutoff, the output is -3 VDC ("1"). This -3 VDC output is provided by the clamping action of CR1 (CR1 conducting). Thus, when the transistor is at cutoff, the output is -3 VDC.

If the inputs are such that the transistor is conducting, the ground applied to the emitter is effective through the low resistance of the transistor and the output is then grounded ("0"). The inputs of Q1 are via CR4, CR5, and CR6. Assume the inputs to be -3 VDC to CR4 and 0 VDC to CR5 and CR6. The -3 VDC on the cathode of CR4 causes it to conduct; the voltage developed across R4 and R10 reverses the bias on CR 5 and CR6 and they are at cutoff. The current through CR4, R4, and R7 also causes the voltage at the base of Q1 to go from a small positive potential to a small negative voltage, (due to the increase in voltage across R7) thus causing the transistor to conduct, and the output at the R1-CR1 junction is effectively grounded. The transistor conducts any time one or more of the inputs is -3 VDC ("1"). The only time the transistor is at cutoff is when all the inputs are grounded.



124.121

Figure 10-5.—First stage of D-register.

If the cathodes of CR4, CR5, and CR6 were all at ground potential, each would conduct a small amount and R4 would be effectively grounded as shown in figure 10-6B. The voltage divider now makes the potential at the base of Q1 slightly positive which will cut off the transistor, and the output is -3 VDC.

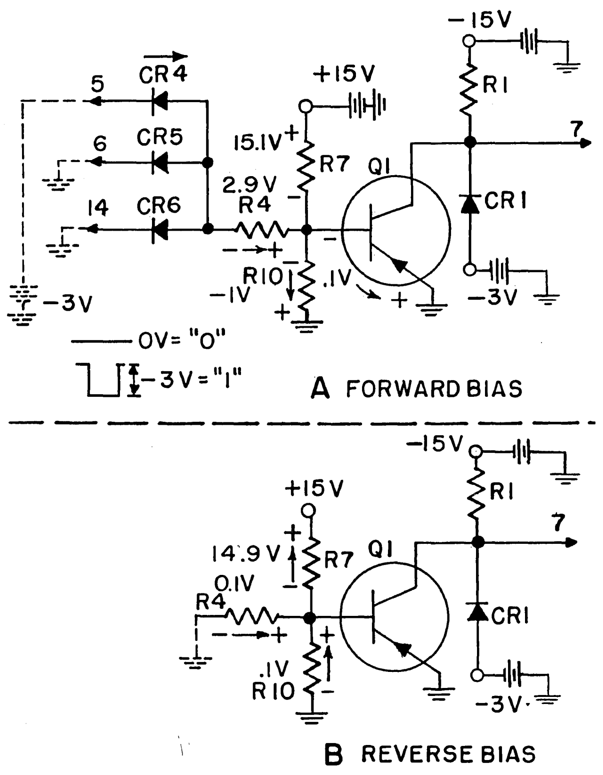
Two statements summarize the results of the diode gate-transistor inverter circuit: (1) If any of the inputs is -3 VDC ("1") the transistor is conducting and the output is ground ("0"); (2) if all of the inputs are at ground potential (0 volts) the transistor is at cutoff and the output is -3 VDC ("1").

FLIP-FLOP CIRCUIT

A flip-flop is made up of two inverters with the output of one connected to the input of the other. If one transistor is conducting, the other is cutoff. Figure 10-5 shows a flip-flop (D^{00}) and some of the circuits associated with it. The transistors of the flip-flops are identified as the one side or the zero side. In this figure, Q1 is the zero side and Q2 is the one side.

Assume a clear command has been initiated, and that the input from $^{10}D^{00}$ via CR4 to $^{11}D^{00}$ is a "0". Also assuming there is no input (open circuit) to pin 6 of the inverter Q1. When phase 2 is "0", Q1 will be cutoff and the output on pin 7 is -3 VDC. This output is applied to CR4 of $^{00}D^{00}$ (whose input was previously short-circuited by the conduction of Q1 of $^{11}D^{00}$). This action causes CR4 to conduct, and a negative potential is applied to the base of Q1, ($^{00}D^{00}$), allowing it to conduct and its output is ground (0 volt). This output is applied to CR5 which is an input to $^{01}D^{00}$.

The other input to $^{01}D^{00}$ (shown at CR6) is also a "0" because the phase input to CR6 from ($^{13}D^{00}$) at phase 3 is a "1". In this condition Q1 of $^{13}D^{00}$ is conducting and its output is 0 volt. Thus, all inputs to $^{01}D^{00}$ are "0"; the base of Q2 is a positive potential, and Q2 is cutoff to produce a -3 VDC output. This -3 VDC is applied to CR3 of Q1 in the flip-flop and holds Q1 in the conducting state ("0" output). Transistor Q2 remains in the cutoff state ("1" output) until other control inputs are applied to Q2. This state of the flip-flop is called the clear state.



124.122
Figure 10-6.—Diode gate and transistor inverter.

Proceeding from this clear state, assume (1) that the command enable to $^{13}D^{00}$ from $^{12}D^{00}$ (via CR7) is a "0" state (command enable for $X_L \rightarrow D_L$ transfer; (2) that phase 3 time is present so the input to CR9 is also a "0"; and (3) that the output from $^{05}X^{00}$ to CR8 is also a "0". Under these conditions all inputs to $^{13}D^{00}$ are at 0 volts; Q1 of $^{13}D^{00}$ is at cutoff, and the output of -3 VDC is applied to CR6 of $^{01}D^{00}$. This causes Q2 of $^{01}D^{00}$ to conduct, and its output is a "0" that is applied to CR3 of $^{00}D^{00}$. Since this is taking place at phase 3 time, the output of $^{11}D^{00}$ is also a "0" and the result is that Q1 ($^{00}D^{00}$) is at cutoff and has a -3 VDC output to CR5 of $^{01}D^{00}$ which maintains the flip-flop in this new state the one side ($^{01}D^{00}$) is conducting and has a ground ("0") output, and the zero side ($^{00}D^{00}$) is at cutoff and has a -3 VDC ("1") output. As previously mentioned, this state is called the SET state. The flip-flop remains set until it is again cleared.

MANUAL SET

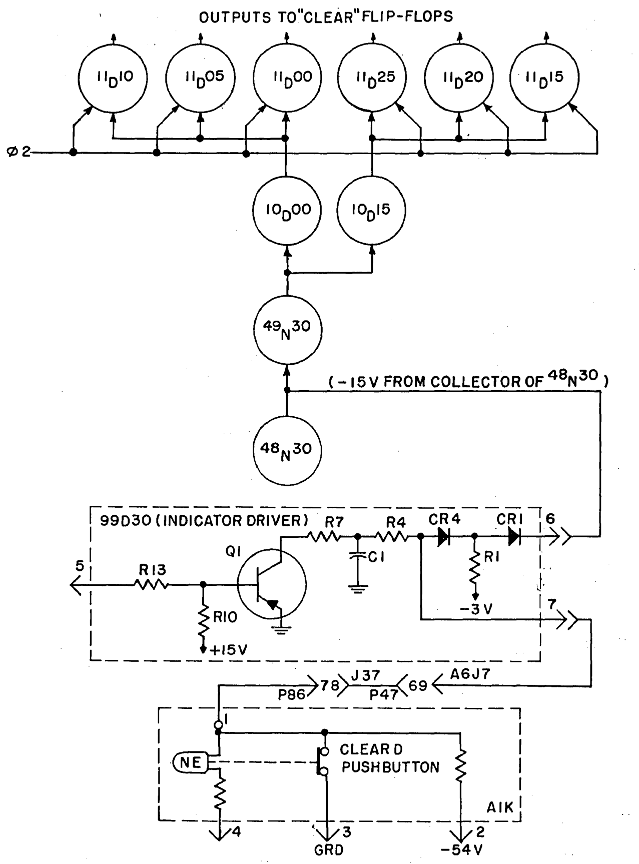
The remaining circuit associated with this flip-flop is the indicator driver, $^{99}D^{00}$. The indicator driver has a direct input from the zero side of the flip-flop. When the flip-flop is set, this input is -3 VDC which causes Q1 ($^{99}D^{00}$) to conduct and supply a near-ground potential to pin 7. This low potential is applied through interassembly wiring to pin 1 of A2K (the indicator pushbutton on the console). In this condition, the neon lamp has sufficient voltage across it (applied in the circuit containing the -90 volt supply, R8, the neon lamp, R4, R7, and Q1) to cause the lamp to be ionized, thus giving an indication that the flip-flop is set.

During the CLEAR state of the flip-flop, Q1 ($^{99}D^{00}$) is at cutoff, and the potential across the lamp is approximately 36 volts, which is not enough to ionize the lamp. The 36 volt potential is a result of the -90 VDC on pin 4 of the indicator module and the -54 VDC on pin 2. Thus, the lamp is extinguished when the flip-flop is cleared.

The other function of the indicator driver is to provide a method to manually set the flip-flop. This is accomplished by depressing the indicator. Depressing the indicator closes a switch below the indicator and causes the lamp to be ionized. The closing of the switch also supplies a ground through the interassembly wiring to the anode of CR4 ($^{99}D^{00}$). Diode CR4 conducts and grounds the anode of CR1. The cathode of CR1 is connected to the output of Q2 ($^{01}D^{00}$). If the output of Q2 is grounded nothing will happen because the flip-flop is already set. If, however, the output of Q2 is -3 VDC, CR1 ($^{99}D^{00}$) conducts and clamps the output of Q2 ($^{01}D^{00}$) at ground. Thus, the input to CR3 ($^{00}D^{00}$) is clamped at ground and the flip-flop is set. When the flip-flop is set, Q1 ($^{99}D^{00}$) conducts and permits a high potential (approximately 90V) to be developed across the lamp. Thus, the lamp remains ionized after the indicator pushbutton is released.

MANUAL CLEAR

Also associated with each register and some sequences is a "manual clear" button. The connections for the clear D pushbutton are shown in figure 10-7. (The circuit is shown for a 6-bit register.) The manual clear circuit is the manual set circuit used with each flip-flop. However, there is no input to Q1 on the indicator driver



124.123

Figure 10-7.—Manual clear.

circuit. Thus, pushing similar button located at A1K (on the console) applies a ground through the interassembly wiring to CR4 ($^{99}D^{30}$). Diode CR4 clamps the anode of CR1 at ground which clamps the output of $^{48}N^{30}$ at ground. Thus, the input to $^{49}N^{30}$ is a "0" that is inverted and applied to $^{10}D00$ and $^{10}D15$. The output from $^{10}D00$ and $^{10}D15$ ("0" output) is applied to $11D00$, $11D05$, $11D10$, $11D15$, $11D20$, and $11D25$. When the phase 2 inputs are "0", the ^{11}D— circuits have a "1" output to the zero side of the flip-flops (not shown) which causes these flip-flops to be cleared. Thus, pushing the clear pushbutton clears the entire register.

OPERATION, MODES, CONTROLS, AND INDICATORS

Before the computer can begin to execute a program, certain manual operations must be performed. The following will explain the functions of the manual controls located on the control panel (fig. 10-8).

Each register in the computer is displayed in the form of neon indicators on the console. When the indicator is ON, that bit position holds a "1"; conversely, when the indicator is off, that bit position holds a "0". (The X register indication is the reverse of the foregoing statement.) The flip-flop for each bit position can be manually "set" to the "1" state, and each register can be "cleared" to the "0" state (as discussed earlier). The pushbutton switch (which is actuated by depressing the indicator) is the "manual set" button for that bit position. Depressing the indicator pushbutton sets the associated flip-flop and lights the indicator. The button to the right of each register is the "manual clear" button. The X register will hold all "1's" when the manual clear-button is depressed and a bit position in X will contain a "0" if its corresponding "manual set" button is depressed. A lighted indicator on all other registers indicates a set condition for that bit position.

The following is a brief explanation of the manual controls across the bottom right corner of the console, (fig. 10-9).

1. MASTER CLEAR.—If the MASTER CLEAR switch is placed in the "momentary" down position, all indicators on the console will be extinguished and all registers cleared. (The computer cannot be "master cleared" in this way if it is in the High Speed mode of operation.) In the "momentary" up position, the A sequence (discussed later) will be enabled. The normal position or center position of the Master Clear switch causes no action in the computer.

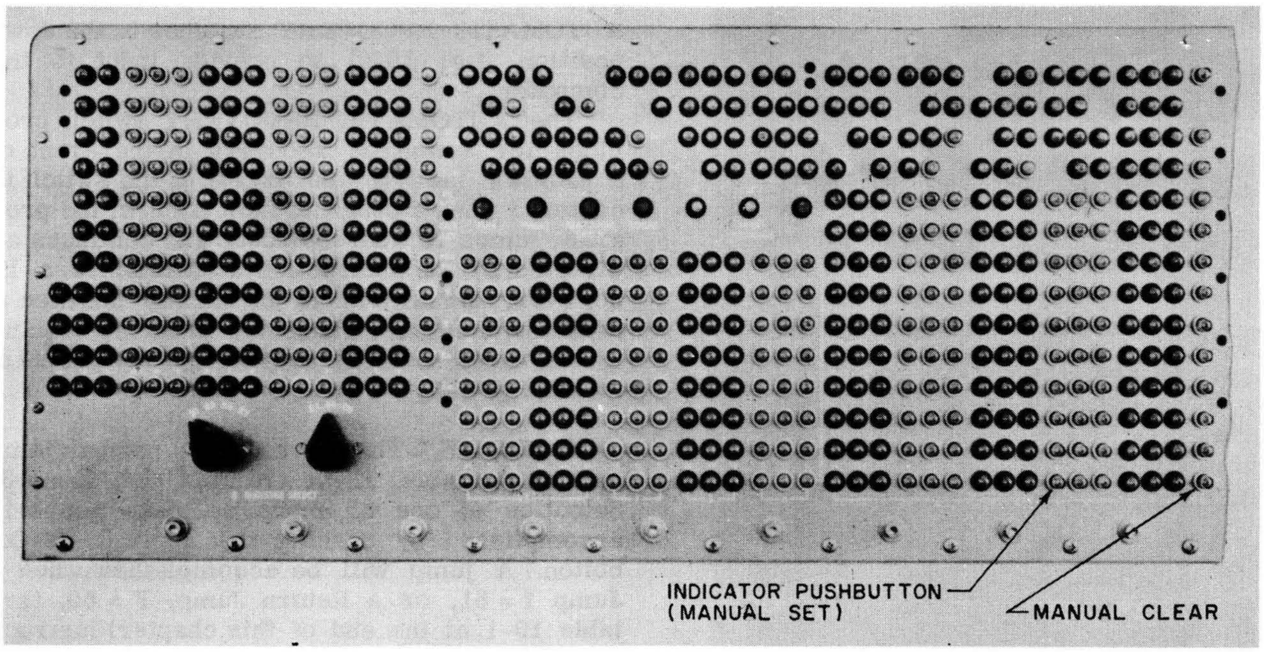
2. HIGH SPEED.—The most common mode of operation for the computer is the high-speed mode. This is the automatic mode of operation in which the computer executes and routines. To enter the high-speed mode of operation set controls as follows:

a. Operate the MASTER CLEAR switch to the down position.

b. Enter the address of the first instruction in the P-register.

c. Operate the MASTER CLEAR switch to the up position. This enables the A sequence.

d. Operate the HIGH SPEED switch to the down position. The RUN indicator lights, and the computer is in the high-speed mode of operation.



124.124

Figure 10-8.—Computer console (bit indicators).

Termination of operations by a STOP condition can be automatically overcome by placing the HIGH SPEED switch to the up (locked) position. This is referred to as abnormal high-speed operation.

3. OPERATION STEP.—The Operation-step mode is the operation that allows only one instruction to be executed and then an automatic stop. The Operation-step mode is performed as follows:

a. Exit from the High-speed mode is accomplished by operating the OPERATION STEP switch to the down position. This causes the computer to stop operation at the beginning of the next B sequence.

b. Operate the OPERATION STEP switch to the down position. This causes the computer to execute one instruction and again stop at the beginning of the next B sequence. Each successive operation of the OPERATION STEP switch causes one instruction to be executed.

Another function of the OPERATION STEP switch is to provide an automatic low-speed operation. Placing the switch in the up position allows the repetition rate of instructions to be governed by a variable, low-speed oscillator (not shown). The low-speed oscillator operates between 2 and 200 cycles per second and is controlled by the LOW-SPEED CONTROL.

During either mode of operation involving the OPERATION STEP switch means of performing these instructions only the repetition rate is controled. The execution time of each instruction is a function of the computer.

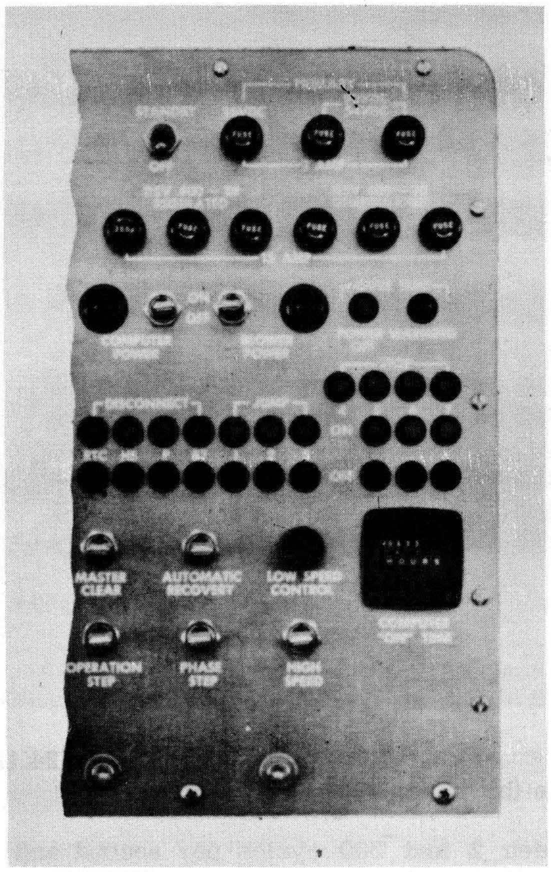
4. PHASE STEP.—The phase-step mode is the operation that allows one clock phase to be produced each time the PHASE STEP switch is depressed. Enter the phase-step mode of operation as follows:

a. Operate the OPERATION STEP switch to the down position. (Exit from high-speed operation.)

b. Select HS DISCONNECT by pushing the HS Disconnect to ON. This disables the automatic cycling of the master clock.

c. Operate the PHASE STEP switch to the down position. This causes one clock phase to be generated. Successive operations of the PHASE STEP switch cause successive clock phases to be generated as indicated by the CLOCK PHASE 1, 2, 3, or 4 indicators.

Automatic low-speed generation of the clock phases becomes a function of the low-speed oscillator when the PHASE STEP switch is



124.125
Figure 10-9.—Computer console (power and control).

placed in the up (locked) position. The rate at which the clock phases are generated is further controlled by varying the LOW-SPEED CONTROL potentiometer. The rate can be varied between 2 and 200 cycles per second. During the phase-step mode of operation, a memory timing chain (which times and controls the read-out of data from memory) is disabled. This action prevents any memory references. Data must be manually inserted via the set pushbuttons on the console.

5. AUTOMATIC RECOVERY.—The loading of routines into the computer is facilitated by the wired memory and the AUTOMATIC RECOVERY switch. The particular wired memory program in the computer is indicated by one of three lights. The WIRED PROGRAM A indicator is lighted for paper tape loading, WIRED PROGRAM B indicator is lighted for intercomputer loading, and WIRED PROGRAM C indicator is lighted for magnetic tape loading.

After preparing one of these units for program loading, operate the MASTER CLEAR and AUTOMATIC RECOVERY switches to the down position (fig. 10-9) to provide input to the computer.

The AUTOMATIC RECOVERY switch provides for automatic recovery in the event of a program fault (f = 00 or 77). If the switch is centered when a program fault occurs, the program jumps to address 00000 and continues as directed by the instruction at that address. If, however, the switch is in the up position a program fault will cause the program to jump to the wired memory and execute the program therein.

6. JUMPS.—There are three manual jump options available. These are JUMPS 1, 2, and 3. Selection of one or more of these jumps is accomplished by pushing the appropriate ON button. A jump will be accomplished when a Jump f = 61, or a Return Jump, F = 65, (see table 10-1 at the end of this chapter) instruction is executed with a j-designator equal to a selected JUMP.

7. STOPS.—Four stop conditions can be imposed on the computer. Three of these can be manually selected, the fourth is a programming function. STOPS 5, 6, and 7 are selected by pushing the appropriate ON button. A stop will be accomplished when a Jump or Return Jump (f = 61 or 65) instruction is executed with a j-designator equal to the selected STOP. When the computer stops, the appropriate STOP indicator lights. If the Jump or Return Jump instruction is executed with the j-designator equal to five, six, or seven, an automatic jump is performed, but the computer stops only if the j-designator is the same as the selected STOP. If the Jump or Return Jump instruction has a j-designator of four, an automatic jump-and-stop is performed and indicated by the STOP 4 indicator. After the computer stops, it can be started by operating the HIGH SPEED or OPERATION STEP switch.

8. DISCONNECTS.—Four disconnects are provided to alter certain computer operations. A DISCONNECT is selected by pushing the appropriate ON button. The disconnect is released by pushing the appropriate OFF button.

a. DISCONNECT RTC.—Selecting this disconnect prevents the updating of the Real-Time address, 00036.

b. DISCONNECT HS.—Selecting this disconnect disables the high-speed generation of clock phases and allows the PHASE STEP switch to assume control of generating the clock phases.

c. DISCONNECT P.—Selecting this disconnect prevents the contents of the P-register from being increased. This causes the same instruction to be read up and executed until the disconnect is released.

d. DISCONNECT B7.—Selecting this disconnect disables the countdown capability of the B7-register.

9. REGISTER CONTROLS.—The remaining array of pushbutton indicators (fig. 10-8) displays the contents of registers, designators, circuits and timing chains. Pushing the light assembly closes switch contacts that set the stage to which the light is connected. When the indicator is lighted, that stage is set. The registers, designator circuits and timing chains may be cleared by pushing the appropriate CLEAR button.

10. OVER TEMP.—There are two over temperature sensing circuits (not shown). When the internal temperature exceeds 46° C (115° F), the OVER TEMP WARNING indicator lights. Also, a 400-cycle alarm horn sounds. Pushing the indicator (fig. 10-9) will silence the horn but the indicator remains on. When the temperature exceeds 60° C (140° F), the computer power is removed and the OVER TEMP POWER OFF indicator lights.

SEQUENCES

The major sequences A, B, C, and D, are used to generate the accurately timed command signals which cause specific actions to be accomplished in the various major sections of the computer in order to effect the execution of each instruction.

In a general sense, the A sequence controls the timing necessary to obtain an instruction from memory and place it in the 30-bit U-register. More specifically the A sequence produces command signals which cause: (1) the B sequence circuit to be enabled in preparation for clearing these circuits; (2) the P-adder, i.e., the number in the P-register which represents the next instruction to be executed, to be transferred to the S-register (the address selector circuit which, as discussed in chapter 4, controls the readout from memory of data necessary to execute a particular instruction); (3) the clearing of the P-register in preparation for receiving the number of the next instruction; (4) the clearing of the U-register in preparation to receive the instruction which is to be read from memory; (5) the transferring of data in the Z-register (an intermediate storage register into which all data is read in transit to or from memory) to the U-register; and (6) the clearing of the R-register in preparation to receive data from one of the index registers. Many other actions are caused to be executed by command signals generated during the A sequence. The six actions just described are of particular importance at this time.

The B sequence is primarily involved in obtaining the operand, although in special cases it performs some prearithmetic operations, such as sign correction for multiply and divide. The B sequence also initiates the C sequence as determined by the function code and k-designator.

The C sequence initiates and controls all arithmetic operations as necessary to execute the required function. The C sequence also controls certain other instructions that utilize registers in the arithmetic section. For example, consider an f=10 instruction (an ENTER Q instruction as shown in table 10-1 at the end of this chapter). During the B sequence, the operand is placed in the D-register (see fig. 8-1). The C sequence then causes the X-register to be cleared, and the transmission of (D) to X ($D \longrightarrow X$). Further, the C sequence causes the Q-register to be cleared and transmits (X) to Q ($X \longrightarrow Q$). Afterwards, it initiates the sensing of the j-designator to determine if a skip is to be executed.

The D sequence provides: (1) the timing for storing or replacing instructions in memory; (2) control for reading an external function word from memory for output transmission on channel j (an input/output channel); (3) commands for storing P+p if f=64, or 65 and the jump condition is satisfied; and (4) commands for positioning of buffer control words.

The major sequences A, B, C, and D are performed successively to execute an instruction. While the execution of every instruction (except the I/O instructions) is accomplished by means of the A, B, and C sequences, many instructions do not use the D sequence. Only those instructions which store the operand use the D sequence.

In addition to the major A, B, C, and D sequences, there are minor sequences which perform specific simple operations. These minor sequences are called subsequences. All subsequences are initiated by a major sequence. For example, during the B sequence the operand is read to the D-register. This transfer of data is accomplished by initiation of a read as modified by K subsequence.

A SEQUENCE

To enable the A sequence, flip-flop T^{00} (comprising $^{10}T^{00}$ and $^{11}T^{00}$ in figure 10-10) must be set. This flip-flop can be set via inverter $^{19}T^{00}$ at phase 4 time if any one of certain conditions exists at the inverter $^{18}T^{00}$ input.

Inverter $^{18}T^{00}$ has the following inputs which receive "1" signals if the conditions indicated are satisfied.

NOTE: Because of limitations in the scope of this text, schematic and logic diagrams are necessarily brief and reveal only the major circuits. Consequently the circuit from which a particular signal originates is not always shown.

CIRCUIT (Input origin)	OUTPUT	CONDITIONS
$^{13}J^{68}$	"1"	MASTER CLEAR in up position. (The MASTER CLEAR switch is a manual control used to clear all major operational registers and control flip-flops.)
$^{02}L^{38}$	"1"	Multiply or divide involved in a previous instruction has been completed.
$^{50}T^{21}$	"1"	D sequence operations for previous instructions are satisfied.
$^{15}T^{00}$	"1"	f = Read, C sequence is in operation but f $\neq$ 64, 65 and jump. (See procedure for executing return-jump instruction in chapter 8.)
$^{17}T^{00}$	"1"	f = Read, f $\neq$ 34-37, 44-47, 54-57, and the C sequence is in operation.

The existence of one or all of the preceding list of conditions causes $^{18}T^{00}$ to produce a 0 output and $^{19}T^{00}$ to produce a 1 output thus causing T^{00} to be set, and the A sequence is partially enabled. Before the A sequence can begin to produce, its enabling outputs must be checked. The output from $^{11}T^{00}$ is applied to $^{13}T^{00}$ where the conditions affecting the A sequence are checked as follows:

CIRCUIT	OUTPUT	CONDITIONS
$^{16}E^{00}$	"0"	r (repeat status designator) = 0, (f $\neq$ 70 or B7 (repeat count) = 1.
$^{14}E^{00}$	"0"	No Interrupt request present or j hold clear.
$^{57}H^{01}$	"0"	Memory is available.
$^{03}G^{73}$	"0"	Operation Step flip-flop is set. (This circuit is actuated by the Op Step Control on the console or by the High Speed Run Control when it is enabled.)
$^{11}T^{00}$	"0"	A sequence enabled.

If all these conditions are satisfied, $^{13}T^{00}$ has a "1" output (at phase 2 time) which sets T^{11} (the first flip-flop in the A sequence chain) and the A sequence is initiated.

There are basically three modes of operation for the A sequence: Normal, Repeat, and Interrupt mode.

Normal

Under normal conditions (no repeat and no interrupt) the following events occur:

CIRCUIT	CONDITION	OPERATION PERFORMED
T^{11}	Set	Clear B sequence enable Enable clear Abort flip-flop P-adder$\rightarrow$ S, Initiate memory
T^{13}	Set	Repeat mode check
T^{21}	Set	Clear r-designator Causes output from 10_N30 to abort flip-flop if f = 70 and $B_7 = 0$, or if f = 70 and $B_7 \neq 0$, also j$\longrightarrow$r
T^{23} 61_N20	Set	Clear P S$\longrightarrow$P Set p to +1 (61_N20 causes some actions to be executed at ∅2 time and some at ∅4 time)
T^{31} 16_T31 14_T31	Set	Clear U, Z$\longrightarrow$U Z$\longrightarrow$U (Several outputs directing the Z$\longrightarrow$U transfer are necessary to prevent overloading a single inverter.)
T^{33} 14_T33	Set	Clear R*, $B_b\longrightarrow$R* Enable B sequence j$\longrightarrow$I/O multiplexer
T^{42} 61_N70	Set	If f = 13, 17, 62, 63, 66, 67, 73—76 set I/O Lockout flip-flop (G15)

Clear B enable is performed to disable the B sequence when an abort is being executed. (The word "abort" is used to describe the condition in the computer which results in the next sequential instruction being skipped.) If a skip instruction is being executed an enable is applied to the B sequence after the execution of the abort so the Abort flip-flop can be cleared before the time that the B sequence of the next instruction is to be activated.

Memory is initiated to read the next instruction word from the memory location specified as a result of the P-adder$\longrightarrow$S command. Later, clear P, and S$\longrightarrow$P commands are issued. This places the address of the present instruction word in the P-register. Next, the p-designator is set to +1 in anticipation of increasing the present memory address by one in order to read the next sequential instruction during the next A sequence.

The timing has now progressed far enough through one phase cycle from the initiate memory command to initiate the transfer of Z$\longrightarrow$U. This places the instruction word in the U-register and the designators for this instruction are translated.

The next step is to clear R* followed by $B_b\longrightarrow$R*. The $B_b\longrightarrow$R* transmission occurs in anticipation of modifying the operand designator y (discussed in chapter 8). The I/O Lockout flip-flop (G15) is set. The action of this circuit is described in a later chapter.

The normal A sequence has now expired and the instruction word has been placed in the U-register and B_b transmitted to R* in preparation of modifying the operand designator, y.

If the instruction is an Input/Output instruction, other commands are issued in addition to those already mentioned. The utilization of these commands is explained later.

Repeat Mode

If the instruction just read from memory is to be repeated, the next time the A sequence is initiated, the Repeat mode of the A sequence will become operative. This is accomplished by 16_E00 which senses the r designator and the repeat count. The four-bit r-designator circuit (shown later) is referred to as the Repeat Status designator. This circuit controls the manner in which the contents of U_L are to be modified when an instruction is repeated. The r-designator circuit is set by the transmission of j$\longrightarrow$r (j-designator to the r-designator circuit) during the 70 (Repeat) instruction.

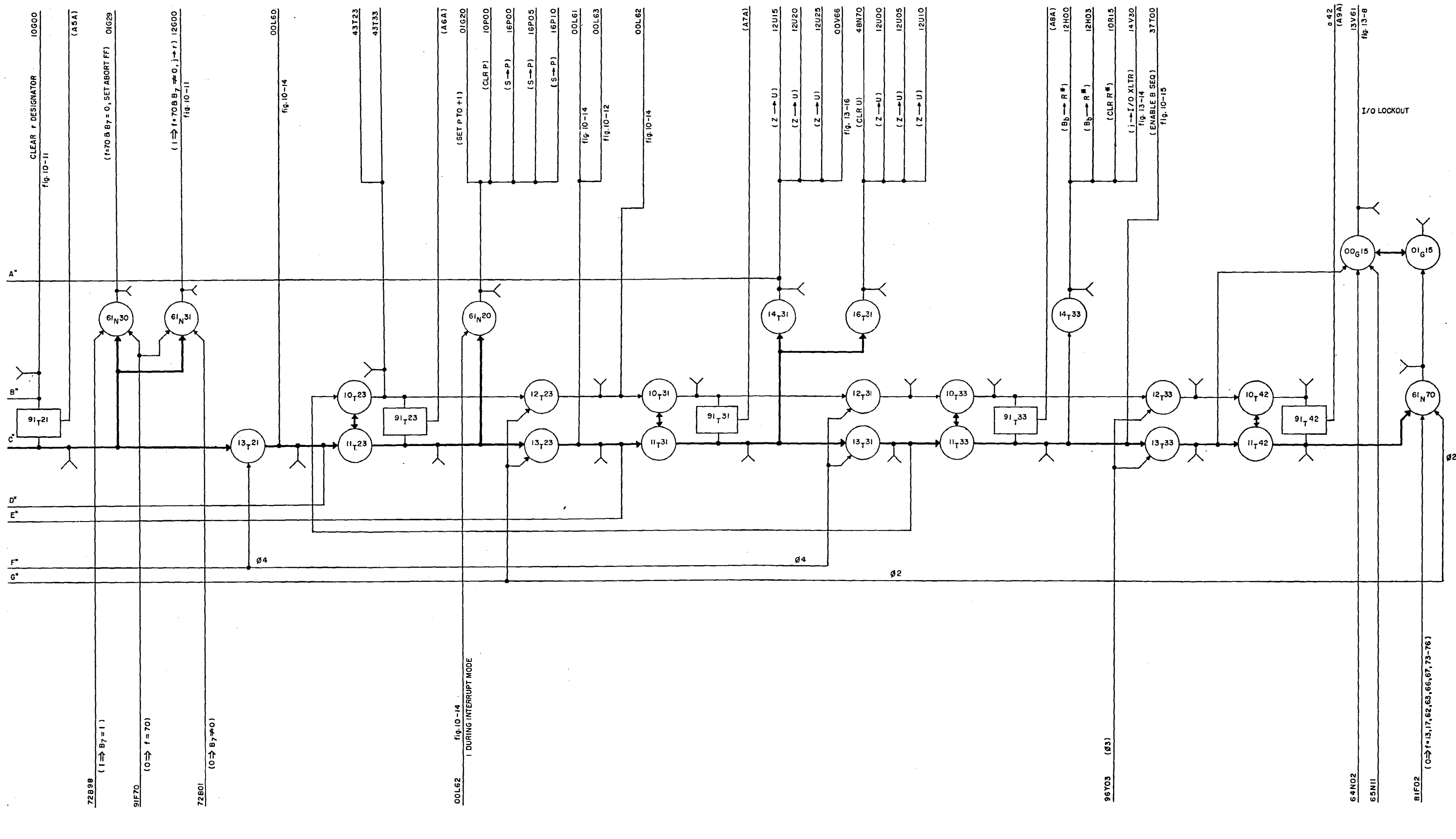
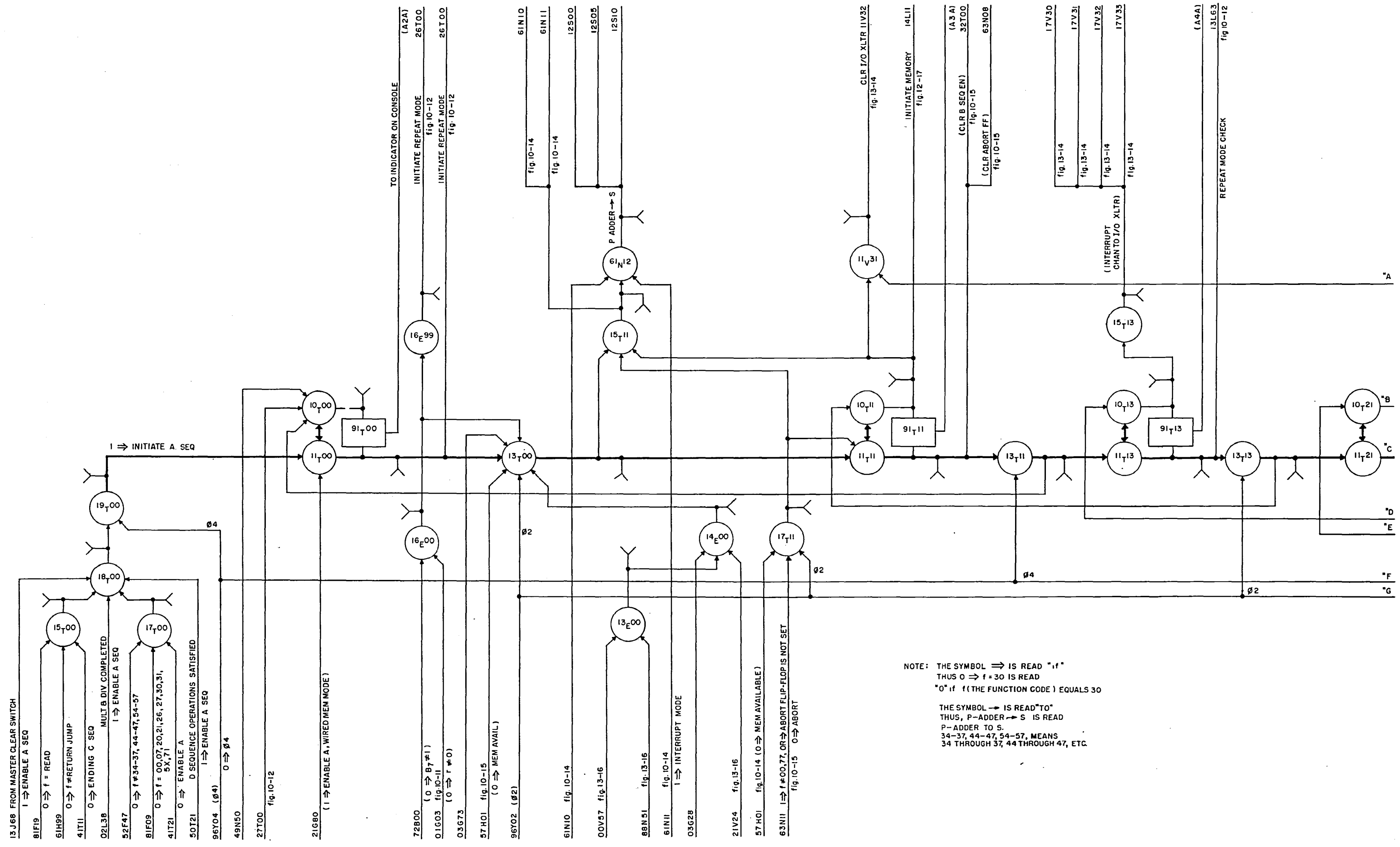


Figure 10-10.—A sequence logic diagram (part I).



NOTE: THE SYMBOL ⇒ IS READ "if"
THUS 0 ⇒ f = 30 IS READ
"0" if f (THE FUNCTION CODE) EQUALS 30

THE SYMBOL → IS READ "TO"
THUS, P-ADDER → S IS READ
P-ADDER TO S.
34-37, 44-47, 54-57, MEANS
34 THROUGH 37, 44 THROUGH 47, ETC.

124.126

Figure 10-10.—A sequence logic diagram (part I)—Continued.

When the command j$\longrightarrow$r is executed, G^{03} (fig. 10-11) of the r-designator unconditionally set via $^{12}G^{00}$ and $^{13}B^{03}$ at Ø1 time. Stages G^{00}, G^{01}, and G^{02}, take on the value of j for the Repeat instruction (f = 70). The r-designator is the j-designator plus an octal 10 (r = 12 is the same as j = 2). Exit from a Repeat mode can only be accomplished by the satisfaction of a conditional skip for the instruction being repeated, or upon exhaustion of the repeat count. If the instruction is to be repeated, $^{01}G^{03}$ has a "0" output and if there is a repeat count greater than one, the B_7 = 1 circuit (not shown) has a "0" output. Thus, the output from $^{16}E^{00}$ (fig. 10-10) is a "1" that disables the normal A sequence (by the action at $^{13}T^{00}$) and initiates the Repeat mode via $^{16}E^{99}$ and $^{26}T^{00}$ if the Operation Step flip-flop (not shown) is set. When the Repeat mode is initiated, two things must happen: (1) decrease the repeat count by one; and (2) make preliminary data transfers in anticipation of modifying y.

1. DECREASE REPEAT COUNT.—The repeat count indication is stored in the B7-register, (not shown) and is decreased by L63 and L64 (fig. 10-12) as follows:

$^{01}L^{63}$	Clear R to negative zero (all "1's"$\longrightarrow$R)
$^{00}L^{63}$	B7 - 1$\longrightarrow$R
$^{00}L^{64}$	Clear B7; R$\longrightarrow$B7 (if DISCONNECT B7 switch is not selected).

As a result, the repeat count is decreased by one at the beginning of the execution of the repeated instruction. The second operation (preliminary modifying operation) is accomplished by the $^{2\text{-}}T^{00}$, (The dash between the 2 and the T indicates either the one or zero side of the flip-flop) $^{2\text{-}}T^{01}$, and $^{2\text{-}}T^{02}$ flip-flops and associated inverters. There are three modifications which can be initiated by the Repeat mode: y + 1, y - 1, or y + B_b. Which of these is done is determined by the r-designator.

Consider the situation where r = 13 or 17 and b = 7. When r = 13, the j-designator is 3, and is interpreted as follows: Skip the next instruction if Y is greater than (Q), (Y>Q). When r = 17, the j-designator is 7 and is interpreted as follows: skip the next instruction if, Y is greater than (A), (Y>A). The b-designator (7) denotes: add (B^7), the contents of the B^7 register, to y. Thus if r = 13 or 17 and b = 7, the computer must wait for the repeat count operations involving B7 to be completed before B7 can be added to y. This is done by $^{23}T^{00}$ which senses $^{21}G^{00}$ (inverter not shown) and supplies a "0" input if r = 13, 17, and $^{13}H^{08}$ (inverter not shown) which supplies "0" if b = 7. At the same time, $^{25}T^{00}$ is sensing $^{22}G^{00}$ (inverter not shown), which supplies a "0" input when r ≠ 13, 17 or b ≠ 7. The conditions affecting $^{24}T^{00}$ and $^{25}T^{00}$ cannot both exist at the same time. Thus, if r = 13 or 17 and b = 7, the timing delay provided by the $^{2\text{-}}T^{01}$ flip-flop is utilized (T^{01} is set at Ø1 time) and on the next Ø4, $^{2\text{-}}T^{02}$ is set. This permits time for the repeat count to be exhausted. If, however, r ≠ 13, 17 or b ≠ 7, the "1" output from $^{25}T^{00}$ sets $^{2\text{-}}T^{02}$ directly, bypassing $^{2\text{-}}T^{01}$. When $^{2\text{-}}T^{02}$ is set, the following events occur:

$^{20}T^{02}$	Clear R* (in preparation for transferring data to this register)
$^{21}T^{02}$	Enable $^{62}N^{30}$ and on the next phase 3 enable B sequence and clear Operation Step flip-flop Enable $^{62}N^{33}$

Also, the "1" output from $^{20}T^{02}$ is inverted by $^{23}T^{02}$ and enables $^{62}N^{32}$.

The modification of y is accomplished by placing the modifier in R* and utilizing the U_L + R* Adder function. The data used as the modifier is determined in this portion of the Repeat mode via $^{62}N^{33}$ and $^{62}N^{32}$. Inverter $^{62}N^{33}$ senses $^{21}G^{00}$ (inverter not shown). Stage $^{62}N^{33}$ produces a "1" output if r = 13 or 17 which initiates B_b—R*. Inverter $^{62}N^{32}$ senses inverter $^{01}G^{01}$ and $^{00}G^{00}$ (not shown), and initiates -1$\longrightarrow$R* if r = 12 or 16. If r ≠ 12, 16, 13, or 17 a "0" output from $^{23}T^{02}$ enables +1 -R*. If r = 10 or 14, y is not modified. The Repeat mode of the A sequence continues until B7 = 1, indication that the final execution is underway. When B7 = 1, $^{16}E^{00}$ (fig. 10-10) is disabled and when the next A sequence is enabled, the Repeat mode is not utilized.

There is another situation that can exist concerning f = 70. If f = 70 and B7 = 0 it means to repeat the next instruction zero times (or rather do not perform any repeats). This is an Abort condition that is sensed by $^{61}N^{30}$ (fig. 10-10) when T^{21} (normal A sequence) is set. The input from $^{91}F^{70}$ (inverter not shown) is a "0" if f = 70; and if B7 = 0, the input from $^{72}B^{98}$ (inverter not shown) is a "0" and the "1" output from $^{61}N^{30}$ sets the Abort flip-flop which causes the next instruction to be skipped (repeated zero times).

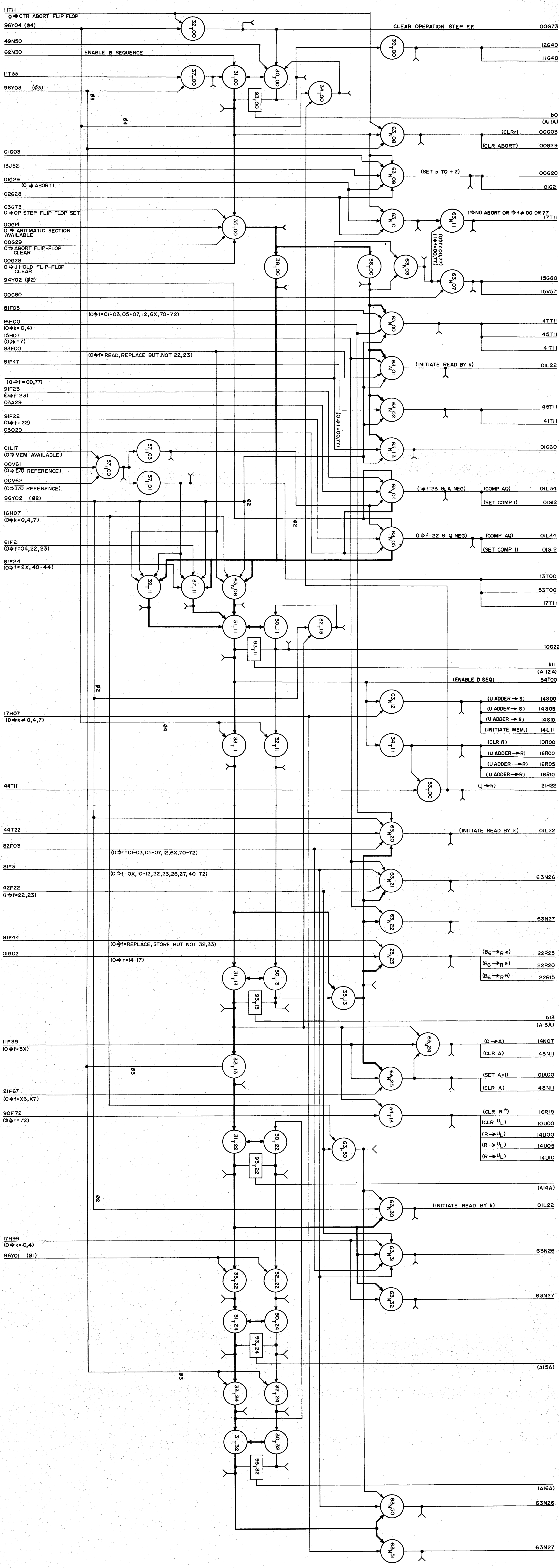
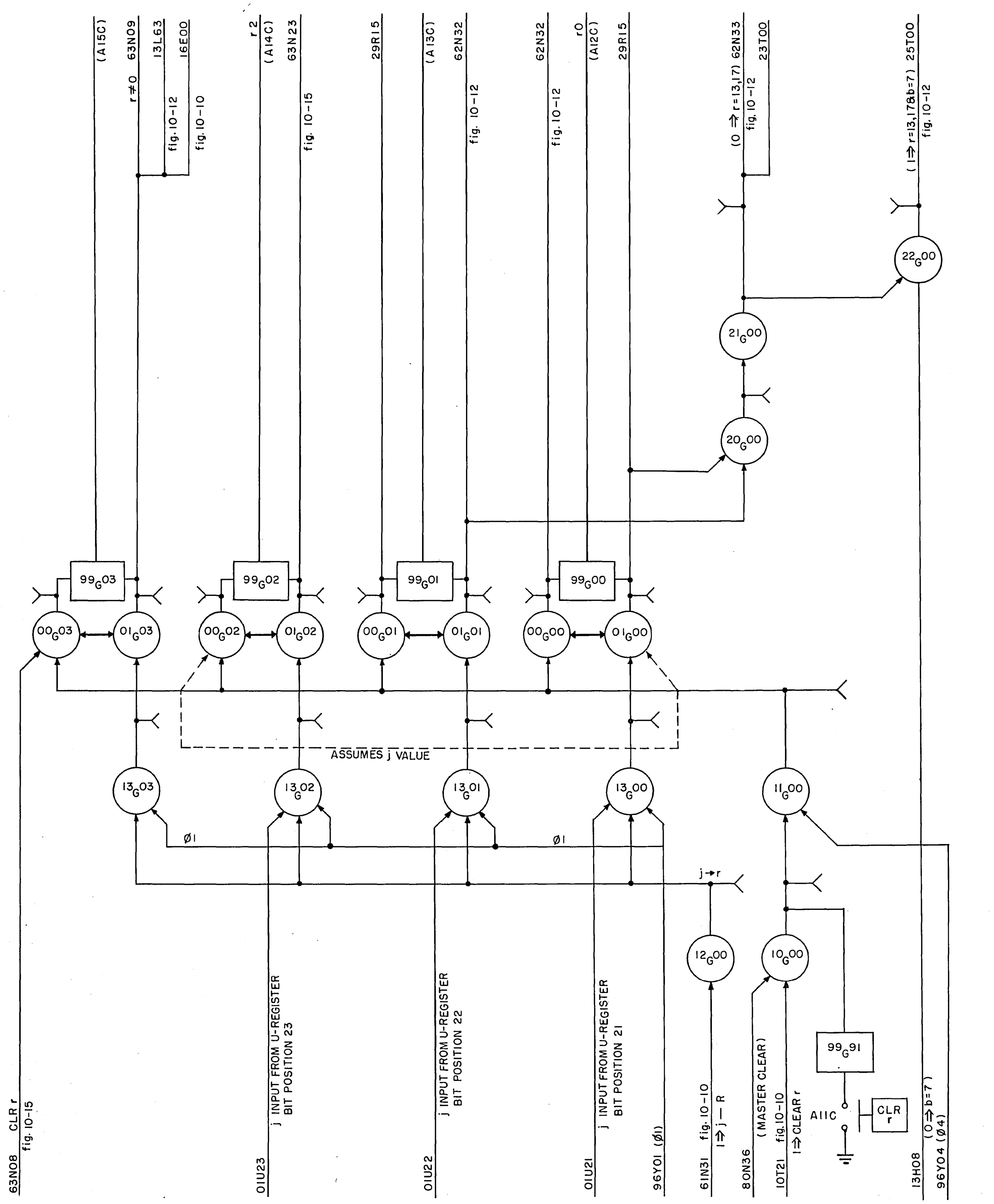


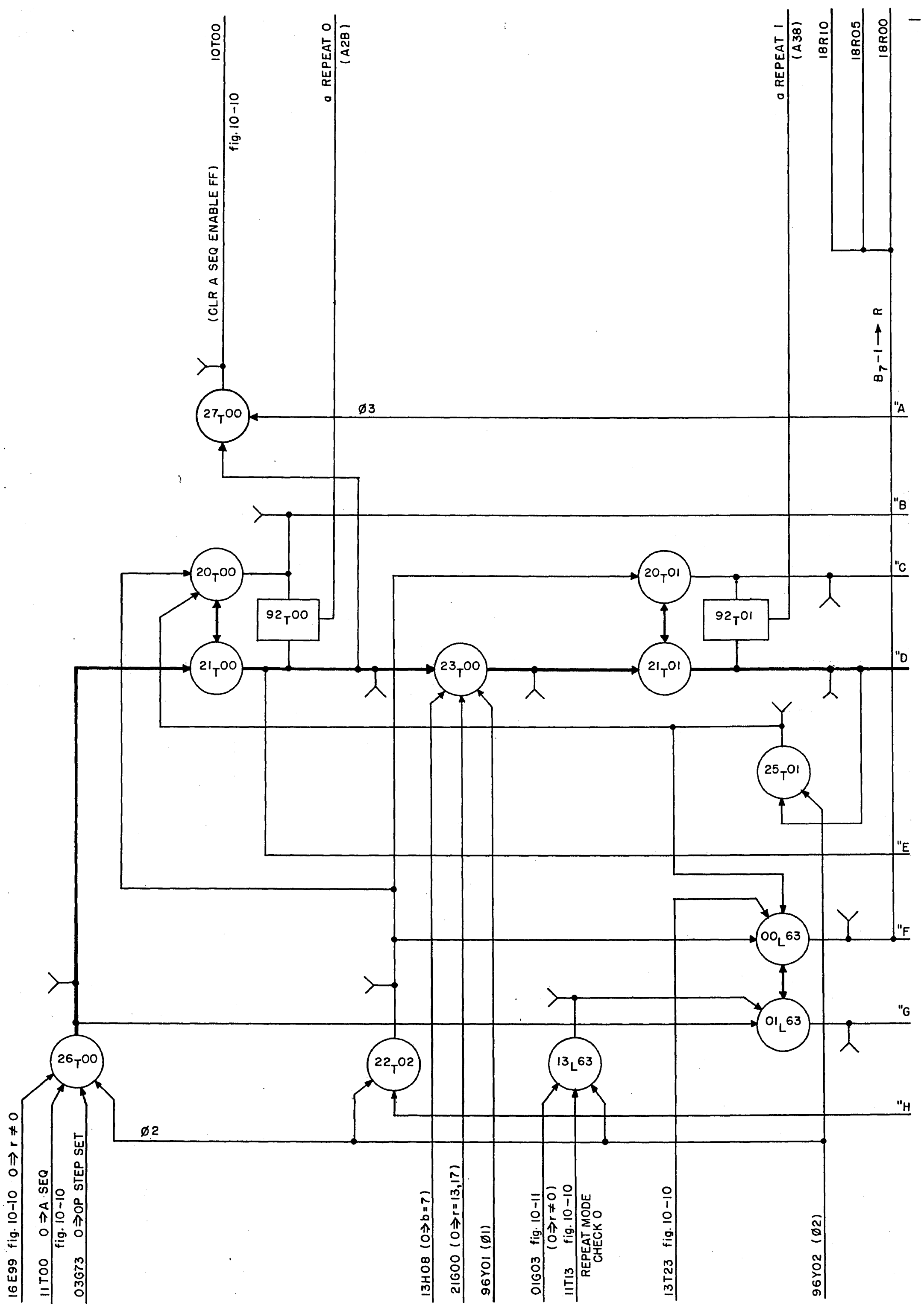
Figure 10-15.—B sequence.

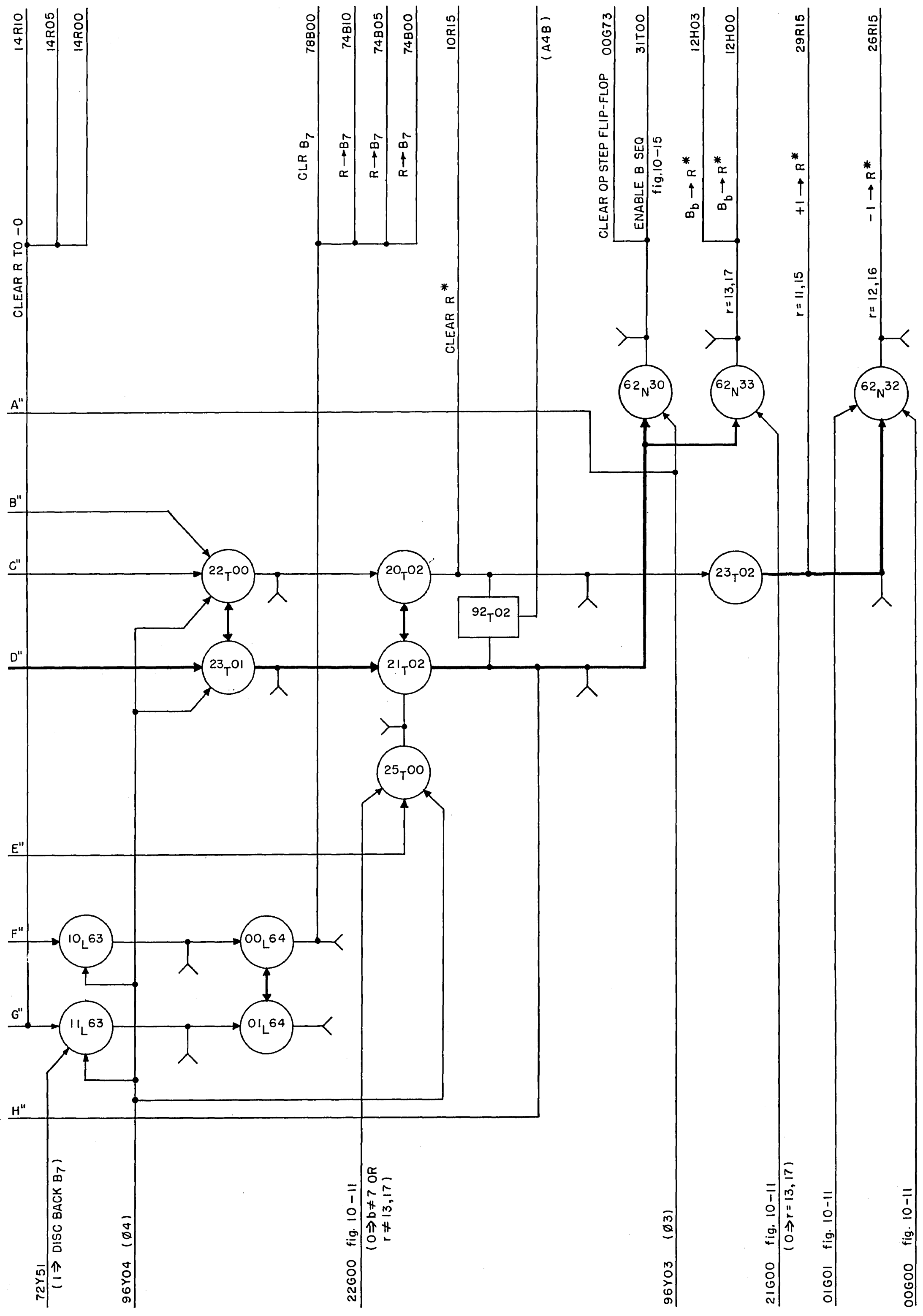
124.131



124.127

Figure 10-11.—r-designator circuit.


10T00
(CLR A SEQ ENABLE FF) fig. 10-10
a REPEAT 0 (A2B)
a REPEAT 1 (A38)
18R10
18R05
18R00
B7-1 → R
Ø3
"A
"B
"C
"D
"E
"F
"G
"H
27T00
20T00
21T00
92T00
23T00
20T01
21T01
92T01
25T01
00L63
01L63
26T00
22T02
13L63
Ø2
16E99 fig. 10-10 0→r ≠ 0
11T00 0→A·SEQ fig. 10-10
03G73 0→OP STEP SET
13H08 (0→b=7)
21G00 (0→r=13,17)
96Y01 (Ø1)
01G03 fig. 10-11 (0→r≠0)
11T13 fig. 10-10 REPEAT MODE CHECK 0
13T23 fig. 10-10
96Y02 (Ø2)



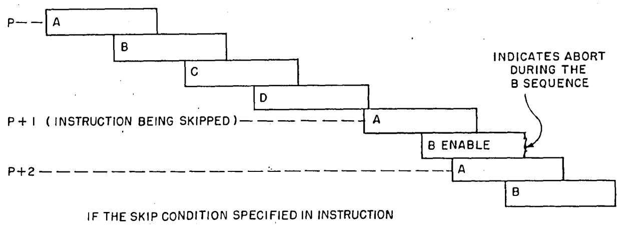
124.128

Figure 10-12.—A sequence logic diagram (part II).

Abort

If an Abort condition exists (other than f = 70 and B7 = 0), the next instruction must be skipped. The skip is accomplished during the A sequence of the instruction which is to be skipped. The A sequence of the instruction that is going to be skipped proceeds normally until T^{33} (fig. 10-10) is set and the B sequence is enabled. The B sequence does not proceed, however, because the Abort flip-flop (not shown) is set. An inverter in the B sequence (not shown) has a "0" output to $^{17}T^{11}$ in the A sequence. If memory is available, flip-flop T^{11} is set and the A sequence is restarted. Consider the following example (fig. 10-13). Instruction P is executed and the Abort flip-flop is set, indicating a skip condition.

The A sequence reads P + 1. During the execution of this instruction the B sequence senses an Abort condition which causes this instruction to be skipped. When the skip condition is sensed, the next instruction to be executed is P + 2.


P
A
B
C
D
P+1 (INSTRUCTION BEING SKIPPED)
A
B ENABLE
INDICATES ABORT DURING THE B SEQUENCE
P+2
A
B
IF THE SKIP CONDITION SPECIFIED IN INSTRUCTION P IS SATISFIED, INSTRUCTION P+1 WILL BE STARTED BUT WILL BE STOPPED, AND THEN INSTRUCTION P+2 WILL BE READ UP BY THE A SEQUENCE AND EXECUTED.

124.129

Figure 10-13.—Abort sequence (skip next instruction).

Interrupt Mode

The Interrupt mode of operation of the A sequence is initiated by $^{61}N^{11}$ or $^{61}N^{10}$ (fig. 10-14). An illegal function code (00 to 77) utilizes $^{61}N^{10}$ and is discussed following the other interrupt ($^{61}N^{11}$). Whenever an interrupt is generated, computer control is assumed by the A sequence using the Interrupt mode of operation. The inputs to $^{61}N^{11}$ are as follows:

CIRCUIT	OUTPUT	CONDITIONS
$00_{G}29$	"0"	No Abort condition
$42_{F}07$	"0"	f ≠ 00, 77 (illegal function codes)
$90_{F}70$	"0"	f ≠ 70
$01_{V}57$	"0"	An interrupt has been generated
$21_{V}24$	"0"	Interrupt address is correct
$15_{T}11$	"0"	A sequence has been initiated

When all of these conditions exist, $^{61}N^{11}$ has a "1" output that disables $^{61}N^{12}$ (fig. 10-10) and prevents P adder⟶S. Instead it initiates Interrupt address (as specified in the interrupt instructions)⟶S and initiates the Interrupt mode timing chain, L^{60}, L^{61}, and L^{62} (fig. 10-14). The A sequence reads the instruction word from the Interrupt address. Then the A sequence proceeds to initiate memory and Interrupt channel⟶I/O translator. When flip-flop T^{23} (fig. 10-10) is set, it enables $^{61}N^{20}$. However, the Interrupt timing chain has been operating in parallel (i.e., at the same time) and L^{62} (fig. 10-14) is set, which disables $^{61}N^{20}$ (fig. 10-10) and prevents clear P, S⟶P, and sets p to +1. This allows storage of (P) +p (where +p represents the numerical value of the jump to be executed) if the instruction read from the Interrupt address is a return jump. This will allow reentry to the main program (at P +p) when the interrupt is completed.

Illegal Function Codes

If f = 00 or 77, it causes the Interrupt flip-flop (not shown) to be set, via the B sequence. The A sequence is also initiated by the B sequence. The P adder⟶S transmission is disabled and memory address 00000 is referenced. By proper programming this address could enter the computer into an error detection or remedial subroutine.

B SEQUENCE

The B sequence can be enabled only from the A sequence using $^{11}T^{33}$ (fig. 10-10) for Normal or Interrupt operation, or via $^{62}N^{30}$

(fig. 10-12) for Repeat operation. When the B sequence is enabled, it first checks the Abort conditions. If the instruction is f = 70 (Repeat instruction) and if B7 = 0, and Abort condition exists (the Abort flip-flop, not shown, is set by $^{61}N^{30}$ shown in figure 10-10 in the A sequence). The Abort condition is sensed by $^{63}N^{10}$ (fig. 10-15) which applies a "1" to $^{63}N^{11}$. The $^{63}N^{11}$ output (0) then initiates the A sequence causing the repeated instruction to be skipped.

When the A sequence is initiated, $^{63}N^{08}$ senses this condition and clears the Abort flip-flop and r-designator circuit. (The r-designator circuit holds the j-designator.) If the Skip condition is a result of j being satisfied, a signal from the B sequence causes the A sequence to be re-initiated and the skip accomplished as explained in the A sequence (see fig. 10-13).

When the B sequence Enable flip-flop ($^{31}T^{00}$) is set, it applies a "0" to $^{35}T^{00}$ where conditions affecting operations are checked. The other inputs to $^{35}T^{00}$ are shown below.

CIRCUIT INPUTS	OUTPUT	CONDITIONS
(Input circuits not shown)		
$^{00}G^{14}$	"0"	Arithmetic section available
$^{03}G^{73}$	"0"	Operation Step flip-flop set.
$^{00}G^{28}$	"0"	j Hold flip-flop clear.
$^{00}G^{29}$	"0"	Abort flip-flop clear.

NOTE: When the A sequence is initiated, $^{32}T^{00}$ (fig. 10-15) clears the Operation Step flip-flop. If the high-speed mode of operation is in effect, the Operation Step flip-flop is immediately set again. If Operation Step mode is in progress, clearing the flip-flop will delay the B sequence until the OPERATION STEP switch is depressed. Clearing the Operation Step flip-flop during a Repeat mode is a function of $^{62}N^{30}$ (fig. 10-12A sequence).

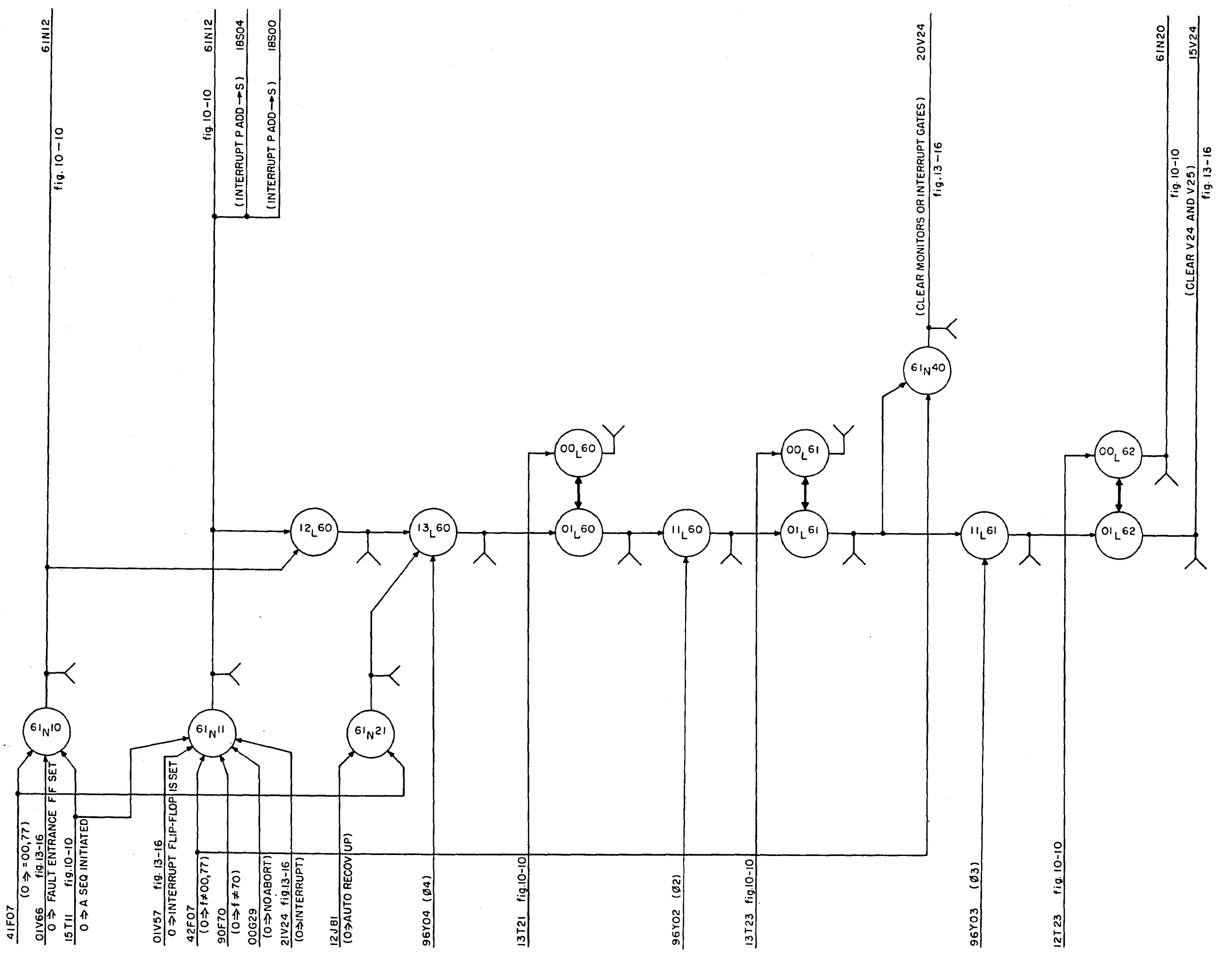
If these conditions exist and the B sequence has been enabled (fig. 10-15) $^{35}T^{00}$ has a "1" output to $^{36}T^{00}$ and $^{38}T^{00}$. These two circuits effectively divide the B sequence into two sections. The circuits associated with $^{36}T^{00}$ are utilized under the following conditions:

CIRCUIT	CONDITIONS	OPERATION
$^{63}N^{00}$	f=01-03, 05-07, 12, 60-67, 70-72, and k=0, 4	Initiate C1 sequence chain (one of two sequence chains C1 and C2, provided in the C sequence, not shown).
$^{63}N^{01}$	f=Read or Replace but not 22 or 23 and k= 7	Initiate read as modified by k.
$^{63}N^{02}$	f= STORE	Initiate C1 sequence.

These operations can proceed immediately because they do not require a memory reference at this time (verify by referring to instructions in table 10-2). (The operand is not obtained from memory when k= 0, 4, and 7.) The circuits associated with $^{38}T^{00}$ are utilized under the following conditions:

CIRCUIT	CONDITIONS	OPERATIONS
$^{63}N^{04}$	f= 23 and A negative, and memory available	Initial sign correction for divide.
$^{63}N^{05}$	f= 22 and Q negative, and I/0	Initial sign correction for multiply.
$^{63}N^{06}$	f= READ or REPLACE and k=0, 4, 7	Initiate B sequence timing chain.

The first operation (from $^{63}N^{06}$) can occur because no memory reference is required at this time. If a memory reference is required to obtain the operand, the status of memory must first be checked. This is accomplished



124.130

Figure 10-14.—Interrupt mode enable.

by ^{57}H00 which checks ^{01}L^{17} (not shown, "0" if no memory reference is in progress) and 00V^{61} and 00V^{62} (not shown, "0" if no I/O reference). If memory is available ^{57}H00 has a "1" output, inverted by ^{57}H01 and enables ^{37}T11 and ^{39}T11 which then initiates the B sequence timing chain if f=04, 22, 23; or Read or Replace instruction. As the timing chain progresses, the circuits associated with the flip-flops reference the appropriate storage location to obtain the operand. After the operand has been obtained, the C1 or C2 sequence is initiated and the arithmetic operations are performed.

Most arithmetic operations are relatively simple and the C sequence maintains all control. Multiply, divide, and shift instructions are more time consuming and complicated. Control of these operations is yielded to a subsequence which controls the execution of the particular instruction.

A schematic diagram and a description of the actions which take place in the C and D sequences are not contained in this discussion.

U-REGISTER

Each instruction, as it is read from a memory location, is placed in the 30-bit U-register (see fig. 10-1). The upper 15 bits of U are used to store the f, j, k, and b designator values. The lower 15 bits of U are the operand address designator, y. The U-register holds the instruction word while it is being executed. All operations that are necessary to execute an instruction are governed by the contents of U.

Basically, an instruction is a 30-bit word that is read from memory and transmitted to the U-register via the Z-register (see figure 8-1 in chapter 8) during the A sequence. The instruction remains in the U-register during execution and until the next instruction is obtained during the next A sequence. Each instruction word which enters the U-register is divided into five designators: f, j, k, b, and y as shown in table 10-1 below.

Table 10-1—Interpretation of Instruction Word Designators.

F	j	k	b	y
Function Code designator (6 bits)	Branch Condition designator (3 bits)	Operand Interpretation designator (3 bits)	Address Modification designator (3 bits)	Operand Address designator (15 bits)

An instruction word sets these designators to certain values. Each of these designators controls a part of an instruction.

The Function Code designator (f), consisting of the six higher order stages of U, holds the code for the current instruction. The function code specifies the functions to be performed by the instruction. A total of 64 function codes make up the repertoire to which the computer will respond. However, Function codes 00 and 77 are nonvalid codes which, if interpreted, cause the computer to jump to a Fault Entrance register located at memory address 00000. The Fault Entrance register could, for example, be used to execute an error detection routine; however, this depends on the program.

The Branch Condition designator (j) interprets for no-skip, unconditional skip, or conditional skip satisfaction for the next sequential instruction. An instruction programmed with a j equal to zero means that no-skip evaluation is to be made; whereas a j equal to one means skip the next instruction unconditionally.

In general, it can be said that for all jump instructions, j specifies the condition for executing the jump. If the jump condition is not satisfied the next sequential instruction is executed in a normal manner. If a jump condition is satisfied, however, then a different program or a remote section of the program presently in use is executed.

The Operand Interpretation designator (k) specifies the manner in which the operand will be treated. Three classes of instructions exist: READ, STORE, and REPLACE. The k-designator is interpreted differently for each class. The READ instructions use k to determine where the operand for the specified operation is to come from. This operand could come from memory, the U_L register, or A-register. For the STORE class instruction, k indicates where data is to be stored; this could be memory locations or the operational registers, A or Q. During the execution of a REPLACE instruction, k signifies not only where the operand is to come from, but also where the result of this operation is to be stored.

The Index Modification designator (b) specifies which B-register, if any, will be used to modify U_L. During the A sequence a command is generated, $(B_b) \longrightarrow R^*$. Should b have a value of 2, for example, the contents of B2 would be inserted in R*. Initiation of the B sequence generates the subsequent command U-adder$\rightarrow$R, or U-adder$\longrightarrow$S. Upon satisfaction of those commands, U_L is now modified by B_b.

The Operand Address designator (y) comprises the 15 lower order bits of the U-register. While its usual function is to specify the memory location of an operand, it can also be used for different functions. Two conditions specify the function of y, one being the class of instruction being executed, and the other being the value of k. For the READ instructions a k of 0 or 4 specifies U_L as the operand. When k is equal to 1, 2, 3, 5, or 6, y specifies the memory location of the operand. When k = 7, the A register is the location of the operand. STORE instructions with k of 1, 2, 3, 5, 6, or 7 causes y to specify the memory location in which the operand will be stored after the operations specified by the instructions are completed. A value of k = 0 specifies

Q and k = 4 specifies the A register as the storage location. For REPLACE instructions with a k of 1, 2, 3, 5, or 6, y specifies the memory location for obtaining the operand and for storing the result. The k values 0, 4, 7, are not used for REPLACE instructions.

FUNCTION CODE DESIGNATOR f (6 BITS)

The function Code designator consists of the six higher order stages of the U-register and holds the code for the current instruction. Octal notation is used in describing this code; e.g., f=22_8 = 010_2 010_2.

There is a total of 64 different function codes as previously stated. All of these except 00 and 77 specify a valid instruction. The two values f = 00 and f = 77 will, if executed, result in a special memory address, called the Fault Entrance address, to be utilized. Each of the six stages of the f-designator circuit contain a flip-flop and a gate circuit. Data from the Z-register (in which all data to and from memory is intermediately stored) is gated into the U-register on clock phase 1 and the flip-flops can be cleared on clock phase 4. Also associated with each stage of the Z-register is an indicator driver, by which each stage can be set and a console indication of the stage condition (set or clear) obtained. (The indicator is lighted if the stage is set.)

Lower Digit Translation

The lower digit translation is made by sensing U26, U^{25}, and U^{24} (fig. 10-16). The $^{10}F^{9-}$ circuits in the function code translator have inputs from circuits of the U-register, and translate accordingly. The following list shows the first lower digit translations:

NOTE: An X preceding or following a number may represent any permissible octal value. This two-digit code may or may not be translated completely. For example: f = X2 indicates the following function codes: 02, 12, 22, 32, 42, 52, 62, and 72. The term f = 2X (shown later) indicates the following function codes: 20, 21, 22, 23, 24, 25, 26, and 27.

In the discussion which follows (using figure 10-16) only these inverters which produce outputs when f = 13 or 17 are shown.

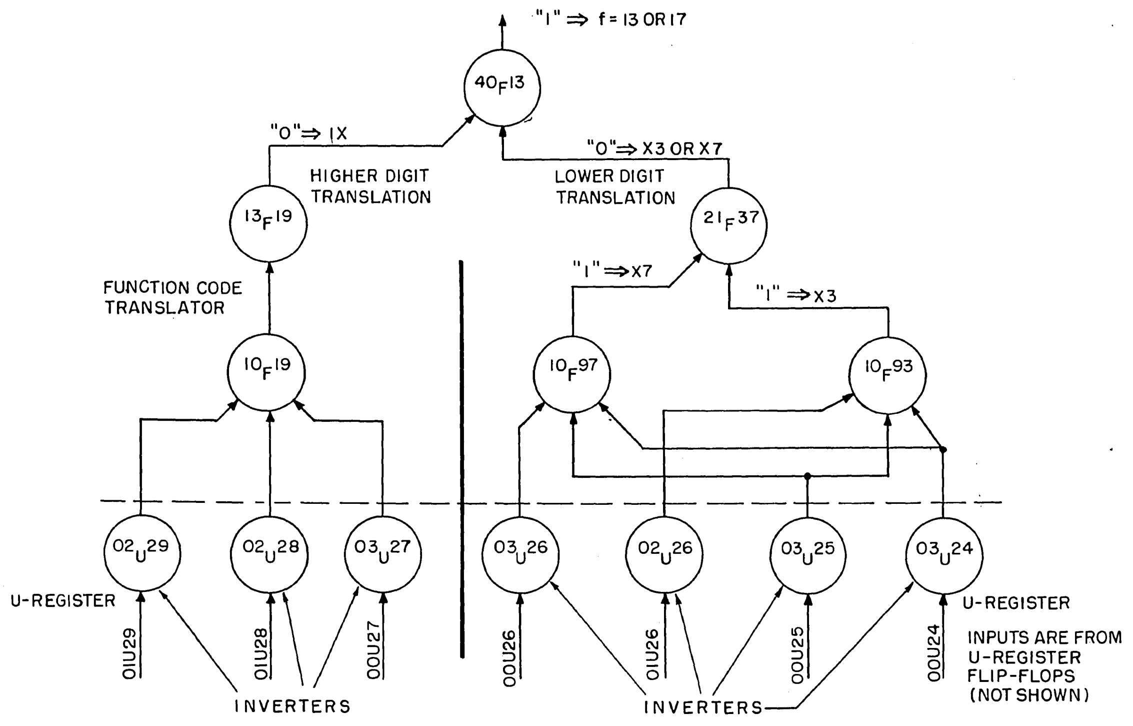
CIRCUIT	OUTPUT	CONDITIONS
02_U26	"0"	f=X0, X1, X2, X3
03_U26	"0"	f=X4, X5, X6, X7
02_U25 (not shown)	"0"	f=X0, X1, X4, X5
03_U25	"0"	f=X2, X3, X6, X7
02_U24 (not shown)	"0"	f=X0, X2, X4, X6
03_U24	"0"	f=X1, X3, X5, X7

The $^{10}F^{9-}$ circuits in the function code translator have inputs from the circuits listed and provide a single-digit translation. For example, $^{10}F^{97}$ has inputs from $^{03}U^{26}$, $^{03}U^{25}$, and $^{03}U^{24}$. All three inputs are "0" only when the lower digit is seven thus, $^{10}F^{97}$ has a "1" output only when f=X7.

The translation of the $^{10}F^{9-}$ series can be determined by noting the right superscript. The $^{10}F^{97}$ inverter has a "1" output when f = X7. The $^{10}F^{95}$ inverter (not shown) has a "1" output when f = X5. Inverter $^{10}F^{93}$ (not shown) has a "1" output when f = X3, etc. The ^{21}F—series (only one circuit shown) provides the lower digit translation for two digits. For example, $^{21}F^{37}$ has inputs from $^{10}F^{97}$ ("1" when f = X7) and $^{10}F^{93}$ ("1" when f = X3). The output from $^{21}F^{37}$ has a "0" when either input is a "1"; thus $^{21}F^{37}$ has a "0" output when f = X3 or X7. The translation of the ^{21}F—series can be noted by examining the right superscript ($^{21}F^{37}$ has a "0" output when f =X3 or X7, $^{21}F26$ has a "0" output when f = X2 or X6, etc.). Most of the outputs from the lower digit translation are applied to circuits that combine them with the higher digit translation for a complete two-digit translation. The method of combining the lower and higher digits is explained below.

Higher Digit Translation

The higher digit translation of the two-digit function code is accomplished in much the same manner as the lower digit translation. The translation is made by the $^{10}F^{-9}$ series (fig. 10-16) that sense the three upper bits (29, 28, and 27) of the U-register. The following list shows the first translation of the higher digit.



124.132

Figure 10-16.—Function code translator (f = 13 or 17).

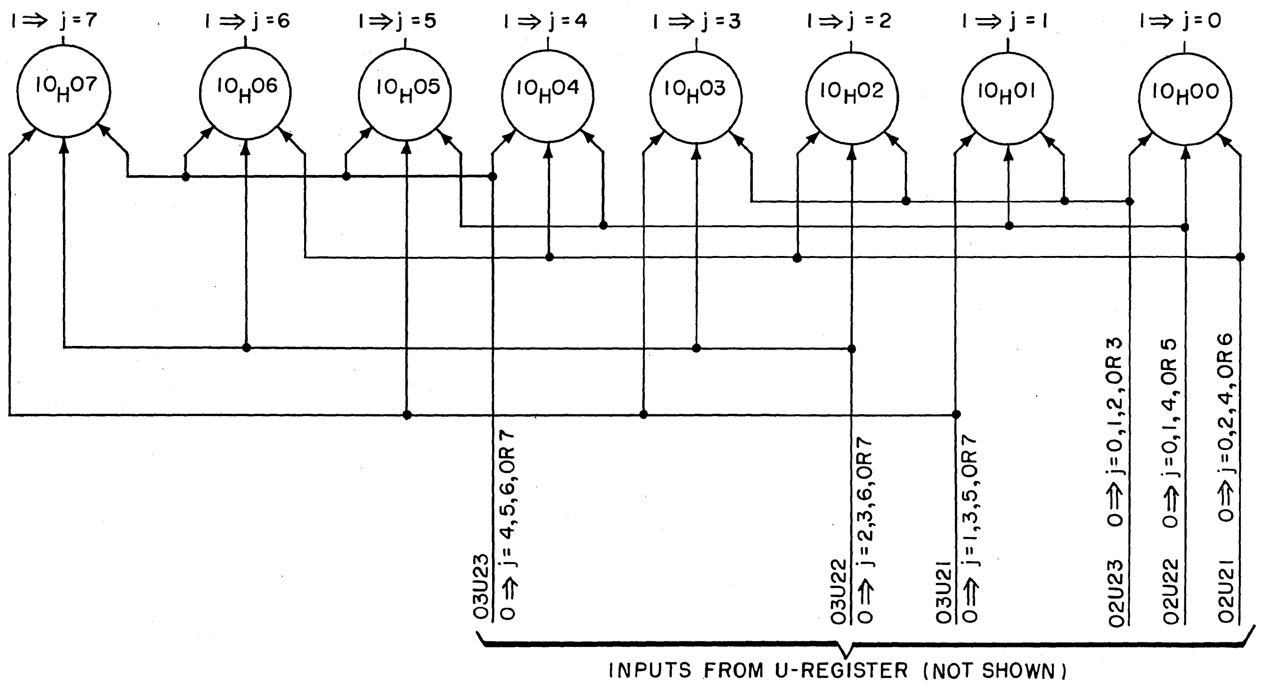
CIRCUIT	OUTPUT	CONDITIONS
02_U29	"0"	f=0X, 1X, 2X, 3X
03_U29 (not shown)	"0"	f=4X, 5X, 6X, 7X
02_U28	"0"	f=0X, 1X, 4X, 5X
03_U28 (not shown)	"0"	f=2X, 3X, 6X, 7X
02_U27 (not shown)	"0"	f=0X, 2X, 4X, 6X
03_U27	"0"	f=1X, 3X, 5X, 7X

The $^{10}F^{-9}$ series (only one circuit shown) have inputs from the inverter circuits listed and provide a single-digit translation. For example, 10_F19 has inputs from 02_U29, 02_U28, and 03_U27. All inputs are "0" only when the higher digit of the function code is one (f= 1X). The translation of the $^{10}F^{-9}$ series can be noted by examining the right superscript ($^{10}F^{19}$ has a "1" output when f= 1X, $^{10}F^{49}$ (not shown) has a "1" output when f= 4X, etc). The outputs resulting from the higher digit translation circuit and lower digit translation circuit are applied to $^{40}F^{13}$ which performs further translation by combining these inputs, thereby producing a complete function code translation.

BRANCH CONDITION DESIGNATOR j (3 BITS)

The branch condition designator is stored in bit positions 23, 22, and 21, of the U-register. This designator is used for all instructions, however, its interpretation varies depending on the instruction (see table 10-2 of instructions and data flow between major registers at the end of this chapter).

The translation of the j-designator 0 through 7 is accomplished by the $^{10}H^{0-}$ series (fig. 10-17) that sense U^{23}, U^{22}, and U^{21}. The outputs



124.133

Figure 10-17.—j-designator translation.

from the U-register to the j translator are listed as follows:

CIRCUIT	OUTPUT	CONDITIONS
02_U21	0	j = 0, 2, 4, 6
03_U21	0	j = 1, 3, 5, 7
02_U22	0	j = 0, 1, 4, 5
03_U22	0	j = 2, 3, 6, 7
02_U23	0	j = 0, 1, 2, 3
03_U23	0	j = 4, 5, 6, 7

The 10_H0- series senses these inputs and provides the proper translation. For example, 10_H03 has inputs from 02_U23, 03_U22, and 03_U21. The only time these inputs are 0 is when j = 3; thus, a one output from 10_H03 means j = 3. The translation of this series can be determined by noting the right superscript on all diagrams (10_H03 has a 1 output when j = 3; 10_H02 has a 1 output when j = 2; 10_H07 has a 1 output when j = 7, etc.).

In the instructions for which j gives the conditions for skipping the next sequential instruction, an evaluation is made at the end of the execution of the instruction to determine if the skip conditions specified by j have been satisfied. In the case of jump instructions an evaluation is made at the beginning of the instruction whether the jump condition specified by j is satisfied. The execution of the jump instruction is completed only when the jump conditions are satisfied.

OPERAND INTERPRETATION DESIGNATOR k (3 BITS)

The k-designator specifies the manner in which the operand is treated (from where the operand is read, and where it is stored). The k-designator is interpreted by translating the data in bit positions 20, 19 and 18 of the instruction word in the U-register. The translation circuits comprising $14H^{00}$ through $14H^{07}$ (fig. 10-18) for the k-designator are basically the same as described for the j-designator in figure 10-17. The inputs are received from the U^{20}, U^{19}, and U^{18} bit position flip-flops in the U-register (not shown).

The repertoire of instructions is divided into three classes of instructions: READ, STORE, and REPLACE. The k-designator is interpreted in a different manner for each class.

Read Instructions

A Read instruction is one which (during the B sequence) obtains an operand from memory or an operational register but does not store it after the basic operation is performed. This group of instructions includes: f = 01-13, 20-23 26-31, 40-43, 50-53, and 60-76. For these instructions, the value of k governs the transmission of the operand from $U_L \rightarrow X$, $Z \rightarrow X$, or $A \rightarrow X$ as necessary to execute the instruction. Certain I/O instructions (f = 13, 17, 62, 63, 66, 67 and 73-76) use a four-bit $\hat{j}$-designator. In these instructions, it is necessary to use a two-bit $\hat{k}$-designator (bit positions 19 and 18). This special $\hat{k}$-designator has the same interpretation as the normal $\hat{k}$-designator ($\hat{k}$ = 1 means the same as k = 1).

Store Instructions

In order to execute a STORE instruction, the destination of the information must be known before it can be stored. For these STORE instructions, the k-designator indicates to the computer where to store the information. The STORE instructions are: f = 14 through 17, 32, 33, and 47. (See table 10-2 at the end of this chapter.)

Replace Instructions

The third class of instructions are those that read and store (replace instructions). The REPLACE instruction are: f = 24, 25, 34 through 37, 44 through 46, and 54 through 57. For these instructions the k-designator specifies from where the operand is read and where the information is to be stored.

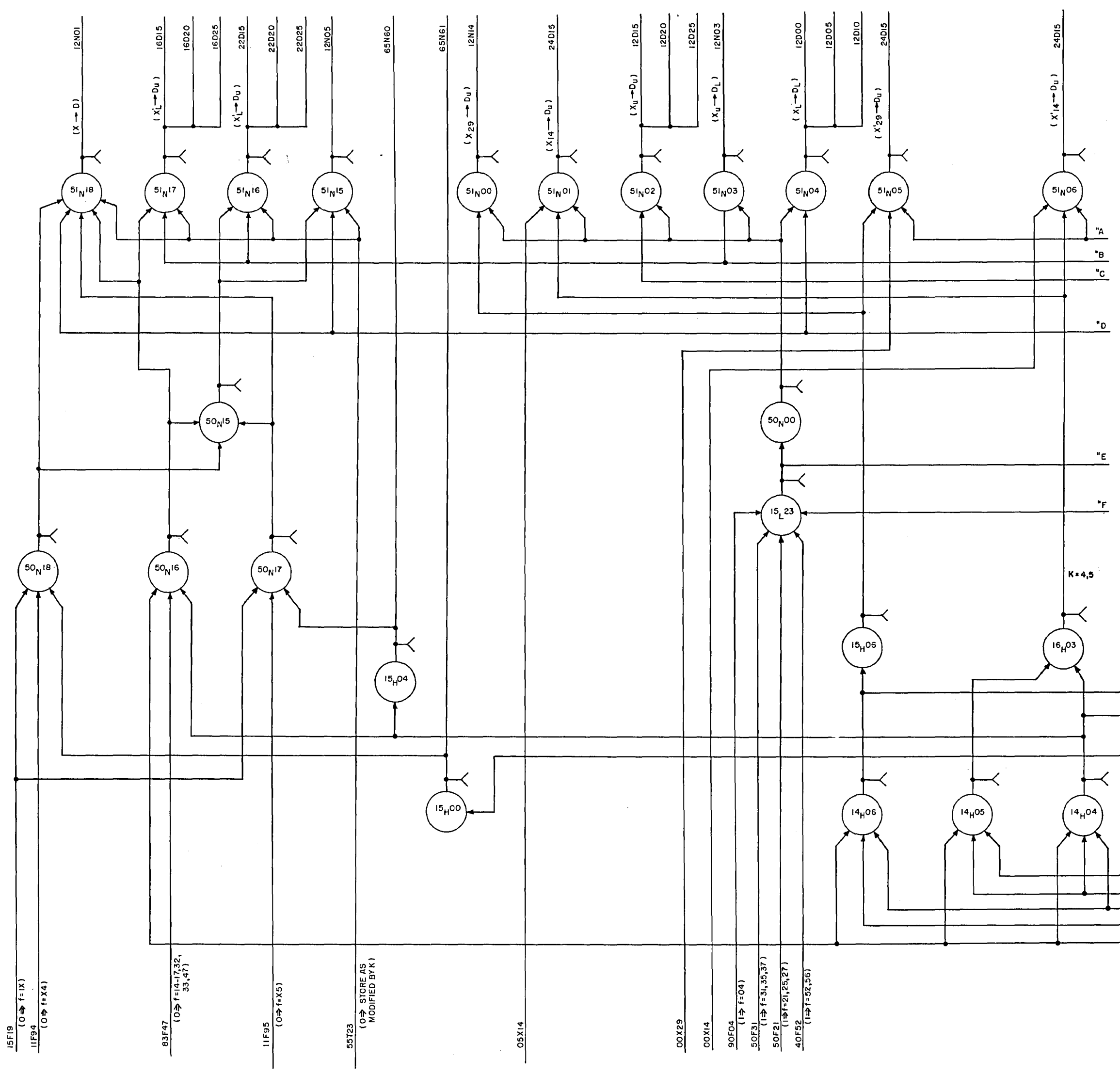
Read As Modified By k

The k-designator for READ instructions or the read portion of REPLACE instructions controls where the operand is read from and the manner in which it is transmitted. This sequence is normally initiated by the B sequence, but is also initiated by the C sequence when f = 53 or 57.

The C sequence initiation of "read as modified by k" sequence is necessary because of the arithmetic operations that must be performed. The read sequence is initiated (when f denotes a READ or REPLACE instruction) by setting L^{22} (fig. 10-18). When L^{22} is set enables are applied to $48N^{00}$ (clear X, not shown), $51N^{11}$, $51N^{12}$, and $51N^{13}$. The other inputs to these three circuits are from the k-translator, and it is here (in $51N^{11}$, $51N^{12}$, and $51N^{13}$) that the origin of the operand is determined. For example, $51N^{13}$ has a "0" input from $^{01}L^{22}$ and the input from $17H^{07}$ is a "0" when k = 1, 2, 3, 5, or 6. Thus, a "1" output from $51N^{13}$ means the operand must be read from memory and transmitted from $Z \rightarrow X$ (Z to X). There is a "1" output from $51N^{12}$ when k = 0, or 4, and it means that the operand must be read from U_L and transmitted from $R \rightarrow X$. Since U_L is only 15 bits, only the lower 15 bits of X contain data, and the upper 15 bits of X receive all "0's". There is a "1" output from $51N^{11}$ when k = 7 which means the operand must be read from the A-register.

OPERAND ADDRESS DESIGNATOR y (15 BITS)

This designator is composed of the lower 15 stages of the U register (U_L). Although its usual function is to specify the memory address of the operand, it can also be used for other functions. The conditions that specify the use of U_L are: (1) the instruction being performed, and (2) the value of the k designator. The Repertoire of Instructions has been divided into three groups, each utilizing k in a different manner. These groups are: (1) Read, (2) Store, and (3) Replace instructions. Normally, the modification of y consists of adding the contents of B_b to y. In the Repeat mode, the modification may take this form (r = 13 or 17), or it may consist of increasing or decreasing by one. The following is a summary of the functions of y for each group of instructions.

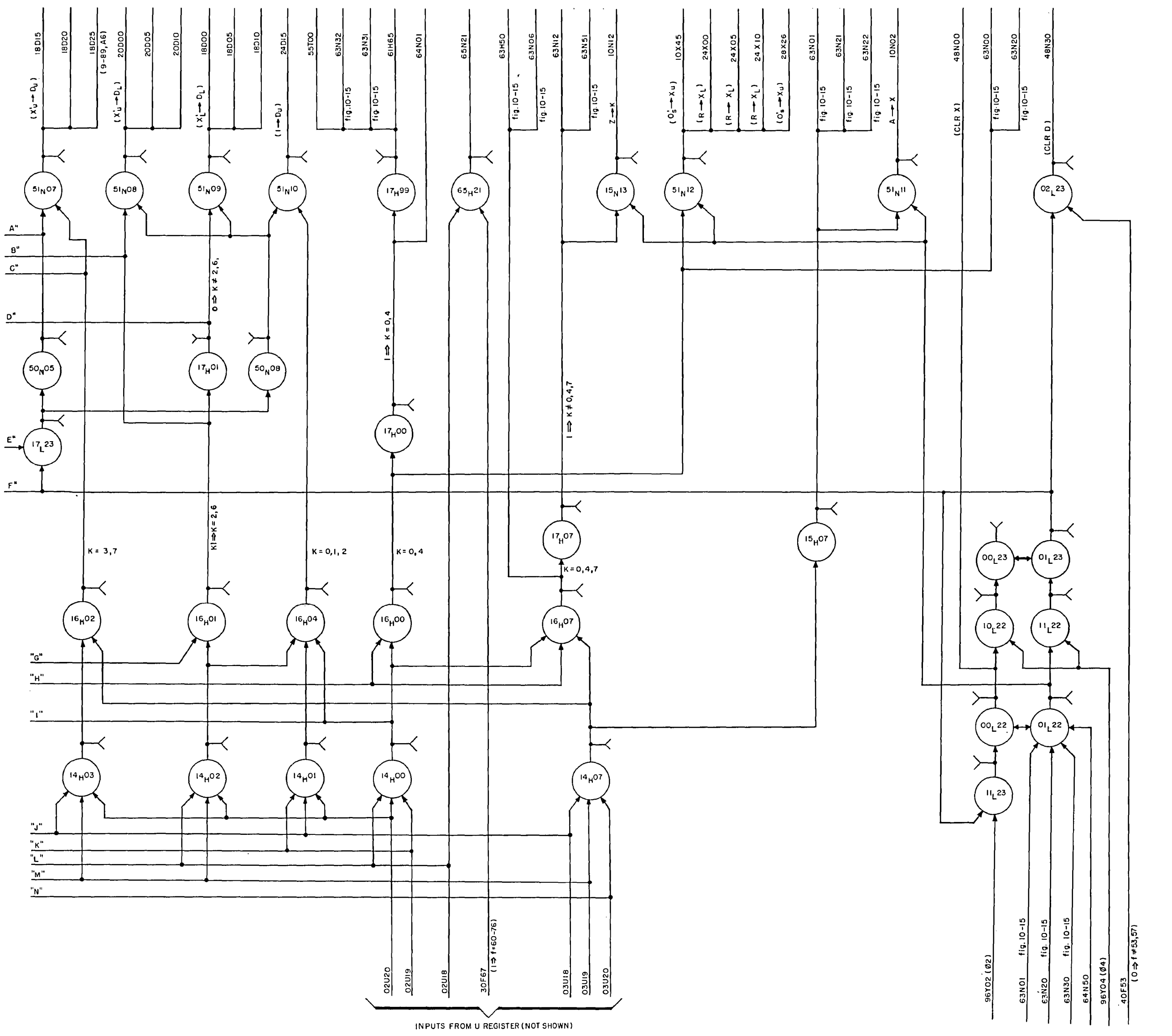


1. READ INSTRUCTIONS.—When k = 0 or 4, y is used directly as the 15-bit operand and no memory reference is made. When k = 7, the content of the A-register is used as the 30-bit operand. In this case y is not used and, again, no memory reference is made. When k = 1, 2, 3, 5, or 6, y specifies the memory location of the operand.

2. STORE INSTRUCTION.—When k = 0 or 4, y is not used, as the operand is stored directly in the Q register or the A register, respectively. When k = 1, 2, 3, 5, 6, or 7, y specifies the memory location for storage.

3. REPLACE INSTRUCTIONS.—When k = 1, 2, 3, 5, or 6; y specifies the memory location of the operand. After the operations specified by the instructions are completed, the result is returned to memory address y. For these instructions, the result is returned to the memory location from which the operand was read. The values k = 0, 4, and 7 are not used for Replace instructions.

On phase 4, L^{23} is set, which clears D (if $f \neq 53$ or 57) and enables circuits (discussed presently) which sense the transmission of the operand to D. There are two ways the operand can be transmitted to D; operand to D, or the complement of the operand to D. The complement of the operand is transmitted to D for the subtractive type Read instructions. These instructions are f = 04, 21, 25, 27, 31, 35, 52, and 56. These subtractive instructions are sensed by



124.134

Figure 10-18.—Read or store as modified by k and k translations.

15_L23 which has a "0" output if the present instruction is a subtractive instruction. The inputs to $15_L{}^{23}$ are as follows:

CIRCUIT	OUTPUT	CONDITIONS
01_L23	"0"	L^{23} is set (sequence initiated)
50_F21	"1"	f = 21, 25, 27
50_F31	"1"	f = 31, 35, 37
90_F04	"1"	f = 04
40_F52	"1"	f = 52 or 56

If the present instruction is a subtractive type, 17_L23 has both inputs "0" and the 1 output enables 51_N0- series to sense k. For example, if f = 04 and k = 3; 51_N09 has a "1" output that means $X'_L \rightarrow D_L$ and 51_N07 has a "1" output that means $X'_U \rightarrow D_U$. The final result of the read portion for f = 04 is: Read operand from memory$\rightarrow$Z and Z$\rightarrow$X; (the X register produces the complement of Z) $X'_U \rightarrow D_U$ and $X'_L \rightarrow D_L$. The result is the complement of the operand in the D-register. Comparison of the D-register content with other data can then be made.

For a nonsubtractive instruction (f $\neq$ subtractive instruction), all inputs to ${}^{15}L^{23}$ are "0", producing a "1" output that is inverted by ${}^{50}N^{00}$ and enables the ${}^{51}N^{04}$ - ${}^{51}N^{00}$ circuits to sense k. For example, f = 22 and k = 3; ${}^{51}N^{04}$ has a "1" that means $X_L \rightarrow D_L$. Circuit ${}^{51}N^{02}$ has a "1" output that means $X_U \rightarrow D_U$ and the result is the operand entered in the D register. In some cases, such as k = 6 X_U is transmitted to D_L and the sign bit of X (X_{29}) is extended to the upper 15 bits of D (D_U). However, for the subtractive instructions, the complement of the sign bit (X'_{29}) would be entered in D_U. After the operand has been entered in the D register, the computer proceeds with the operation according to the instruction and other designators. Now consider the store as modified by k operation. There are four ways data can be entered into D (prior to storage) as determined by k. They are: 1) the 30-bit word to D, 2) the 15-bit word to D, 3) the complement of the 30-bit word to D, and 4) the complement of the 15-bit word to D. These interpretations are provided by the ${}^{51}N$—circuits (fig. 10-18). For example, ${}^{51}N^{18}$ has inputs from: ${}^{55}T^{23}$ -"0" from D sequence when f = Store, Replace, 73 through 76; and ${}^{50}N^{16}$ - "0" if f $\neq$ Store, k = 0, 1, 2, 3, 4. These two inputs have limited the output to mean a Replace instruction or k =0, 1, 2, 3, 4. The remaining inputs further condition the value of k as follows:

${}^{50}N^{18}$ - "0" if k $\neq$ 0
${}^{50}N^{17}$ - "0" if k $\neq$ 4
${}^{17}H^{01}$ - "0" if k $\neq$ 2, 6

Thus, the output from ${}^{51}N^{18}$ will be a "1" if f = Replace, and k = 1 or 3. The other translations of the ${}^{51}N$—circuits (shown in figure 10-18) are:

${}^{51}N^{17}$ - "1" if f = Replace and k = 2
${}^{51}N^{16}$ - "1" if f = Store and k = 6
${}^{51}N^{15}$ - "1" if f = Store and k - 5 or 7

These translations control the manner in which the information is transmitted. The storage location is determined in the D sequence. For example, ${}^{64}N^{60}$ initiates the storage of information in the A-register if f = Store and k = 4; ${}^{65}N^{51}$ stores the information in Q in f = Store and k = 0.

ADDRESS MODIFICATION DESIGNATOR b (3 BITS)

The b designator specifies which B register that is used to modify the operand address designator, y, before an instruction is read out of memory. The b-designator is stored in U_u, stages 15, 16, and 17.

TRANSLATION.—The b translation is done by the ${}^{13}H^{0-}$ circuits not shown. The translation circuits are similar to those shown for the j designator translation of figure 10-17.

P-REGISTER

The P-register (see fig. 10-1) is a 15-bit nonaddressable register. It is commonly called the Program Address Register because it holds the address of the instruction being executed. The inputs to the P-register are from either the S-register or the R-register. The transmission to P (after clearing of P) is gated in on clock phase 3 using ${}^{15}P$—(R$\rightarrow$P) or ${}^{17}P$—(S$\rightarrow$P).

REGISTER OPERATION

The command enable, S$\rightarrow$P, can be generated in the A or D sequence depending upon certain conditions. During a normal series of instructions, the A sequence initiates Clear P, S$\rightarrow$P, and set p to plus one. This is done in anticipation of reading the next sequential instruction. (When the present instruction has been completed and the A sequence initiated for the next instruction, one of the first commands is P-adder$\rightarrow$S, placing in S the address of the next instruction.) The other origin of the S$\rightarrow$P command is the D sequence.

The other transmission to P is from the R-register, initiated by the C sequence. This is done to accomplish a jump to address Y. Previously, U_L was placed in R and now R$\rightarrow$P occurs; at the same time, Set p to plus zero occurs. During the next sequence, the command, P-adder$\rightarrow$S, places Y in the S-register and the jump to Y is accomplished.

The outputs from the P-register are applied to the P-adder and to the associated indicator driver circuits (not shown).

P-ADDER

The P-adder is a 15-bit, additive, open-ended (no end-around carry) type adder. Its purpose is to modify the contents of P by the addition of plus one, plus two, or it can add zero to (P). Zero is added to (P) (during the A sequence) when DISCONNECT P or disable the P-register is requested. This causes a given instruction to be repeated until the disconnect is released.

The contents of the P-register (P) is advanced by plus one during the normal execution of instructions stored at consecutive addresses. The P-register content (P) is advanced by plus two when the instruction following a Repeat instruction (f = 70) is programmed with j $\neq$ 0, 1, (see discussion of designators in chapter 8) and the specified skip condition is satisfied.

p-DESIGNATOR

The modification of the P-register is determined by the flip-flops. The contents of the P-register are incremented by plus two plus one, or zero, depending on the state of these flip-flops.

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS.

CODE	FUNCTION	CODE	FUNCTION
00	(Fault Interrupt)	42	SUBTRACT LOGICAL PRODUCT
01	RIGHT SHIFT Q	43	COMPARE MASKED
02	RIGHT SHIFT A	44	REPLACE LOGICAL PRODUCT
03	RIGHT SHIFT AQ	45	REPLACE A+ LOGICAL PRODUCT
04	COMPARE A, Q, AQ	46	REPLACE A- LOGICAL PRODUCT
05	LEFT SHIFT Q	47	STORE LOGICAL PRODUCT
06	LEFT SHIFT A		
07	LEFT SHIFT AQ		
		50	SELECTIVE SET
10	ENTER Q	51	SELECTIVE COMPLEMENT
11	ENTER A	52	SELECTIVE CLEAR
12	ENTER Bj	53	SELECTIVE SUBSTITUTE
13	EXTERNAL FUNCTION ON Cĵ	54	REPLACE SELECTIVE SET
14	STORE Q	55	REPLACE SELECTIVE COMPLEMENT
15	STORE A	56	REPLACE SELECTIVE CLEAR
16	STORE Bj	57	REPLACE SELECTIVE SUBSTITUTE
17	STORE Cĵ		
		60	JUMP (Arithmetic)
20	ADD A	61	JUMP (Manual)
21	SUBTRACT A	62	JUMP ON Cĵ ACTIVE INPUT BUFFER
22	MULTIPLY	63	JUMP ON Cĵ ACTIVE OUTPUT BUFFER
23	DIVIDE	64	RETURN JUMP (Arithmetic)
24	REPLACE A+Y	65	RETURN JUMP (Manual)
25	REPLACE A-Y	66	TERMINATE Cĵ INPUT BUFFER
26	ADD Q	67	TERMINATE Cĵ OUTPUT BUFFER
27	SUBTRACT Q		
		70	REPEAT
30	ENTER Y+Q	71	B SKIP ON Bĵ
31	ENTER Y-Q	72	B JUMP ON Bĵ
32	STORE A+Q	73	INPUT BUFFER ON Cĵ (without Monitor mode)
33	STORE A-Q		
34	REPLACE Y+Q	74	OUTPUT BUFFER ON Cĵ (without Monitor mode)
35	REPLACE Y-Q		
36	REPLACE Y+1	75	INPUT BUFFER Cĵ (with Monitor mode)
37	REPLACE Y-1		
		76	OUTPUT BUFFER ON Cĵ (with Monitor mode)
40	ENTER LOGICAL PRODUCT		
41	ADD LOGICAL PRODUCT	77	(Fault Interrupt)

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

01 RIGHT SHIFT Q [SHIFT (Q) RIGHT BY Y]

This instruction shifts (Q) to the right Y bit positions. The higher-order bits are replaced with the original sign bit as the word is shifted. Only the lower-order six bits of Y are recognized for this instruction. The higher-order bits are ignored.

B SEQUENCE	C SEQUENCE
Clear D	$k \neq 0, 4$; Clear R, $D \rightarrow R$
$k \neq 0, 4$; Shift Count $\rightarrow D$	Initiate Shift Sequence
$k = 0, 4$; U Adder $\rightarrow D$	

02 RIGHT SHIFT A [SHIFT (A) RIGHT BY Y]

This instruction shifts (A) to the right Y bit positions. The higher-order bits are replaced with the original sign bit as the word is shifted. Only the lower-order six bits of Y are recognized for this instruction. The higher-order bits are ignored.

B SEQUENCE	C SEQUENCE
Clear D	$k \neq 0, 4$; Clear R, $D \rightarrow R$
$k \neq 0, 4$; Shift Count $\rightarrow D$	Initiate Shift Sequence
$k = 0, 4$; U Adder $\rightarrow D$	

03 RIGHT SHIFT AQ [SHIFT (AQ) RIGHT BY Y]

This instruction shifts (A) and (Q) as one 60-bit register. The shift is to the right Y bit positions, with the lower bits of (A) shifting into the higher bit positions of (Q). The higher-order bits of (A) are replaced with the original sign bit as the word is shifted. Only the lower-order six bits of Y are recognized for this instruction. The higher-order bits are ignored.

B SEQUENCE	C SEQUENCE
Clear D	$k \neq 0, 4$; Clear R, $D \rightarrow R$
$k \neq 0, 4$; Shift Count $\rightarrow D$	Initiate Shift Sequence
$k = 0, 4$; U Adder $\rightarrow D$	

04 COMPARE: SENSE j: $[(A)_i = (A)_f]$

This instruction compares the signed value of Y with the signed value of (A) and/or (Q), but does not alter either (A) or (Q). Branch condition designator j is interpreted in a special way for this instruction, as follows:

$j = 0$:	Do not skip the next instruction.
$j = 1$:	Skip the next instruction.
$j = 2$:	Skip the next instruction if Y is less than or equal to (Q). $(Y \leq Q)$
$j = 3$:	Skip the next instruction if Y is greater than (Q). $(Y > Q)$.
$j = 4$:	Skip the next instruction if (Q) is greater than or equal to Y and Y is greater than (A). $(Q \geq Y$ and $Y > A)$

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

	j = 5: Skip the next instruction if Y is greater than (Q) or if Y is less than or equal to (A). $(Y > Q \text{ or } Y \leq A)$ j = 6: Skip the next instruction if Y is less than or equal to (A). $(Y \leq A)$ j = 7: Skip the next instruction if Y is greater than (A). $(Y > A)$	**B SEQUENCE** Operand$\rightarrow$D	**C SEQUENCE** Sense j (for A) Clear X, A$\rightarrow$X Clear A, Q$\rightarrow$A Sense j (for Q) Clear A, X$\rightarrow$A
05	LEFT SHIFT Q [SHIFT (Q) LEFT BY Y] This instruction shifts (Q) circularly to the left Y bit positions.* The lower-order bits are replaced with the higher-order bits as the word is shifted. Only the lower-order six bits of Y are recognized for this instruction. The higher-order bits are ignored.	**B SEQUENCE** Clear D $k \neq 0, 4$; Shift Count$\rightarrow$D $k = 0, 4$; U Adder$\rightarrow$D	**C SEQUENCE** $k \neq 0, 4$; Clear R, D$\rightarrow$R Initiate Shift Sequence
06	LEFT SHIFT A [SHIFT (A) LEFT BY Y] This instruction shifts (A) circularly to the left Y bit positions.* The lower-order bits are replaced with the higher-order bits as the word is shifted. Only the lower-order six bits of Y are recognized for this instruction. The higher-order bits are ignored.	**B SEQUENCE** Clear D $k \neq 0, 4$; Shift Count$\rightarrow$D $k = 0, 4$; U Adder$\rightarrow$D	**C SEQUENCE** $k \neq 0, 4$; Clear R, D$\rightarrow$R Initiate Shift Sequence
07	LEFT SHIFT AQ [SHIFT (AQ) LEFT BY Y] This instruction shifts (A) and (Q) as one 60-bit register. The shift is circular to the left Y bit positions.* The lower bits of (A) are replaced with the higher bits of (Q) and the lower bits of (Q) are replaced with the higher bits of (A). Only the lower-order six bits of Y are recognized by this instruction. The higher-order bits are ignored.	**B SEQUENCE** Clear D $k \neq 0, 4$; Shift Count$\rightarrow$D $k = 0, 4$; U Adder$\rightarrow$D	**C SEQUENCE** $k \neq 0, 4$; Clear R, D$\rightarrow$R Initiate Shift Sequence

*Maximum shift permitted is 59 places.

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

10 ENTER Q $[Y \rightarrow Q]$

Clear the Q register, then transmit Y to Q.

B SEQUENCE	C SEQUENCE
Operand $\rightarrow$ D	Clear X, D $\rightarrow$ X
	Clear Q, X $\rightarrow$ Q
	Sense j

11 ENTER A $[Y \rightarrow A]$

Clear A, then transmit Y to A.

B SEQUENCE	C SEQUENCE
Operand $\rightarrow$ D	Clear X, D $\rightarrow$ X
	Clear A, X $\rightarrow$ A
	Sense j

12 ENTER B^j $[Y \rightarrow B_j]$

Clear the contents of B register j. Then transmit 15 bits of Y to B register j. Branch condition designator j is used to specify the selected B register for this instruction and is not available for its normal function.

B SEQUENCE	C SEQUENCE
Operand $\rightarrow$ D	k $\neq$ 0, 4; Clear R, D $\rightarrow$ R
	Clear B_j, R $\rightarrow$ B_j

13 EXTERNAL FUNCTION ON $C^{\hat{j}}$

$\hat{j}$ = 0 or 1. Interrogate the two bits connected to the input active designator (flip-flops) on an interconnected computer. If the interconnected computer has input buffer active, skip the next instruction. If the interconnected computer has input buffer not active, execute the next instruction. There are no external function lines on C_0 or C_1. When $\hat{j} \neq$ 0 or 1, transmit Y, the external function, over the channel specified by $\hat{j}$. Only $\hat{k}$ = 3 permitted.

B SEQUENCE	D SEQUENCE	
	M $\rightarrow$ Z $\rightarrow$ C_0; Set External Function line	$\hat{j} \neq$ 0, 1
	Skip if channel 0 or 1 on other computer has input buffer active; Set p to +2	$\hat{j}$ = 0, 1

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

14 STORE Q $[(Q) \rightarrow \underline{Y}]$

Store (Q) at storage address $\underline{Y}$ as directed by operand interpretation designator k. If k = 0, complement (Q). If k = 4, store in A.

B SEQUENCE	C SEQUENCE	D SEQUENCE
	Clear X, Q $\rightarrow$ X	Store as modified by k
		Sense j

15 STORE A $[(A) \rightarrow \underline{Y}]$

Store (A) at storage address $\underline{Y}$ as directed by operand interpretation designator k. If k = 4, complement (A). If k = 0, store in Q.

B SEQUENCE	C SEQUENCE	D SEQUENCE
	Clear X, A $\rightarrow$ X	Store as modified by k
		Sense j

16 STORE B^j $[(B)_j \rightarrow \underline{Y}]$

Store a 30-bit quantity whose lower-order 15 bits correspond to the content of the B register j, and whose higher-order 15 bits are zero at storage address $\underline{Y}$, as directed by operand interpretation designator k. Branch condition designator j, is used to specify the selected B register for this instruction and is not available for its normal function.

B SEQUENCE	C SEQUENCE	D SEQUENCE
		Clear R*, $B_j \rightarrow$ R*
		Clear X, R* $\rightarrow X_L$
		0's $\rightarrow X_U$
		Store as modified by k

17 STORE $C^{\hat{j}}$ $[(C)_{\hat{j}} \rightarrow \underline{Y}]$

Store the content of the channel specified by $\hat{j}$ at storage address $\underline{Y}$. An input acknowledge signal is then sent on the channel. Only k = 3 permitted.

Note

Instruction 17, Store $C^{\hat{j}}$, is intended for use in the computer's reply to an interrupt; consequently it is not synchronized with the input buffering process. Therefore, the execution of $\underline{n}$ sequential 17 instructions on the same channel will not place $\underline{n}$ sequential Input Acknowledge signals on the Input Acknowledge line associated with that channel. It will generate a signal $\underline{n}$ times 14.8 microseconds wide on that input acknowledge line. Moreover, the execution of f = 17 on a given channel, while an input buffer is in progress on that channel will seriously interfere with the buffered transfer of data in most cases. It should be noted that any other instruction executed between two 17's will allow the Input Acknowledge line to return to the logical zero state for a time consistent with input/output specifications before it rises a second time.

B SEQUENCE	C SEQUENCE	D SEQUENCE
		$(C)\hat{j} \rightarrow Z \rightarrow M$
		Send Input Acknowledge

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

20 ADD A $[(A)+Y \rightarrow A]$

Add Y to the previous content of the A register. D receives the exact number to be added to A.

B SEQUENCE	C SEQUENCE
Operand $\rightarrow$ D	Clear X, Adder $\rightarrow$ X
	Sense j from Adder
	Clear A, X $\rightarrow$ A

21 SUBTRACT A $[(A)-Y \rightarrow A]$

Subtract Y from the previous content of the A register. D receives the complement of the number to be subtracted from A.

B SEQUENCE	C SEQUENCE
Operand' $\rightarrow$ D	Clear X, Adder $\rightarrow$ X
	Sense j from Adder
	Clear A, X $\rightarrow$ A

22 MULTIPLY $[(A)Y \rightarrow AQ]$

Multiply (Q) times Y forming the double-length product in AQ. If the factors are considered as integers, the product is an integer in AQ. Branch condition designator j is interpreted prior to final sign correction, permitting double-length product detection. When (A) $\neq$ 0, a double-length product has been formed with significant bit(s) in the A register. Also, if Q is negative, there is a double-length product. The Multiply instruction can be executed with a j = 2 followed by an instruction that checks the value of (A). k = 7 should not be used in this instruction.

B SEQUENCE	C SEQUENCE
If Q is negative:	If D is negative:
Set Complement 1 flip-flop and initiate Complement AQ sequence	Set Complement 2 flip-flop Clear X, D $\rightarrow$ X Clear D, X' $\rightarrow$ D
Operand $\rightarrow$ D	If D is positive:
	Clear X, D $\rightarrow$ X
	Clear D, X $\rightarrow$ D
	Clear K, Clear A;
	Set K15; set K60 if Q00 through Q04 = 0;
	Initiate Multiply

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

23 DIVIDE $[(AQ/Y \longrightarrow Q]$

Divide (AQ) by Y, leaving the quotient in Q register and the remainder in the A register. The remainder bears the same sign as the quotient. k = 7 should not be used in this instruction.

Note

If a divide fault condition exists, no console indication is given. However, by coding each Divide instruction with j = 3, a program test for the Divide fault is automatic. With this selection of j, a skip of the next instruction occurs if a divide fault exists. The skip should be made to a Jump instruction which provides the remedial means of noting the error or correcting it. Therefore, the instruction that follows the Divide instruction should have its j = 1 in order to skip the Jump instruction whenever the Divide sequence culminates in a correct answer. A Divide fault can be detected also if the Divide instruction is executed with j = 2. In this case, a correct answer is indicated when a skip occurs.

B SEQUENCE	C SEQUENCE
If A is negative:	If D is negative:
Set Complement 1 flip-flop and initiate Complement AQ sequence	Set Complement 2 flip-flop Clear X, D $\longrightarrow$ X Clear D, X $\longrightarrow$ D
Operand $\longrightarrow$ D	If D is positive:
	Clear X, D $\longrightarrow$ X
	Clear D, X' $\longrightarrow$ D
	Clear K, Set K15
	Initiate Divide

24 REPLACE A + $\underline{Y}$ $(A) + \underline{Y} \longrightarrow \underline{Y}$ & A

Add $\underline{Y}$ to the previous content of A. Store $(A)_f$ at storage address $\underline{Y}$.

B SEQUENCE	C SEQUENCE	D SEQUENCE
Operand $\longrightarrow$ D	Clear X, Adder $\longrightarrow$ X	Store as modified by k
	Sense j from Adder	
	Clear A, X $\longrightarrow$ A	

25 REPLACE A - Y $(A) - \underline{Y} \longrightarrow \underline{Y}$ & A

Subtract Y from the previous content of A. Then store $(A)_f$ at storage address $\underline{Y}$.

B SEQUENCE	C SEQUENCE	D SEQUENCE
Operand' $\longrightarrow$ D	Clear X, Adder $\longrightarrow$ X	Store as modified by k
	Sense j from Adder	
	Clear A, X $\longrightarrow$ A	

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

26 ADD Q $(Q) + Y \longrightarrow Q$

Interchange (A) and (Q). Then add Y to (A). Interchange (A) and (Q). The content of A is undisturbed by this instruction. Branch condition designator j has special meaning in this instruction, as listed in the 27 instruction.

B SEQUENCE	C SEQUENCE
Operand $\longrightarrow$ D	Clear X, A $\longrightarrow$ X
	Clear A, Q $\longrightarrow$ A
	Clear Q, X $\longrightarrow$ Q
	Clear X, Adder $\longrightarrow$ X
	Sense j from Adder
	Clear A, Q $\longrightarrow$ A
	Clear Q, X $\longrightarrow$ Q

27 SUBTRACT Q $[(A) - Y \longrightarrow Q]$

Interchange (A) and (Q). Then subtract Y from (A). Interchange (A) and (Q). The content of A is undisturbed by this instruction. Branch condition designator j has special meaning in this instruction, as listed below.

Note

In instructions 26 and 27 the branch condition designator j has the following meaning:

j = 0:	Do not skip the next instruction.
j = 1:	Skip the next instruction.
j = 2:	Skip the next instruction if (A) is positive.
j = 3:	Skip the next instruction if (A) is negative.
j = 4:	Skip the next instruction if (Q) is zero.
j = 5:	Skip the next instruction if (Q) is nonzero.
j = 6:	Skip the next instruction if (Q) is positive.
j = 7:	Skip the next instruction if (Q) is negative.

B SEQUENCE	C SEQUENCE
Operand' $\longrightarrow$ D	Clear X, A $\longrightarrow$ X
	Clear A, Q $\longrightarrow$ A
	Clear Q, X $\longrightarrow$ Q
	Clear X, Adder $\longrightarrow$ X
	Sense j from Adder
	Clear A, Q $\longrightarrow$ A
	Clear Q, X $\longrightarrow$ Q

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

30	ENTER Y + Q	$[Y + (Q) \longrightarrow A]$	
	Clear A. Then transmit (Q) to (A). Then add Y to (A).		
	B SEQUENCE	**C SEQUENCE**	
	Clear A, Q $\longrightarrow$ A	Clear X, Adder $\longrightarrow$ X	
	Operand $\longrightarrow$ D	Clear A, X $\longrightarrow$ A	
		Sense j	
31	ENTER Y - Q	$[Y - (Q) \longrightarrow A]$	
	Clear A. Then transmit (Q) to (A). Then subtract Y from (A). Finally, complement (A) if Y - (Q) $\neq$ +0.		
	B SEQUENCE	**C SEQUENCE**	
	Clear A, Q $\longrightarrow$ A	Clear X, Adder $\longrightarrow$ X	
	Operand' $\longrightarrow$ D	Clear D, X' $\longrightarrow$ D	if A $\neq$ D
		Clear X, D $\longrightarrow$ X	if A $\neq$ D
		Clear A, X $\longrightarrow$ A	
		Sense j	
32	STORE A + Q	$[(A) + (Q) \longrightarrow \underline{Y}$ & $A]$	
	Add (Q) to the previous content of A. Then store $(A)_f$ at storage address $\underline{Y}$ as directed by the operand interpretation designator, k.		
	B SEQUENCE	**C SEQUENCE**	**D SEQUENCE**
		Clear X, Q $\longrightarrow$ X	Store as modified by k
		Clear D, X $\longrightarrow$ D	Sense j
		Clear X, Adder $\longrightarrow$ X	
		Clear A, X $\longrightarrow$ A	
33	STORE A - Q	$[(A) - (Q) \longrightarrow \underline{Y}$ & $A]$	
	Subtract (Q) from the previous content of A register. Then store $(A)_f$ at storage address $\underline{Y}$ as directed by the operand interpretation designator k.		
	B SEQUENCE	**C SEQUENCE**	**D SEQUENCE**
		Clear X, Q $\longrightarrow$ X	Store as modified by k
		Clear D, X' $\longrightarrow$ D	Sense j
		Clear X, Adder $\longrightarrow$ X	
		Clear A, X $\longrightarrow$ A	

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

34 REPLACE Y + Q $[Y + (Q) \rightarrow \underline{Y} \ \& \ A]$

Clear A. Then transmit (Q) to (A). Then add Y to (A). Then store $(A)_f$ at storage address $\underline{Y}$.

B SEQUENCE	C SEQUENCE	D SEQUENCE
Operand $\rightarrow$ D	Clear X, Adder $\rightarrow$ X	Store as modified by k
Clear A, Q $\rightarrow$ A	Clear A, X $\rightarrow$ A	
	Sense j	

35 REPLACE Y - Q $[Y - Q \rightarrow \underline{Y} \ \& \ A]$

Clear A. Then transmit (Q) to (A). Then subtract Y from (A). Then complement (A) if Y - (Q) = +0 and store at storage address $\underline{Y}$.

B SEQUENCE	C SEQUENCE	D SEQUENCE
Clear A, Q $\rightarrow$ A	Clear X, Adder $\rightarrow$ X	Store as modified by k
Clear A, Q $\rightarrow$ A	If A $\neq$ D'	
Operand' $\rightarrow$ D	Clear D, X' $\rightarrow$ D	
	Clear X, D $\rightarrow$ X	
	Clear A, X $\rightarrow$ A	
	Sense j	

36 REPLACE Y + 1 $[Y + 1 \rightarrow \underline{Y} \ \& \ A]$

Clear A. Then set (A) = +1. Then add Y to (A). Then store $(A)_f$ at storage address $\underline{Y}$.

B SEQUENCE	C SEQUENCE	D SEQUENCE
Clear A, +1 $\rightarrow$ A	Clear X, Adder $\rightarrow$ X	Store as modified by k
Operand $\rightarrow$ D	Clear A, X $\rightarrow$ A	
	Sense j	

37 REPLACE Y - 1 $[Y - 1 \rightarrow \underline{Y} \ \& \ A]$

Clear A. Then set (A) = 1. Then subtract Y from (A). Then complement (A) if (Y) - 1 $\neq$ +0 and store at storage address $\underline{Y}$.

B SEQUENCE	C SEQUENCE	D SEQUENCE
Clear A, +1 $\rightarrow$ A	Clear X, Adder $\rightarrow$ X	Store as modified by k
Operand' $\rightarrow$ D	If A $\neq$ D	
	Clear D, X' $\rightarrow$ D	
	Clear X, D $\rightarrow$ X	
	Clear A, X $\rightarrow$ A	
	Sense j	

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

40 ENTER LOGICAL PRODUCT $[L(Y)(Q) \longrightarrow A]$

Enter in A the bit-by-bit product of Y and (Q). The j designator is interpreted in a special way for this instruction for the values j = 2 or 3. If j = 2, skip if the parity of $(A)_f$ is even. If j = 3, skip if the parity of $(A)_f$ is odd.

Note

Even parity: An even number of ones in the A register.
Odd parity: An odd number of ones in the A register.

B SEQUENCE	C SEQUENCE
Operand $\longrightarrow$ D	Clear X, Q & D $\longrightarrow$ X
	(L(D)(Q) $\longrightarrow$ X)
	Clear A, X $\longrightarrow$ A
	If j $\neq$ 2, 3; Sense j
	Clear D, $X_U \longrightarrow D_L$
	Inhibit sense j (for above)
If j = 2, 3	Inhibit evaluation of skip
	if Q positive or Q negative
	Sense j from Adder.

41 ADD LOGICAL PRODUCT $[L(Y)(Q) + A \longrightarrow A]$

Add to (A) the bit-by-bit product of Y and (Q).

B SEQUENCE	C SEQUENCE
Operand $\longrightarrow$ D	Clear X, Q & D $\longrightarrow$ X,
	(L(D)(Q) $\longrightarrow$ X)
	Clear D, X $\longrightarrow$ D
	Clear X, Adder $\longrightarrow$ X
	Sense j from Adder
	Clear A, X $\longrightarrow$ A

42 SUBTRACT LOGICAL PRODUCT $[A - L(Y)(Q) \longrightarrow A]$

Subtract from (A) the bit-by-bit product of Y and (Q).

B SEQUENCE	C SEQUENCE
Operand $\longrightarrow$ D	Clear X, Q & D $\longrightarrow$ X
	(L(D)(Q) $\longrightarrow$ X)
	Clear D, X' $\longrightarrow$ D
	Clear X, Adder $\longrightarrow$ X
	Sense j from Adder
	Clear A, X $\longrightarrow$ A

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

43 COMPARE MASKED [A - L(Y)(Q); Sense j]

Subtract from (A) the bit-by-bit product of Y and (Q) and perform the branch point evaluation for skip of next sequential instruction as directed by branch condition designator j. This instruction results in no net change in the content of any operational register. It provides a comparison of a portion of Y with (A) through branch condition designator j.

B SEQUENCE	C SEQUENCE
Operand $\longrightarrow$ D	Clear X, Q & D $\longrightarrow$ X
	(L(D)(Q) $\longrightarrow$ X)
	Clear D, X' $\longrightarrow$ D
	Sense j from Adder

(A):	Contains the value that is the basis for the comparison.
(Q):	Determines which part of the operand (any/all) is to be compared to (A).
(D):	Contains the value the comparison is to be made upon; if $(A)_f = 0$, the value has been found (use j = 4).

44 REPLACE LOGICAL PRODUCT [L(Y)(Q) $\longrightarrow$ $\underline{Y}$ & A]

Enter in A the bit-by-bit product of Y and (Q). Then store (A) at storage address $\underline{Y}$. The j designator is interpreted in a special way for this instruction for the value j = 2, 3. If j = 2, skip if the parity of $(A)_f$ is even. If j = 3, skip if the parity of $(A)_f$ is odd.

B SEQUENCE	C SEQUENCE	D SEQUENCE
Operand $\longrightarrow$ D	Clear X, Q & D $\longrightarrow$ X	Store as modified by k
	(L(D)(Q) $\longrightarrow$ X)	
	Clear A, X $\longrightarrow$ A	
	If j $\neq$ 2, 3;	
	Sense j	
	If j = 2, 3;	
	Clear D, $X_U \longrightarrow D_L$	
	Inhibit sense j	
	(for above)	
	Inhibit evaluation	
	of "skip if Q	
	positive or Q	
	negative"	
	Sense j from Adder	

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

45 REPLACE A + LOGICAL PRODUCT [A + L(Y)(Q)$\longrightarrow\underline{Y}$ & A]

Add to (A) the bit-by-bit product of Y and (A). Then store (A) at storage address Y.

B SEQUENCE	C SEQUENCE	D SEQUENCE
Operand$\longrightarrow$D	Clear X, Q & D$\longrightarrow$X	Store as modified by k
	(L(D)(Q)$\longrightarrow$X)	
	Clear D, X$\longrightarrow$D	
	Clear X, Adder$\longrightarrow$X	
	Clear A, X$\longrightarrow$A	
	Sense j	

46 REPLACE A - LOGICAL PRODUCT [A - L(Y)(Q)$\longrightarrow\underline{Y}$ & A]

Subtract from (A) the bit-by-bit product of Y and (Q). Then store (A) at storage address $\underline{Y}$.

B SEQUENCE	C SEQUENCE	D SEQUENCE
Operand$\longrightarrow$D	Clear X, Q & D$\longrightarrow$X	Store as modified by k
	(L(D)(Q)$\longrightarrow$X)	
	Clear D, X'$\longrightarrow$D	
	Clear X, Adder$\longrightarrow$X	
	Clear A, X$\longrightarrow$A	
	Sense j	

47 STORE LOGICAL PRODUCT [L(A)(Q)$\longrightarrow\underline{Y}$]

Store in address $\underline{Y}$ the bit-by-bit product of (A) and (Q) as directed by operand interpretation designator k.

B SEQUENCE	C SEQUENCE	D SEQUENCE
	Clear X, Q & A$\longrightarrow$X	Store as modified by k
	(L(A)(Q)$\longrightarrow$X)	Sense j

50 SELECTIVE SET [SET $(A)_n$ FOR Y_n = 1]

Set the individual bits of (A) to one corresponding to ones in Y, leaving the remaining bits of (A) unaltered.

B SEQUENCE	C SEQUENCE
Operand$\longrightarrow$D	D$\longrightarrow$A (without clearing A)
	Sense j

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS— CONTINUED.

51 SELECTIVE COMPLEMENT [COMPLEMENT $(A)_n$ FOR $Y_n = 1$]

Complement the individual bits of (A) corresponding to ones in Y, leaving the remaining bits of (A) unaltered.

B SEQUENCE	C SEQUENCE
Operand $\rightarrow$ D	Clear X, A & D $\rightarrow$ X
	D $\rightarrow$ A
	Clear D, X' $\rightarrow$ D
	Clear X, A & D $\rightarrow$ X
	Clear A, X $\rightarrow$ A
	Sense j

52 SELECTIVE CLEAR [CLEAR $(A)_n$ FOR $Y_n = 1$]

Clear the individual bits of (A) corresponding to ones in Y, leaving the remaining bits of (A) unaltered. k = 7 should not be used in this instruction.

B SEQUENCE	C SEQUENCE
Operand' $\rightarrow$ D	Clear X, A & D $\rightarrow$ X
	Clear A, X $\rightarrow$ A
	Sense j

53 SELECTIVE SUBSTITUTE [$Y_n \rightarrow (A)_n$ FOR $(Q)_n = 1$]

Set the individual bits of (A) with bits of Y corresponding to ones in Q, leaving the remaining bits of A unaltered. k = 7 should not be used; if repeated, k = 0, 4 should not be used.

B SEQUENCE	C SEQUENCE
	Clear X, Q $\rightarrow$ X
	Clear D, X' $\rightarrow$ D
	Read operand $\rightarrow$ D (without clearing D)
	Q $\rightarrow$ A (without clearing A)
	Clear X, A & D $\rightarrow$ X
	Clear A, X $\rightarrow$ A
	Sense j

54 REPLACE SELECTIVE SET [SET $(A)_n$ FOR $(Y)_n = 1 \rightarrow$ Y & A]

Set the individual bits of (A) to one corresponding to ones in Y, leaving the remaining bits of (A) unaltered. Then store (A) at storage address Y.

B SEQUENCE	C SEQUENCE	D SEQUENCE
Operand $\rightarrow$ D	D $\rightarrow$ A	Store as modified by k
	Sense j	

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

55 REPLACE SELECTIVE COMPLEMENT [COMPLEMENT $(A)_n$ FOR $Y_n = 1 \longrightarrow \underline{Y}$ & A]

Complement the individual bits of (A) corresponding to ones in Y, leaving the remaining bits of (A) unaltered. Then store (A) at storage address $\underline{Y}$.

B SEQUENCE	C SEQUENCE	D SEQUENCE
Operand $\longrightarrow$ D	Clear X, A & D $\longrightarrow$ X	Store as modified by k
	D $\longrightarrow$ A	
	Clear D, X' $\longrightarrow$ D	
	Clear X, A & D $\longrightarrow$ X	
	Clear A, X $\longrightarrow$ A	
	Sense j	

56 REPLACE SELECTIVE CLEAR [CLEAR $(A)_n$ FOR $Y_n = 1 \longrightarrow \underline{Y}$ & A]

Clear individual bits of (A) corresponding to ones in Y, leaving the remaining bits of (A) unaltered. Then store (A) at storage address $\underline{Y}$.

B SEQUENCE	C SEQUENCE	D SEQUENCE
Operand' $\longrightarrow$ D	Clear X, A & D $\longrightarrow$ X	Store as modified by k
	Clear A, X $\longrightarrow$ A	
	Sense j	

57 REPLACE SELECTIVE SUBSTITUTE [$Y_n \longrightarrow (A)_n$ FOR $(Q)_n = 1 \longrightarrow \underline{Y}$]

Clear individual bits of (A) corresponding to ones in (Q), leaving the remaining bits of (A) unaltered. Then form the bit-by-bit product of Y and (Q) in the X register and set ones in this product in corresponding bits of (A), leaving the remaining bits of (A) unaltered. Then store (A) at storage address $\underline{Y}$.

B SEQUENCE	C SEQUENCE	D SEQUENCE
	Clear X, Q $\longrightarrow$ X	Store as modified by k
	Clear D, X' $\longrightarrow$ D	
	Operand $\longrightarrow$ D	
	(without clearing D)	
	Q $\longrightarrow$ A	
	(without clearing A)	
	Clear X, A & D $\longrightarrow$ X	
	Clear A, X $\longrightarrow$ A	
	Sense j	

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

60 JUMP (Arithmetic) [j = 0, NO JUMP;

j ≠ 0, JUMP TO ADDRESS Y̲ IF j SATISFIED]

This instruction clears program address register P and enters a new program address in P for certain conditions of either the A or Q register content. Branch condition designator j is interpreted in a special way for this instruction, and determines the conditions under which a jump in program address occurs. If the jump condition is not satisfied, the next sequential instruction in the current sequence is executed in a normal manner. If the jump condition is satisfied, as listed below, Y̲ becomes the address of the next instruction and the beginning of a new program sequence.

j = 0:	No jump. Set the interrupt enable to remove interrupt lockout, thus clearing Bootstrap and Interrupt modes. Continue with current program sequence.
j = 1:	Execute jump. Set interrupt enable to remove interrupt lockout, thus clearing Bootstrap and Interrupt modes.
j = 2:	Execute jump if (Q) is positive.
j = 3:	Execute jump if (Q) is negative.
j = 4:	Execute jump if (A) is zero.
j = 5:	Execute jump if (A) is nonzero.
j = 6:	Execute jump if (A) is positive.
j = 7:	Execute jump if (A) is negative.

B SEQUENCE	C SEQUENCE
Operand ⟶D	k = 0, 4;
	Clear R, D ⟶R
	If j is satisfied
	Clear P, R⟶P
	Set p to 0

61 JUMP (Manual) [JUMP TO ADDRESS Y̲ IF j SATISFIED]

This instruction clears program address register P and enters a new program address in P for certain conditions for manual JUMP selections. Branch condition designator j is interpreted in a special way for this instruction and determines the condition under which a jump in program address occurs. If the jump condition is not satisfied, the next sequential instruction in the current sequence is executed in a normal manner. If the jump condition is satisfied, Y̲ becomes the address of the next instruction and the beginning of a new program sequence, as listed below.

Program execute may be stopped by certain STOP selections upon execution of this instruction. Branch condition designator j specifies which selections are effective.

j = 0:	Execute jump (unconditional).
j = 1:	Execute jump if JUMP 1 is selected.
j = 2:	Execute jump if JUMP 2 is selected.

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

j = 3:	Execute jump if JUMP 3 is selected.
j = 4:	Execute jump. Stop computation.
j = 5:	Execute jump. Stop computation if STOP 5 is selected.
j = 6:	Execute jump. Stop computation if STOP 6 is selected.
j = 7:	Execute jump. Stop computation if STOP 7 is selected.

B SEQUENCE	C SEQUENCE	
Operand $\longrightarrow$ D	k $\neq$ 0, 4;	
	Clear R, D $\longrightarrow$ R	
	Clear P, R $\longrightarrow$ P	j = 1, 2, 3 and
	Clear p (0 $\longrightarrow$ p)	JUMP selected
	Clear P, R $\longrightarrow$ p	
	Clear p (0 $\longrightarrow$ p)	j = 4
	Stop at beginning	
	of next B sequence.	
	Clear P, R $\longrightarrow$ P	
	Clear p, 0 $\longrightarrow$ P	j = 5, 6, 7 and
	Stop at beginning	STOP selected
	of next B sequence	
	if option j set.	

62 JUMP ON $\hat{C}^j$ ACTIVE INPUT BUFFER [JUMP TO $\underline{Y}$ IF $C_{\hat{j}}$ INPUT BUFFER ACTIVE]

This instruction clears program address register P and enters a new program address in P for certain input buffer conditions on the channel designated by $\hat{j}$. If the buffer is active, the jump condition is satisfied, and $\underline{Y}$ becomes the address of the next instruction. If the buffer is inactive, the jump condition is not satisfied, and the next sequential instruction in the current sequence is executed in the normal manner. $\hat{k}$ = 0, 1, 2, or 3 is permitted.

B SEQUENCE		C SEQUENCE
Operand $\longrightarrow$ D		k $\neq$ 0, 4;
		Clear R, D $\longrightarrow$ R
	If Input	Clear P, R $\longrightarrow$ P
	Buffer Active	Clear p (0 $\longrightarrow$ p)

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

63 JUMP ON $C\hat{j}$ ACTIVE OUTPUT BUFFER [JUMP TO $\underline{Y}$ IF $C_{\hat{j}}$ OUTPUT BUFFER ACTIVE]

This instruction clears program address register P and enters a new address in P for certain output buffer conditions on the channel designated by $\hat{j}$. If the buffer is active, the jump condition is satisfied, and $\underline{Y}$ becomes the address of the next instruction. If the buffer is inactive, the jump condition is not satisfied and the next sequential instruction in the current sequence is executed in a normal manner. If the return jump condition is satisfied, as listed below, the following sequence is performed: Store (P) + p in the lower half of memory address $\underline{Y}$. Then jump to $\underline{Y}$ + 1.

j = 0:	No action. Continue with the current program sequence.
j = 1:	Execute return jump.
j = 2:	Execute return jump if (Q) is positive.
j = 3:	Execute return jump if (Q) is negative.
j = 4:	Execute return jump if (A) is zero.
j = 5:	Execute return jump if (A) is nonzero.
j = 6:	Execute return jump if (A) is positive.
j = 7:	Execute return jump if (A) is negative.

B SEQUENCE	C SEQUENCE	D SEQUENCE
Operand $\rightarrow$ D	k $\neq$ 0, 4	Clear S, U adder $\rightarrow$ S
	Clear R, D $\rightarrow$ R	If j satisfied $\left\{ \begin{array}{l} \text{(P) + p} \rightarrow Z \rightarrow M_L \\ \text{Clear P, S} \rightarrow P \\ +1 \rightarrow P \end{array} \right.$
	Clear R* & U_L	
	R $\rightarrow U_L$	

65 RETURN JUMP (Manual) [JUMP TO $\underline{Y}$ + 1 & (P) + p $\rightarrow \underline{Y}_L$ IF j SATISFIED]

This instruction executes a return jump sequence for certain conditions of manual JUMP or STOP options. Branch condition designator j is interpreted in a special way for this instruction, and determines the conditions under which the return jump sequence is executed. If the return jump condition is not satisfied, the next sequential instruction in the current sequence is executed in a normal manner. If the return jump condition is satisfied, as listed below, the following sequence is performed.

Store (P) + p in the lower half of memory address $\underline{Y}$. Then jump to $\underline{Y}$ + 1.

j = 0:	Execute return jump (unconditional).
j = 1:	Execute return jump if JUMP 1 is selected.
j = 2:	Execute return jump if JUMP 2 is selected.
j = 3:	Execute return jump if JUMP 3 is selected.
j = 4:	Execute return jump, then stop computation.
j = 5:	Execute return jump. Stop computation if STOP 5 is selected.
j = 6:	Execute return jump. Stop computation if STOP 6 is selected.
j = 7:	Execute return jump. Stop computation if STOP 7 is selected.

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

B SEQUENCE	C SEQUENCE	D SEQUENCE
Operand$\longrightarrow$D	k = 0, 4	If j satisfied:
	Clear R, D$\longrightarrow$R	Clear S, U Adder$\longrightarrow$S
	Clear R* & U_L	(P) + p$\longrightarrow Z_L \longrightarrow M_L$
	R$\longrightarrow U_L$	Clear P, S$\longrightarrow$P
		+1$\longrightarrow$p

Note

If a return jump immediately follows recognition of an interrupt by the control section (that is, if the return jump is the instruction stored at the Interrupt Entrance register), it is described as follows: Store P + 0, 1, or 2 in the lower half of memory address $\underline{Y}$. Then jump to $\underline{Y}$ +1. The p designator controls the modification of (P) and is set up by the instruction preceding the return jump caused by the interrupt. Therefore, the return jump causes the storage of the address of the next instruction that would have been executed if the interrupt had not occurred. The general description of the return jump is the jump to $\underline{Y}$ +1, with the understanding that in non-interrupt cases, p is set to "one", which causes the storage of P + 1 in $\underline{Y}$.

66 TERMINATE $C^{\hat{j}}$ INPUT BUFFER [CLEAR INPUT ACTIVE & MONITOR FLIP-FLOPS ON $C_{\hat{j}}$]

This instruction terminates the input buffer on channel $\hat{j}$. No input buffer monitor interrupt will occur. The operand interpretation designator $\hat{k}$, the index designator b, and the operand address designator y bits are not translated for this instruction.

B SEQUENCE	C SEQUENCE
	Clear Input Active Flip-Flops on channel $\hat{j}$
	Clear Input Monitor Flip-Flops on channel $\hat{j}$

67 TERMINATE $C^{\hat{j}}$ OUTPUT BUFFER [CLEAR OUTPUT ACTIVE & MONITOR FLIP-FLOPS ON $C_{\hat{j}}$]

This instruction terminates the output buffer on channel $\hat{j}$. No output buffer monitor interrupt will occur.

B SEQUENCE	C SEQUENCE
	Clear Output Active Flip-Flops on channel $\hat{j}$.
	Clear Output Monitor Flip-Flops on channel $\hat{j}$.

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

70 REPEAT [EXECUTE NEXT INSTRUCTION Y TIMES, $Y \longrightarrow B7$]

Clear B7 and transmit the lower 15 bits of Y to B7. If Y is nonzero, transmit j to r (designator register) thereby initiating the Repeat mode. If Y is zero, skip the instruction that was to be repeated.

REPEAT MODE: The Repeat mode executes the instruction immediately following the Repeat instruction Y times; B7 contains the number of executions remaining throughout the Repeat mode. If no skip condition is met for the repeated instruction, the Repeat mode terminates and the instruction following the repeated instruction is executed. If the skip condition for the repeated instruction is met, the Repeat mode terminates and the instruction following the repeated instruction is skipped. Following the Repeat mode termination, the count remains in B7. In no way does the Repeat mode alter a repeated instruction as stored in memory. The r designator (lower three bits of j from instruction 70) effects $(U)_L$ of repeated instruction are as follows:

Note

$r = j + 10$

r = 10:	Do not modify $(U)_L$ of the repeated instruction after each individual execution.
r = 11:	Increase $(U)_L$ of the repeated instruction by one after each execution of the repeated instruction.
r = 12:	Decrease $(U)_L$ of the repeated instruction by one after each execution of the repeated instruction.
r = 13:	Repeat the initial B register modification of the repeated instruction before each execution.
r = 14:	Do not modify $(U)_L$ of the repeated instruction after each individual execution; also, if the repeated instruction is a replace instruction, $(U)_L$ is incremented by (B_6) for the store portion of the Replace instruction.
r = 15:	Increase $(U)_L$ of the repeated instruction by one after each execution of the repeated instruction; also, if the repeated instruction is a replace instruction, $(U)_L$ is incremented by (B_6) for the store portion of the replace instruction.
r = 16:	Decrease $(U)_L$ of the repeated instruction by one after each execution of the repeated instruction; also, if the repeated instruction is a replace instruction, $(U)_L$ is incremented by (B_6) for the store portion of the replace instruction.
r = 17:	Repeat the initial B register modification of the repeated instruction before each execution; also, if the repeated instruction is a replace instruction, $(U)_L$ is incremented by (B_6) for the store portion of the replace instruction.

B SEQUENCE	C SEQUENCE
Operand $\longrightarrow$ D	k = 0, 4;
	Clear R, D $\longrightarrow$ R
	Clear B7, R $\longrightarrow$ B7

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

71 B SKIP ON B^j [$(B)_j = Y$ SKIP NI, CLEAR B_j: $(B)_j \neq Y$ ADVANCE B_j, EXECUTE NI]

If the content of B register j is equal to Y, skip the next instruction in the current sequence and proceed to the following instruction. Clear B register j. If the content of B register j is not equal to Y, proceed to the next sequential instruction in a normal manner. Increase the content of the B register j by one. Branch condition designator j is used to designate the selected B register in this instruction and is not available for its normal function. Only the lower-order 15 bits of Y are used in the comparison described.

B SEQUENCE	C SEQUENCE
Operand $\longrightarrow$ D	$k \neq 0, 4$;
	Clear R, D $\longrightarrow$ R
	Clear R* & U_L, R $\longrightarrow U_L$
	Clear R*, $B_j \longrightarrow$ R*
	$B_j = Y$ Set Abort FF
	$+1 \longrightarrow U_L$
	Clear R & B_j
	U Adder $\longrightarrow$ R
	$B_j \neq Y$ (Do not set Abort)
	R $\longrightarrow B_j$ (B_j +1)

72 B JUMP ON B^j [$(B)_j = 0$, EXECUTE NI: $(B)_j \neq 0$, $(B)_j - 1 \longrightarrow B_j$, JUMP TO $\underline{Y}$]

If the content of B register j is nonzero, execute a jump to program address $\underline{Y}$. Reduce the content of B register j by one. If the content of B register j is zero, proceed to next instruction in a normal manner. Branch condition designator j is used to designate the selected B register in this instruction and is not available for its normal function. If the jump condition is satisfied, the lower-order 15 bits of Y become the address of the next instruction and the beginning of a new program sequence. The higher-order 15 bits of Y are not used in this instruction.

B SEQUENCE	C SEQUENCE
Operand $\longrightarrow$ D	$k \neq 0, 4$
	Clear R, D $\longrightarrow$ R
	Clear R*, $B_j \longrightarrow$ R*
	$-1 \longrightarrow U_L$
	$B_j \neq 0$
	Clear P, R $\longrightarrow$ P
	Clear p ($0 \longrightarrow$ p)
	Clear R, U Adder $\longrightarrow$ R
	Clear B_j, R $\longrightarrow B_j$

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

73 INPUT BUFFER ON $C_{\hat{j}}$ (without monitor mode) [Set Input Active Flip-Flop on $C_{\hat{j}}$]

This instruction establishes an input buffer (input buffer channel $\hat{j}$) to magnetic core storage with an initial storage address $\underline{Y}$. Individual transfers will be executed at a rate determined by an external device. The storage address initially established by this instruction will be advanced by one following each individual transfer. The next address will be maintained throughout the buffer process in the lower-order 15 bits of magnetic core storage address 00100 plus $\hat{j}$. This mode will continue until it is terminated by f = 66 or until the higher-order half and the lower-order half of storage address 00100 plus $\hat{j}$ contain equal quantities. This instruction is implemented as follows: If $\hat{k}$ = 3, store (Y) in storage location 00100 plus $\hat{j}$. If $\hat{k}$ = 1, store the lower-order 15 bits of (Y) in the lower-order half of storage location 00100 plus $\hat{j}$, leaving the higher-order half undisturbed. If $\hat{k}$ = 0, store Y in the lower-order half of storage location 00100 plus $\hat{j}$, leaving the higher-order half undisturbed. ($\hat{k}$ = 2 is not permitted.)

B SEQUENCE	D SEQUENCE
Operand $\longrightarrow$ D & X	Clear Input Monitor Flip-Flop on $C_{\hat{j}}$
	Set Input Active Flip-Flop on $C_{\hat{j}}$
	Store as modified by $\hat{k}$

74 OUTPUT BUFFER ON $C_{\hat{j}}$ (without monitor mode) [SET Output Active Flip-Flop on $C_{\hat{j}}$]

This instruction establishes an output buffer (output buffer channel $\hat{j}$) from initial storage address $\underline{Y}$ in magnetic core storage. The individual transfers will be executed at a rate determined by an external device. The storage address initially established by this instruction will be advanced by one following each individual transfer. The next address will be maintained throughout the buffer process in the lower-order 15 bits of magnetic storage address 00120 plus $\hat{j}$. This mode will continue until it is terminated by f = 67 or until the higher-order half and the lower-order half of storage address 00120 plus $\hat{j}$ contain equal quantities. This instruction is implemented as follows: If $\hat{k}$ = 3, store (Y) in storage location 00120 plus $\hat{j}$. If $\hat{k}$ = 1, store the lower-order 15 bits of (Y) in the lower-order half of storage location 00120 plus $\hat{j}$, leaving the higher-order half undisturbed. If $\hat{k}$ = 0, store Y in the lower-order half of storage location 00120 plus $\hat{j}$, leaving the higher-order half undisturbed. ($\hat{k}$ = 2 is not permitted.)

B SEQUENCE	D SEQUENCE
Operand $\longrightarrow$ D & S	Clear Output Monitor Flip-Flop on $C_{\hat{j}}$
	Set Output Active Flip-Flop on $C_{\hat{j}}$
	Store as modified by $\hat{k}$

75 INPUT BUFFER ON $C_{\hat{j}}$ (with monitor mode) [Set Input Active & Monitor Flip-Flops on $C_{\hat{j}}$]

This instruction establishes an input buffer (input buffer channel $\hat{j}$) to magnetic core storage with an initial storage address $\underline{Y}$. The individual transfers will be executed at a rate determined by an external device. The storage address initially established by this instruction will be advanced by one following each individual transfer. The next address will be maintained throughout the buffer process in the lower-order 15 bits of magnetic core storage address 00100 plus $\hat{j}$. This mode will continue until it is terminated by f = 66, or until the higher-order half and lower-order half of storage address 00100 plus $\hat{j}$ contains equal quantities. The initiation of this input buffer selects the input channel $\hat{j}$ and establishes a buffer monitor on input channel $\hat{j}$. When the monitor interrupt follows the completion of the buffering operation, the next instruction is located at address

TABLE 10-2.—REPERTOIRE OF INSTRUCTIONS—CONTINUED.

(00040 + $\hat{j}$). This instruction is implemented as follows: If $\hat{k} = 3$, store (Y) in storage location 00100 plus $\hat{j}$. If $\hat{k} = 1$, store the lower-order 15 bits of (Y) in the lower-order half of storage location 00100 plus $\hat{j}$, leaving the higher-order bits undisturbed. If $\hat{k} = 0$, store Y in the lower-order half of storage location 00100 plus $\hat{j}$, leaving the higher-order half undisturbed. ($\hat{k} = 2$ is not permitted.)

B SEQUENCE	D SEQUENCE
Operand ⟶ D & X	Set Input Monitor Flip-Flop on $C_{\hat{j}}$
	Set Input Active Flip-Flop on $C_{\hat{j}}$
	Store as modified by $\hat{k}$

76 OUTPUT BUFFER ON $C_{\hat{j}}$ (with monitor mode) [Set Output Active & Monitor Flip-Flops on $C_{\hat{j}}$]

This instruction establishes an output buffer (output buffer on channel $\hat{j}$) from initial storage address <u>Y</u> in magnetic core storage. The individual transfers will be executed at a rate determined by an external device. The storage address initially established by this instruction will be advanced by one following each individual transfer. The next address will be maintained throughout the buffer process in the lower-order 15 bits of magnetic core storage address 00120 plus $\hat{j}$. This mode will continue until it is terminated by $f = 67$ or until the higher-order half and the lower-order half of storage address 00120 plus $\hat{j}$ contains equal quantities. The initiation of this output buffer selects the output channel $\hat{j}$ and establishes a buffer monitor on output channel $\hat{j}$. When the monitor interrupt follows the completion of the buffering operation, the next instruction is located at address (00060 + $\hat{j}$). This instruction is implemented as follows: If $\hat{k} = 3$, store (Y) at storage location 00120 plus $\hat{j}$. If $\hat{k} = 1$, store the lower-order 15 bits of (Y) in the lower-order half of storage location 00120 plus $\hat{j}$, leaving the higher-order half undisturbed. If $\hat{k} = 0$, store Y in the lower-order half of storage location 00120 plus $\hat{j}$, leaving the higher-order half undisturbed. ($\hat{k} = 2$ is not permitted.)

B SEQUENCE	D SEQUENCE
Operand ⟶ D & S	Set Ouptut Active Flip-Flop on $C_{\hat{j}}$
	Set Output Monitor Flip-Flop on $C_{\hat{j}}$
	Store as modified by $\hat{k}$

CHAPTER 11

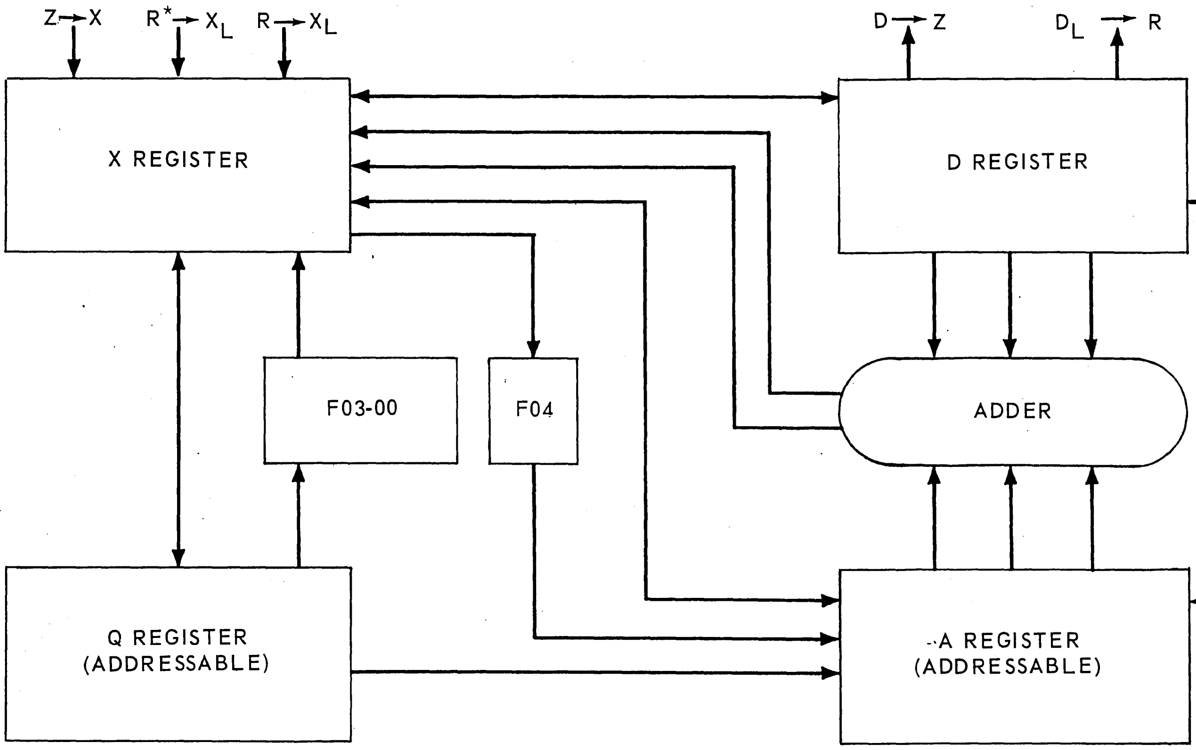
NTDS COMPUTER ARITHMETIC SECTION

The arithmetic section of the computer consists of registers, gates, and modifying circuits which perform the logical and arithmetic functions in the computer. Arithmetic operations are initiated and controlled by the control section and more directly by the C sequence. The principal registers in the arithmetic section are the A, D, Q, X and F-registers.

REGISTERS

The A, D, Q, X, and F-registers (fig. 11-1) participate in arithmetic operations and provide temporary storage for the results. The A- and Q-registers are addressable (capable of being loaded by an instruction) whereas the other can not. The A, X, D, and Q-registers are identical in that they consist of a total of 30 bit positions. All arithmetic registers are phase enabled to perform the clearing function on phase 2 and they all are phase enabled to receive data on phase 3.

Each stage of the A, D, and Q-registers has an indicator driver circuit (not shown) and dual-purpose indicator-pushbuttons (manual clear and manual set) on the console (fig. 10-8 chapter 8). The indicator is lighted when the



124.135

Figure 11-1.—Arithmetic section, block diagram.

associated indicator driver is set. Depressing a given indicator-pushbutton sets the associated internal flip-flop (not shown) and causes the indicator to light. Also associated with each register is a clear pushbutton. (The indicator-pushbutton module for the clear operation is connected so the indicator does not function.) Depressing the associated clear pushbutton clears the associated register when the phase 2 input is applied to the clearing circuits.

Table 11-1.—Basic Arithmetic Operations. (See table 10-1 for interpretations of f.)

Operation code		
f = 20	10 AUGEND	A Register
	+ 5 ADDEND	D Register
	15 SUM	X Register then to A
f = 21	10 MINUEND	A Register
	- 5 SUBTRAHEND	D Register (contains complement of subtrahend)
	5 DIFFERENCE	X Register then to A
f = 22	10 MULTIPLICAND	D Register
	x 5 MULTIPLIER	Q Register
	50 PRODUCT	AQ Register (double length) Q (single length)
f = 23	DIVISOR $\begin{array}{r}2\\5\overline{)10}\end{array}$	QUOTIENT Q Register
	D Register	DIVIDEND AQ Register
	REMAINDER (if any)	A Register

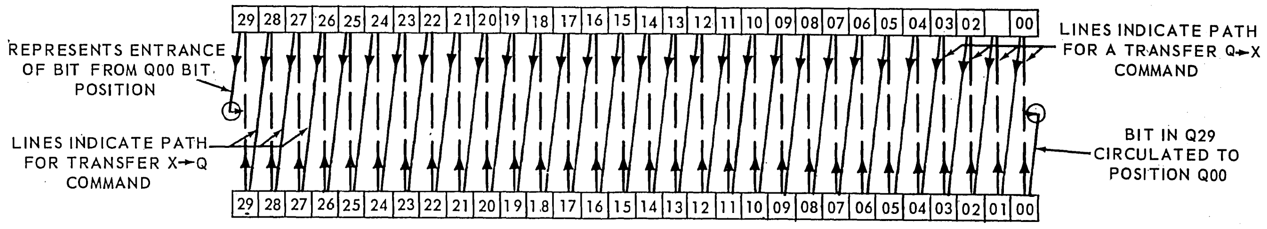
Note: Values shown in decimal numbers for simplicity. Binary numbers used in actual computer operations.

A-, D-, AND Q-REGISTERS

The A-, D-, and Q-registers participate in arithmetic operations in the following ways: in addition, (see table 11-1) the augend is in the A-register, the addend is in the D-register and the sum (from the adder) will be temporarily stored in the X-register and is then generally transferred to the A-register; in subtraction, the minuend is in the A-register the complement of the subtrahend is the D-register and the difference will be temporarily stored in the X-register and is then generally transferred to the A-register; in multiplication the multiplicand is in the D-register, the multiplier is in the Q-register, and the product is formed in the AQ-register (see note below); in division, the dividend is in the AQ-registers (used as a single 60-bit register) the divisor is in the D-register, the quotient is formed (by subtracting the divisor from the dividend and left shifting) in the Q-register, and if there is a remainder it is located in the A-register.

NOTE: During multiplication, the A-and Q-registers (30-bit each) are formed and used as a double-length 60-bit register in order to hold a product greater than 30 bits. Multiplication by each bit in the multiplier (held in the Q-register) is accomplished by adding the multiplicand to itself (when the multiplier bit is 1) and right shifting. (A multiplier bit of 0 causes only 0's to be added to the multiplicand, after which the right shift occurs. The basic multiplication process is explained in chapter 5.) The right shift process after consideration of each multiplier bit causes the multiplier to be right-shifted out of the Q register, being replaced by the least significant digits of the product. The final product is held in the AQ-register. The A-register holds the most significant bits of the product if the product is greater than 30 bits.

Shift operations are performed by displacing digits one or more columns to the left or right during transmission from one register to another. Note the Left Shift Q by One operation as illustrated in figure 11-2. The transmission Q → X is normal (i.e., no shift is involved) but, by design, the transmission X → Q displaces each bit of the X-register one column to the left. The left shift Q by one command is written as XL1 → Q. Thus, XR4 → Q is a right shift of four. A left shift is performed circularly i.e., bits shifted out of a register at the left end are entered in the right end as the shift progresses. Right shifts are open-ended with sign extension, i.e., as the right shift progresses, the displaced digits are replaced with the original sign of the number, with the least significant bits lost.



REPRESENTS ENTRANCE OF BIT FROM Q00 BIT POSITION
LINES INDICATE PATH FOR TRANSFER X→Q COMMAND
LINES INDICATE PATH FOR A TRANSFER Q→X COMMAND
BIT IN Q29 CIRCULATED TO POSITION Q00
29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 00
29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

124.136

Figure 11-2.—Left shift Q by one.

X-REGISTER

Like the A-, D-, and Q-registers the X-register (fig. 11-3) is also divided into six sections with five stages per section. (Only one section is shown in figure 11-3.) The clearing function for the X-register is phase enabled at phase 4 time and data inputs are phase enabled at phase 1 time. The X-register logic is different from other registers. Generally data inputs to registers are applied to the one side of the flip-flops and clear pulses are applied to the zero side. The operation of the X-register is the opposite of this in that data inputs to the X register are applied to the zero side and clear pulses are applied to the one side of the flip-flops. For this reason, the output from the transmitting register is taken from the zero side and transferred to the X-register. During transmissions from the X-register, the output is taken from one side. Transmission from the zero side of the X-register flip-flops results in the number being complemented. The X-register logic facilitates certain functions such as the formation of logical products. (Because the most important operation in transferring data is the clearing and entering of data into the receiving register, most transmissions are discussed in terms of the receiving register.)

F-REGISTER

The F-register (fig. 11-1) is a special-purpose, five-stage register. It is used for storing bits during shifts involving the A- and Q-registers. It also stores Q00 which indicates if addition is necessary during the multiply sequence (discussed later).

The F-register is set or cleared in the same manner as the A-, D-, or Q-registers, except that F is cleared on clock phase 4. With the exception of F04, information is gated into the F-register on clock phase 1. Information is gated into F04 on clock phase 3. The only input commands that affect the F-register are Q03-00 ⟶ F03-00 and X00 ⟶ F04.

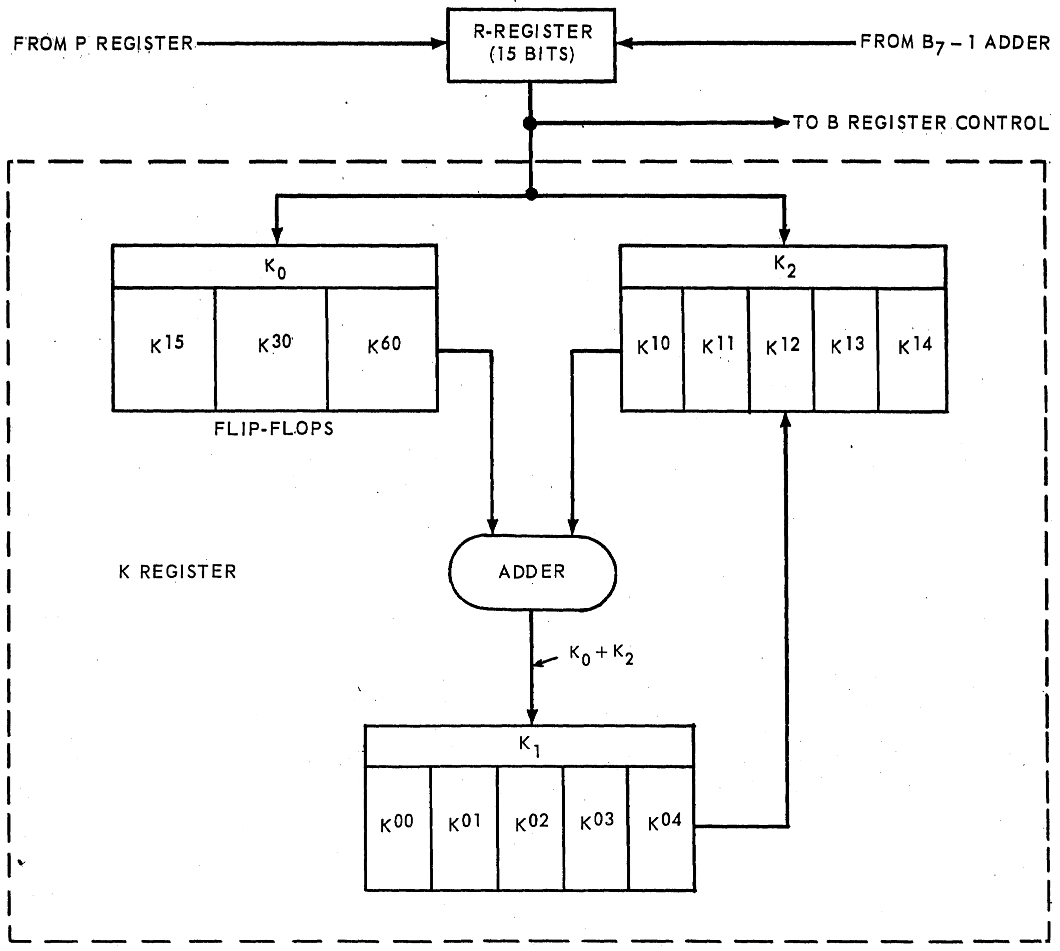
K-REGISTERS

Before a discussion of multiply and divide operations (as will be treated later) can be understood it is necessary to consider the following brief description of the K-register. The K-registers and the adder function together to maintain the shift count for shift instructions and multiply or divide instructions. (The K-register is not a part of the arithmetic unit.)

There are three major sections of the K-register (fig. 11-4): K0 which contains flip-flops K^{15}, K^{30}, and K^{60}; K2 which contains flip-flops $K1^{0}$ through $K1^{4}$; and K1 which contains flip-flops $K0^{0}$ through $K0^{4}$. In addition to the registers there is also an adder that produces the sum of K0 and K2.

K0 Register

The K0 register (fig. 11-4) is a three-bit register which contains K^{15}, K^{30}, and K^{60} (flip-flops shown in block form). Flip-flop K^{15} is unconditionally set to cause a 1-place right shift for multiply or divide instructions and for all shift instructions if the shift count (SK) is 15 through 29 (bit positions of Y for a single length word) or 45 through 59 (bit positions of Y for a single length word). Flipflop K^{30}, is set only during a shift instruction if the shift count is greater than or equal to 30_{10}. Flip-flop K^{60} is used for a multiply instruction and is set when the lower five bits of Q (the multiplier) are equal to zero. This causes a right shift of four in addition to the right shift of one, caused by K^{15}. A number indicating the number of shifts to be executed is temporarily held in



124.138

Figure 11-4.—K-register.

the R-register. This number must be transferred to the K-register which controls and tabulates shifts. During the R⟶K transmission the lower six bits of the R-register are sensed by the K0 input circuitry.

K2 Register

The K2 register consists of stages K^{10} through K^{14}. Stage K^{14} is used only during multiply or divide instructions; the remaining stages are used for shift and multiply or divide instructions. The lower four bits of the shift count are entered into the K2 register form the R-register. K2 receives shift count data from the K1 register during a multiply or divide instruction.

K1 Register

The K1 register consists of stages K^{00} through K^{04}. Flip-flop K^{04} is used only for multiply or divide instructions. The only data input to K1 is from the K adder. The K adder supplies the result of K0 + K2 or its complement. The K1 register holds the total shifts completed for multiply or divide and holds the remaining shifts to be done for shift instruction.

78A04
77A04
00R19
02Z04
02R04
00Q08
00Q04
00A08
00A04
78A03
77A03
00R18
02Z03
00R03
00D03
00Q07
00Q03
00A07
00A03
78A02
77A02
00R17
02Z02
00R02
00D02
00Q06
00Q02
00A06
00A02
78A01
77A01
00R16
02Z01
02R01
00D01
00Q05
00Q01
00A05
00A01
56N34
11N10
78A00
77A00
00R15
02Z00
11N04
02R00
11N08
00D00
11N14
11N21
00Q00
51N12
00A00
56N33
56N11
11N06
56N31
56N13
11N02
80N12
49N01
49N00
95Y01 (Ø1)
95Y04 (Ø2)
D→X
AR4→X
QR4→X
ADDER→X
R*→XL
Z→X
R→XL
Q→X
A→X
CLEAR X
Ø1
19D04
23D19
23Q04
13Q04
15Q03
17Q05
13D04
17D19
19Q19
X4
15A04
17A05
19A03
19D03
23D18
23Q03
13Q03
15Q02
17Q04
13D03
17D18
19Q18
X3
15A03
17A04
19A02
19D02
23D17
23Q02
13Q02
15Q01
17Q03
13D02
17D17
19Q17
X2
15A02
17A03
19A01
19D01
23D16
23Q01
13Q01
15Q00
17Q02
13D01
17D16
19Q16
X1
15A01
17A02
19A00
13Q00
13D00
15A00
17A01
17Q01
15Q29
17F04
19A29
17D15
19Q15
X0
19D00
23D15
23Q00
11X10
11X05

Figure 11-3.—X-register, 00-04.

124.137

An example of the data exchange in K using a shift count of 39_{10} is as follows:

R	100111	$=39_{10}$	(lower six bits of R)
K	0111 (K2)		(lower four bits of R)
K^{30} is set	010 (K0)		(output of K0 register, where $K^{15} = 0$, $K^{30} = 1$, and $K^{60} = 0$)
	1001	K adder $\longrightarrow$ K1	(K1 holds counts remaining)

The shift count of 39_{10} is entered in the R-register. The command R$\longrightarrow$K causes K^{30} to be set and K2 (bits K^{10} - K^{13}) receives 0111 from R. The next command (K adder$\longrightarrow$K1) places the sum of K0 and K2 in the K1 register. The Shift Setup sequence resulting from the setting of K^{30} accomplishes the 30_{10} place shift leaving nine to be done by Shift Step and Shift Step control. The K1 register holds a count of nine which is sensed by Shift Step and Shift Step control, and the nine-place shift is performed.

ADDER

The arithmetic adder (fig. 11-1) in the arithmetic section of the computer is composed of 30 stages subdivided into six sections of five stages each (fig. 11-5). Each section and each corresponding stage in a section are identical with regard to circuits and functional operation. The 30 stages of the Adder consist of gates and inverters (shown later) in a logical array for carrying out the required addition and subtraction operations in the arithmetic section.

HALF-SUBTRACT PROCESS

The Adder is of the subtractive, "1's-complement," end-around borrow, type. In accomplishing its addition and subtraction operations, the Adder utilizes a half-subtract process. (Examples are given later.) In this process, the addend must be complemented and each bit position is considered as an independent subtraction problem. Each half-subtract process uses the conventional rules for binary subtraction without utilizing a borrow from a higher-order bit position.

Before the final result is obtained, any borrow generated by the half-subtract process must be satisfied. The borrow being generated by any bit position is satisfied by a higher-order position. The borrow is carried through each position (even end-around) until it is satisfied.

ADDITION (SUBTRACTIVE)

In the computer, the two inputs to the Adder are from the A-register (accumulator) and the D-register (whose input represents the operand). The contents of the two registers are always combined in the Adder, regardless of the instruction in the computer. The output from the Adder, however, is not transmitted until an Adder$\longrightarrow$X command enable is received. The Adder$\longrightarrow$X command temporarily stores the sum in the X-register. The sum is held in X until the A-register is cleared. Afterwards, the command X$\longrightarrow$A is initiated to cause the sum to be stored in the A-register.

Due to the characteristics of the Adder logic the contents of D are complemented (or apparently complemented) in the Adder when being combined with contents of A in the Adder.

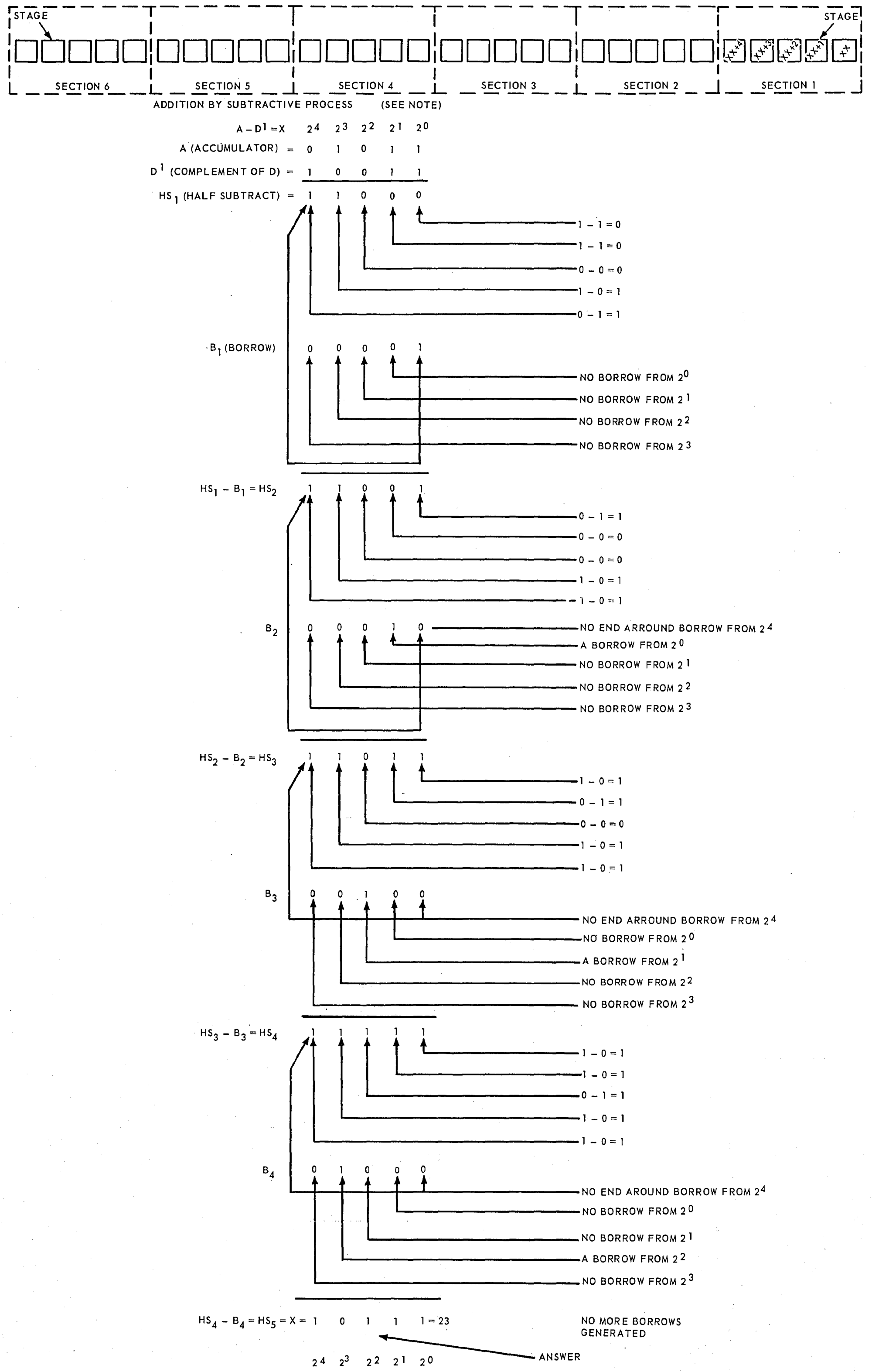
Examples of normal binary addition and addition by the subtractive process are shown below.

Normal Addition:

Y = operand	=	01011	11_{10}
A = accumulator	=	01100	12_{10}
		10111	23_{10}

EXAMPLE .... Adding by subtractive process using 1's complement half - subtract, (HS) end around Borrow.

This example (fig. 11-5) illustrates addition by the subtractive process but does not represent the exact procedure used in the computer. The borrows and half-subtract process are not treated in succession as shown. Note that the Adder accomplished the addition of the A+D by actually doing an A-D' operation (where D' is the complement of D). The resultant sum is the same for both the normal addition method or addition by the subtractive process.



124.139

Figure 11-5.—Block diagram of adder and addition by subtractive process.

The Adder (fig. 11-1) accomplishes a subtract instruction (f = 21) in essentially the same manner except that the original data transmission from X to D enters the complement of X in D. This is complemented in the Adder to give (D'). Using the half-subtract process, the problem becomes A - D' which provides the difference of (A) minus (Y). The following example illustrates subtraction by the subtractive process:

```
           Y  10101  21₁₀
           A  11011  27₁₀
           D  01010         (again comple-
           D' 10101          mented in adder
Minuend       11011  A-(D')  to yield subtra-
Subtrahend    10101          hend 10101 D')
              01110
              01000
              00110
              00000
Final Result  00110  6₁₀
```

NOTE: This example illustrates subtraction by the subtractive process but does not represent the exact procedure used in the computer. The borrows and half-subtract processes are not treated in succession as shown.

ADDER LOGIC

Because all sections in the Adder are identical from a logic standpoint to every other section, the following detailed discussion of one section and its five stages applies to all sections. The make-up of the Adder is such that circuits can be separated into composite circuits, each having definite functional operations. The discussion of the Adder circuits includes: (1) input circuit, (2) intersection borrow request circuit, (3) intersection borrow enable circuit, (4) section borrow input circuit, and (5) intrasection and output circuit.

Input

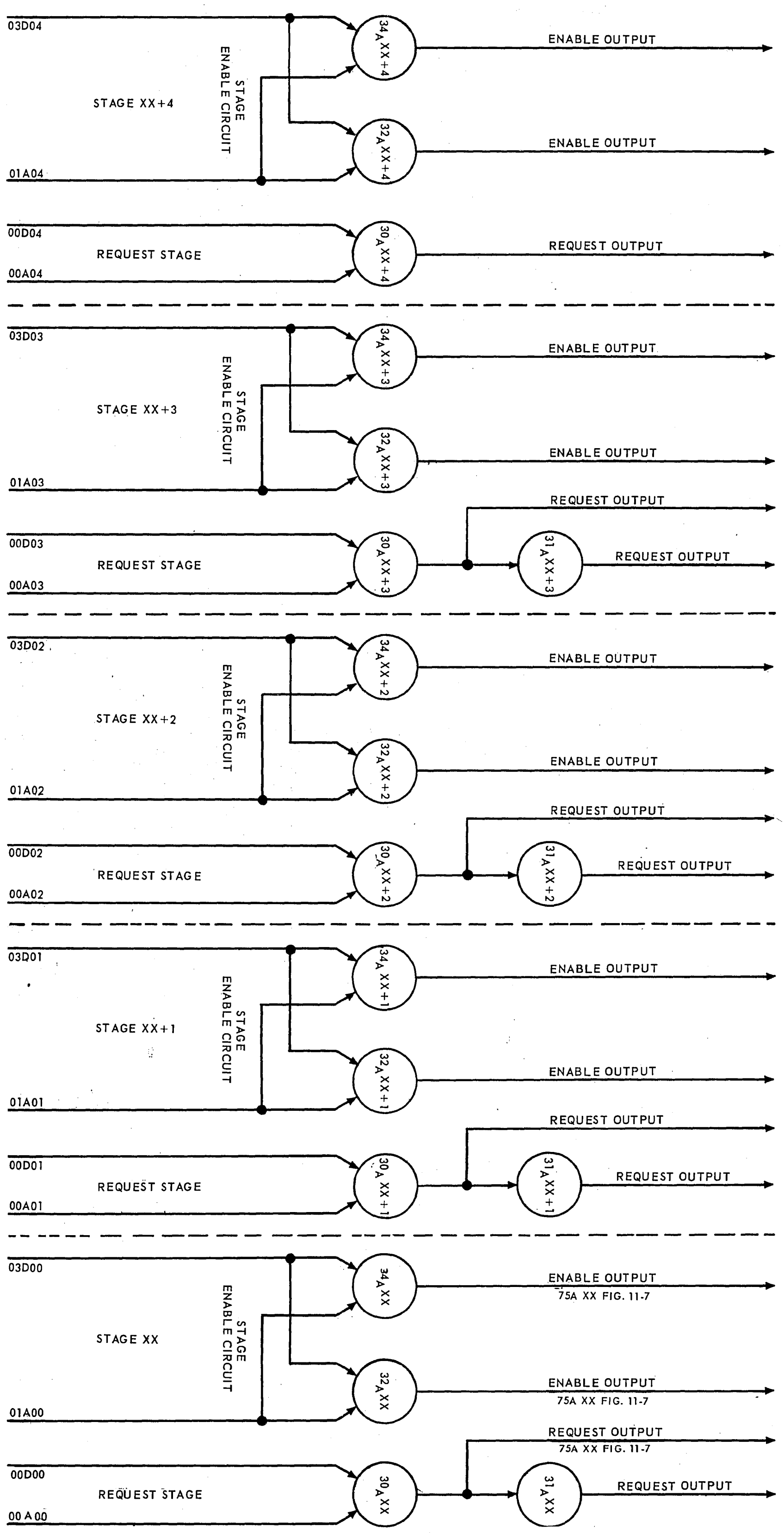
The input circuits of each stage of the Adder (fig. 11-6) examine the state of the A and D register flip-flops (not shown). Each of the first four stages in each section (stages XX thru XX + 3) has identical input circuits, while the fifth stage (XX + 4) differs slightly from the first four. The inputs form the A register and the D-register are combinations of "1's" and "0's". These are combined by the five stages shown to generate outputs that: (1) indicate the borrow request status of each stage; (2) indicate the borrow enable status of each stage; and (3) are used in the half-subtract process of each stage.

The inputs to the stage enable circuits are from the one side of the flip-flops in the A- and D-registers, while the inputs to the request stage circuits are from the zero side of the flip-flops in the two registers. Table 11-2 and the interpretations given below the table are used to explain the logic combinations of 1's and 0's to the various circuits as they are applied in the Adder.

Table 11-2.—Summary of Input Circuit Signals.

Input Combinations	(a)	(b)	(c)	(d)	Outputs	(a)	(b)	(c)	(d)	Signal Functions
00A–	0	0	1	1	30A–	1	0	0	0	"1" output from any 30A—stage indicates borrow request from that stage.*
00D–	0	1	0	1	31A–	0	1	1	1	"1" output indicates no borrow request from first four stages (Complement of 30A—)
01A–	1	1	0	0	32A– 34A–	0	0	0	1	Borrow enable from all stages.
01D–	1	0	1	0						

*The generation of a borrow request when the A and D inputs are both 0, is caused by the logic circuitry used in the "adder", whereby D is apparently complemented. Note that when using NOR logic (as represented here by using inverters) a "1" output is produced only when both inputs are 0's. This produces the same result as the consideration of A - D' where D' is the complement of D, and the half-subtract process involves 0 - 1.



124.140

Figure 11-6.—Input circuits to adder.

Explanation of signal combinations:

(a) When the combination of input signals from the A- and D-registers is such that A = 0 and D = 0, the "1" output from 30A— and the "0" output from 31A— indicate an unconditional borrow request from a stage. The "0" output from 32A— and 34A— indicates that a stage cannot satisfy any borrow request without a borrow.

(b) When A = 0 and D = 1, the "0" output from 30A— and the "1" output from 31A— indicate that a stage is not generating a borrow request. The "0" output from 32A— and 34A— is the same as explained in (a) above.

(c) When A = 1 and D = 0 the condition is exactly as explained in (b).

(d) When A = 1 and D = 1, the "0" output from 30A— and the "1" output from 31A— indicate no borrow as in (b). The "1" output from 32A— and 34A— indicates that a stage can satisfy any borrow request without a reborrow.

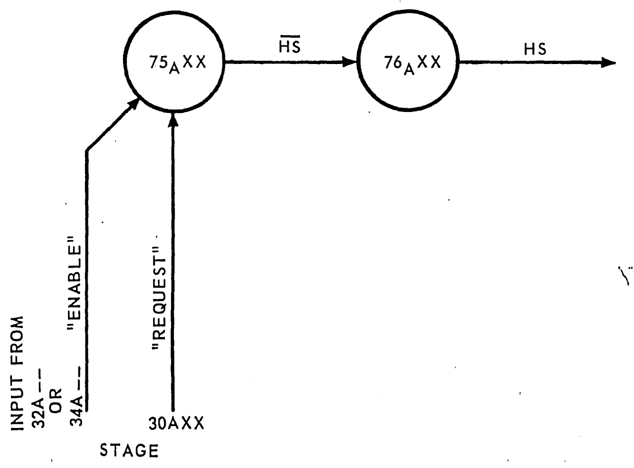
The inputs to the stage enable circuits are from the one side of the flip-flops in the A- and D-registers, while the inputs to the stage requests circuits are from the zero side of the flip-flops in the two registers.

The request outputs indicate whether a particular stage needs, or does not need, a borrow. The enable output indicates whether a particular stage can, or cannot, satisfy a borrow request from a lower-order stage or section without a reborrow. These enable and request outputs are transmitted to other circuits in the Adder to be the basis for signals which are generated as: (1) intersection borrow requests, (2) intersection borrow enables, and (3) intersection communications.

The enable output from either 32A— or 34A— (see stage XX enables output terminals) along with the request output from 30A— are combined in the 75A— circuit (fig. 11-7). The output from 75A— is a logical "1" or "0" that is always the complement of the half-subtract result ($\overline{HS}$) for the inputs to the stage. The output from 75A— ($\overline{HS}$) is then inverted by 76A— to give the half-subtract result (HS). (The purpose of this output is explained later.) It is because of this logical procedure of the half-subtract process that the contents of D appear to be complemented in the Adder.

Intersection Borrow Request

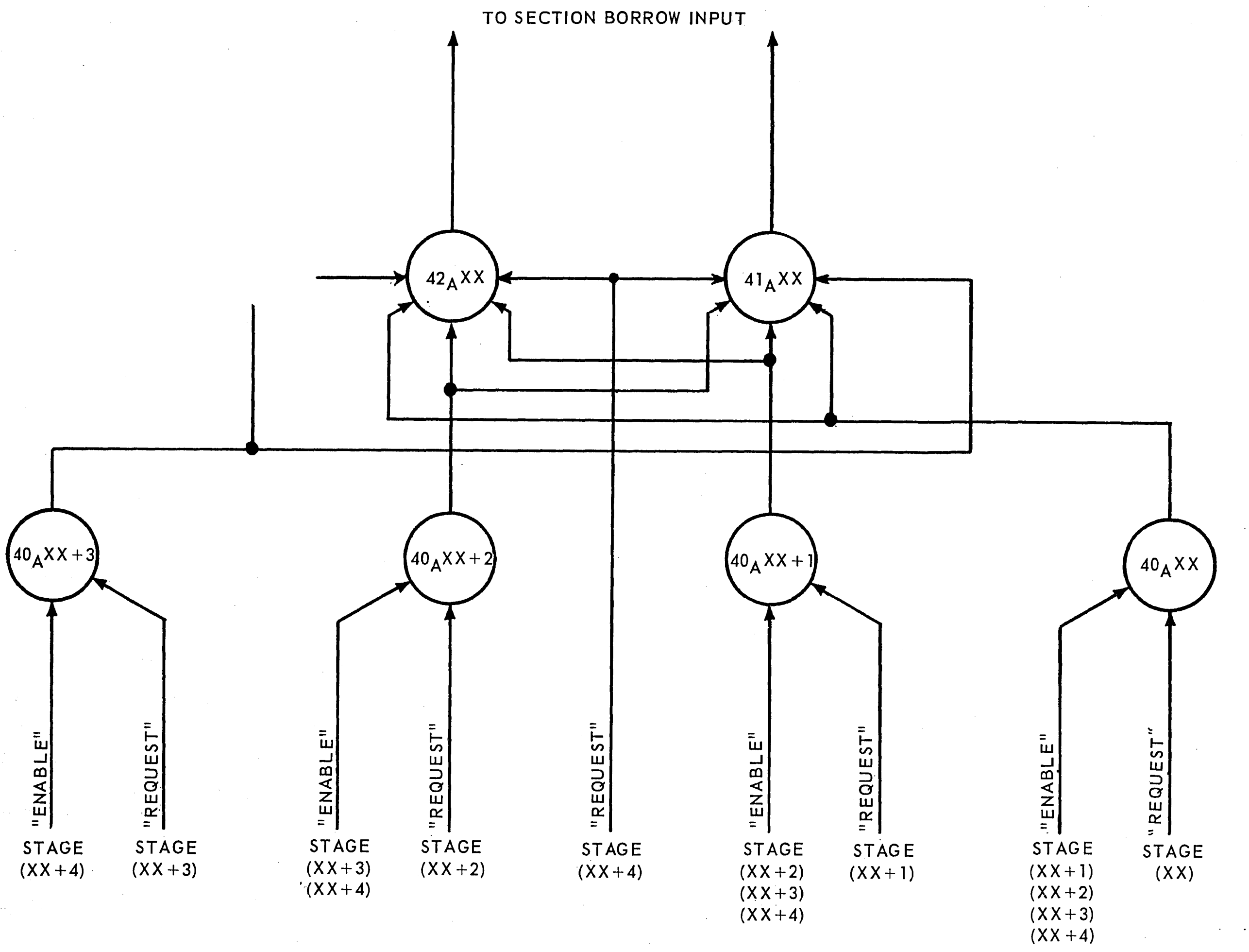
The intersection borrow request signal is generated by a section to inform all sections of its borrow request status. Figure 11-8 is representative of any intersection borrow request circuit in the Adder and receives inputs from stage (XX), the first stage through (XX + 3), the fourth stage (fig. 11-5).



124.141
Figure 11-7.—Simplified schematic of half-subtract.

The input circuits of the Adder (fig. 11-6) generate stage information that is translated by the circuits of figure 11-8. In effect, each 40A— circuit has two inputs from the Adder input: a Request signal and an Enable signal. The request input is from the particular stage under consideration; the enable input is a composite Enable signal from higher order stages. The composite Enable signal is such that it is a "0" if all enable lines are "0's", and is a "1" if any or all enable lines are "1's". If stage (XX) is under consideration, the enable input to 40A— would be the composite Enable signal from the four higher-order stages; that is, stage (XX + 1) through stage (XX + 4).

The output from each of the 40A— circuits can be either a logical "1" or "0". This output indicates the borrow request status of each of the first five stages. If "0's" appear on both the request and enable input lines, the "1" output from 40A— indicates an intersection borrow request from a stage in the section. For example, if stage (XX) generates a borrow request that cannot be satisfied in a higher-order stage without a reborrow, the output from 40AXX is a "1" due to the "0" request input and "0" enable input. The output from 40AXX would be a "0" to indicate a no-borrow condition from



124.142

Figure 11-8.—Simplified schematic of intersection borrow request circuit.

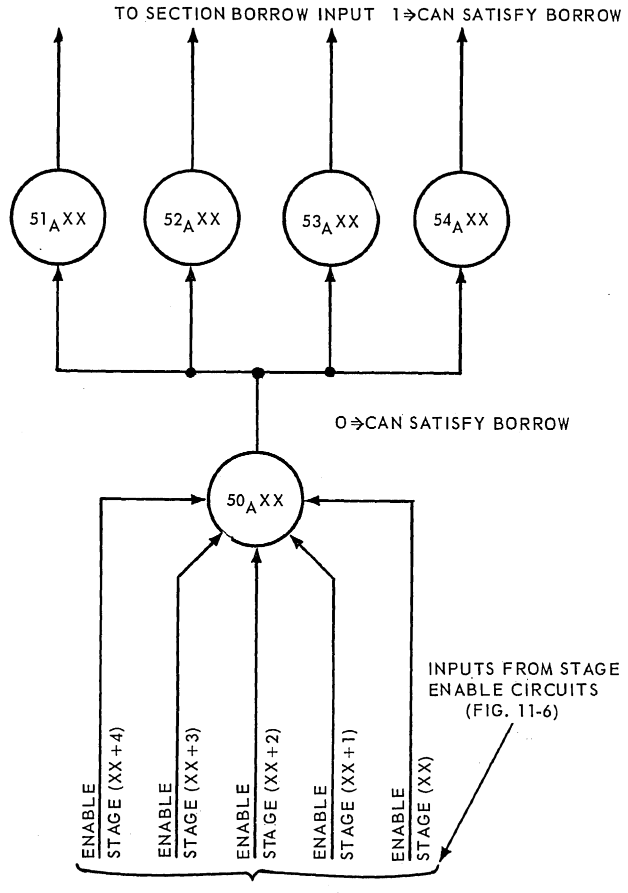
stage (XX) if either one or both inputs are "1's". Likewise, the output from any 40A— circuit is a "0" to indicate a no-borrow condition from the stage which supplies its request input. A "1" request input indicates no-borrow request from stage (XX). A "1" enable input indicates that a borrow request from stage (XX) can be satisfied within the section without a reborrow.

The outputs from the 40A— circuits are combined in the 41AXX and the 42AXX circuits to provide outputs that indicate the overall borrow-request status of a section. If any or all inputs are "1's", the outputs from 41AXX and 42AXX are "0's" to indicate an intersection borrow request. Thus, if the input from stage XX + 4 (the final stage in a section) is a "1", an automatic intersection borrow request is indicated. If the outputs from 41AXX and 42AXX are "1's", a no-borrow condition or "all-borrows-within-the-section-have-been-satisfied" condition is indicated. The output signals from 41AXX and 42AXX are sent to the section borrow input circuits (shown later) in all sections as information on the borrow-request status of the particular section under consideration.

Intersection Borrow Enable

Intersection borrow enables are generated by a section to inform all other sections of its borrow-enable status. Stage information regarding each stage of a section, generated by the input circuits (fig. 11-6) is combined by the 50AXX circuit (fig. 11-9) to generate intersection borrow Enable signals. The 50AXX circuit receives inputs on an enable line from each stage; and the circuit output is a logical "1" or "0". This indicates the borrow enable status of a section. With "0" inputs on the enable lines, the output from 50AXX is a "1". This indicates that a section cannot satisfy a borrow request from any other section without a reborrow. When any or all input enable lines are "1", the output from 50AXX is "0". This indicates that the section can satisfy a borrow request from any other section without a reborrow.

The output signals from 50AXX are complemented by four circuits: 51AXX, 52AXX, 53AXX, and 54AXX before being sent to the section borrow input circuits in all other sections as information on the borrow-enable status of the section under consideration (see table 11-3).



124.143

Figure 11-9.—Simplified schematic of intersection borrow enable circuits.

Table 11-3.—Intersection Borrow Enable Signals.

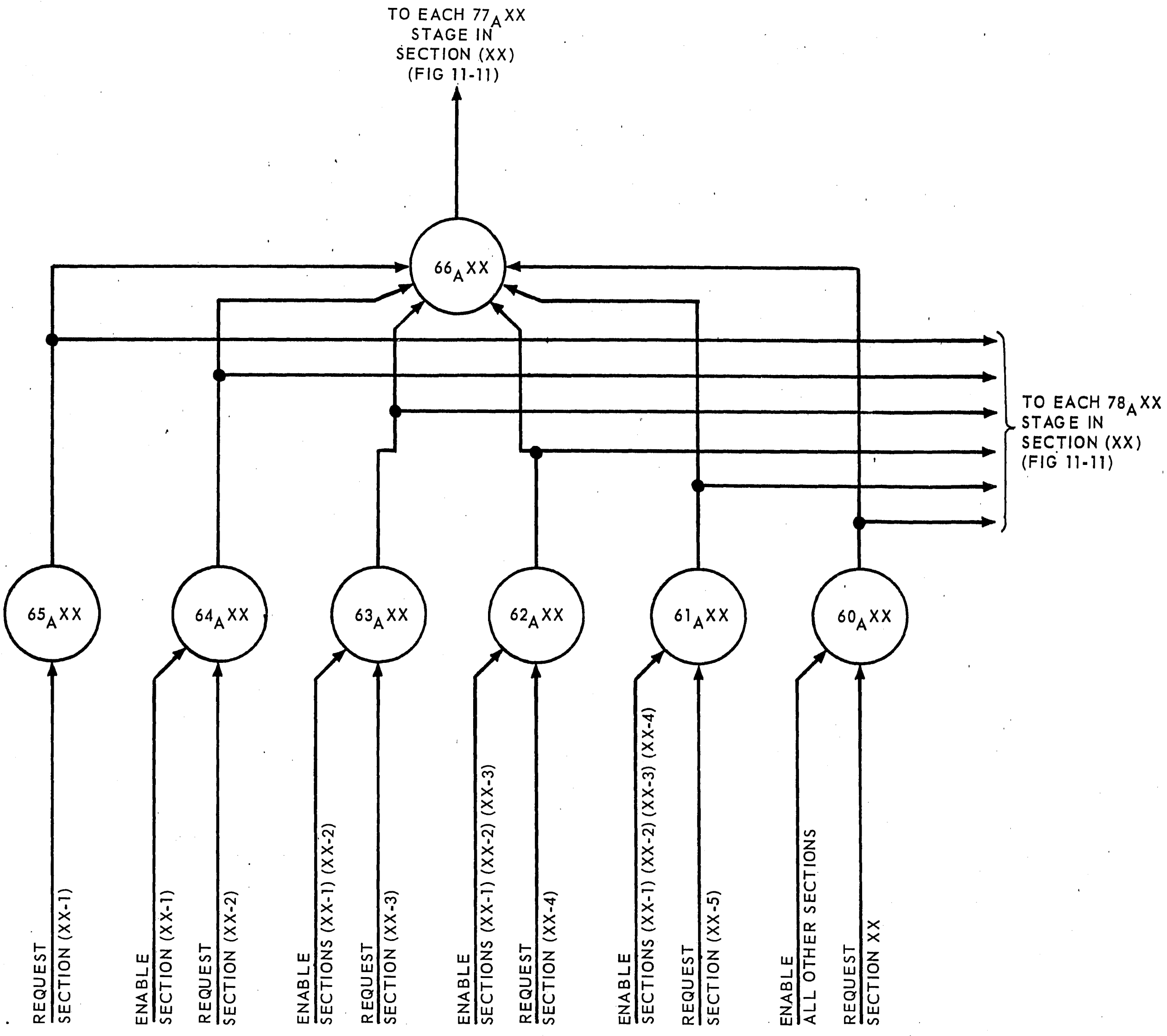
Circuit	Outputs	Signal Function
50AXX	0	Can satisfy an intersection borrow request from any other section.
	1	Cannot satisfy an intersection borrow request without a reborrow.
51AXX, 52AXX, 53AXX, 54AXX	0	Same as "1" output from 50AXX
	1	Same as "0" output from 50AXX

Section Borrow Input

A section borrow input signal is generated in each section to indicate the overall borrow request status of all sections. In order to understand the section borrow action, first note that if Section (XX) is taken to be any section in the Adder; then section (XX - 1) is the preceding section; section (XX - 2) is two sections preceding, etc. In this discussion we will adhere to the following procedure: if section 1 of the Adder is under consideration, section (XX—1) is section 6, and section XX - 5) is section 2 etc. (This is true only in a system of numbering where stage 0 does not exist.)

The section borrow input circuits (fig. 11-10) are used in the Adder to combine the intersection borrow request and the intersection borrow enable output signals from all sections. Its outputs indicate: (1) section or sections having an intersection borrow request that has not been satisfied by any intervening section, and (2) no intersection borrow requests.

With reference to figure 11-10 there is a request input to each of the six circuits (65AXX through 60AXX) from each section including the section under consideration, section (XX). The arrangement is such that 65AXX receives the request input from the preceding section (XX - 1), 64AXX from section (XX - 2) ..., and 60AXX from section (XX). The enable input, (derived from each section as illustrated in table 11-3)


TO EACH 77AXX STAGE IN SECTION (XX) (FIG 11-11)
66AXX
TO EACH 78AXX STAGE IN SECTION (XX) (FIG 11-11)
65AXX
64AXX
63AXX
62AXX
61AXX
60AXX
REQUEST SECTION (XX-1)
ENABLE SECTION (XX-1)
REQUEST SECTION (XX-2)
ENABLE SECTIONS (XX-1) (XX-2)
REQUEST SECTION (XX-3)
ENABLE SECTIONS (XX-1) (XX-2) (XX-3)
REQUEST SECTION (XX-4)
ENABLE SECTIONS (XX-1) (XX-2) (XX-3) (XX-4)
REQUEST SECTION (XX-5)
ENABLE ALL OTHER SECTIONS
REQUEST SECTION XX

124.144

Figure 11-10.—Simplified schematic of section borrow input circuits.

is a composite enable signal from intervening sections between section (XX) and the section initiating the borrow request. The output from any of the six circuits, 65AXX through 60AXX, can be either a logical "1" or "0". This output indicates the borrow request of each section. For example, if "0's" appear on both the request and the enable input lines to 62AXX, its "1" output indicates that a borrow request from section (XX - 4) has not been satisfied in any of the intervening sections. Thus it appears as a borrow input to section XX. If the output from 62AXX is a "0" indicating a no-borrow condition from section (XX - 4), either one or both input lines can be a "1". A "1" enable input indicates that a borrow request from section (XX - 4) can be satisfied by an intervening section. The borrow request status of each section is effectively determined in the same manner, including the status of section (XX). Its status is determined by the output of 60AXX.

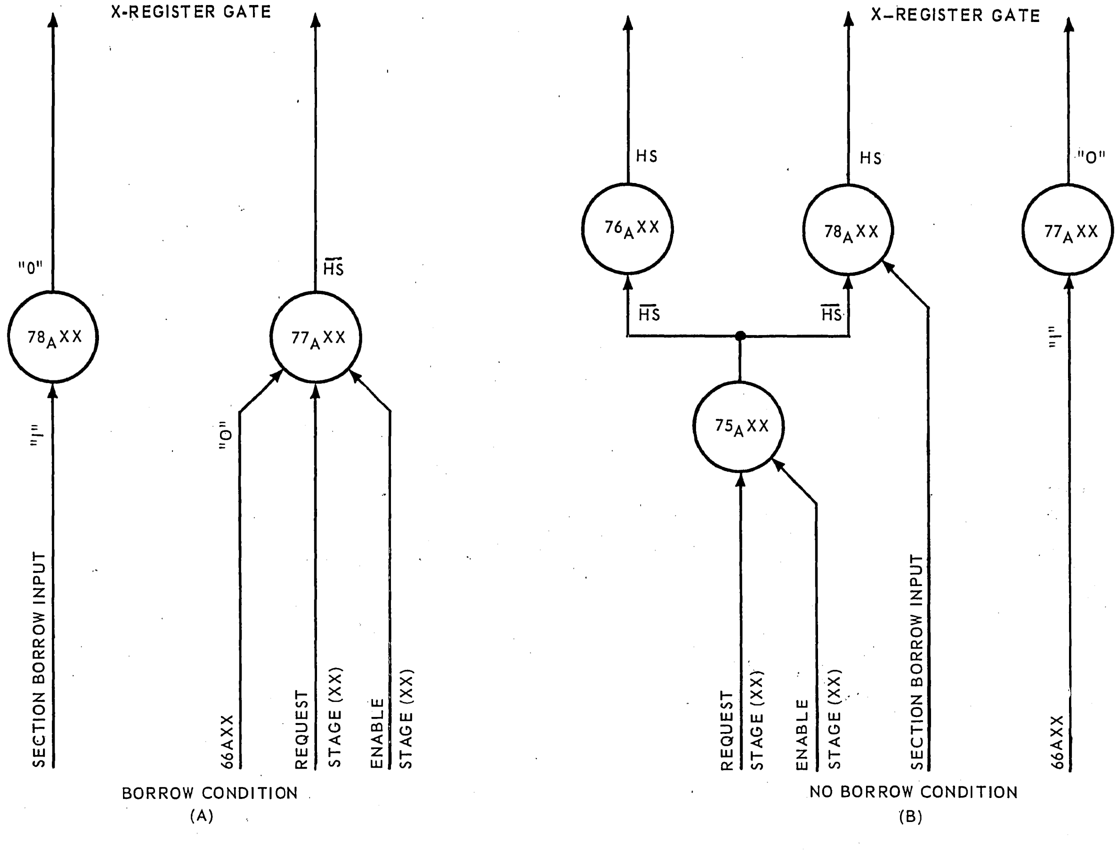
The six separate output status signals (from stages 65AXX thru 60AXX) are combined in a single circuit, 66AXX, which generates an over-all status signal. If all six inputs are "0's", the output from 66AXX is a "1". This indicates no intersection borrow request. If any one or all input lines are "1's", the output from 66AXX is a "0". This indicates an intersection borrow request that has not been satisfied. The "0" output from 66AXX is a signal to each stage in section (XX) indicating an intersection borrow request. A "1" output indicates no intersection borrow request.

Intrasection And Output Circuits

The intrasection and output circuits of each section are alike, stage-for-stage. Except for the order of the bit each stage handles, each stage functions like its respective stage in any other section. Since each stage communicates with every higher-order stage in a section, higher-order stages receive the borrow requests from lower-order stages. Any intersection borrow request not satisfied in a lower-order stage is indicated as such to the next higher-order stage. The output from each stage is the result of the half-subtract process (HS) if no borrow requests to that stage exist or the complement of the half-subtract process ($\overline{\text{HS}}$) if a borrow request to that stage does exist. The outputs, HS or $\overline{\text{HS}}$, are gated into the X-register.

Stage (XX): First Stage in a Section

The intrasection and output circuits (fig. 11-11) of stage (XX) in any section (stage XX not shown) have inputs from the section borrow input circuit (fig. 11-10) and the input circuits of the Adder (fig. 11-6). The section borrow input signal is a composite signal of the six outputs from 65AXX through 60AX via 66AXX (fig. 11-10). The signals from the input circuits (fig. 11-6) are a request and an enable signal. As far as stage (XX), the first stage in any section, is concerned, there is a possibility of two input conditions: (1) borrow condition, and (2) no-borrow condition.



124.145

Figure 11-11.—Simplified schematic of stage (XX).

The borrow request to stage (XX) is always an intersection borrow request. This is indicated by a "1" on the section borrow input line into 78AXX and a "0" input to 77AXX from 66AXX. The output from 78AXX (fig. 11-11A) is forced to a "0" by the "1" input from the section borrow input circuits, regardless of any other inputs. The request and enable inputs to 77AXX, along with the enable "0" from 66AXX, cause the output from 77AXX to be the complement of the half-subtract result ($\overline{HS}$). This output (from 77AXX) is applied to the X-register.

A no-borrow condition (no-intersection borrow request) is indicated by a "1" input to 77AXX (fig. 11-11B) from 66AXX and a "0" input to 78AXX from the section borrow input circuits. Under these conditions, the output from 77AXX is forced to a "0" by the "1" input from 66AXX, regardless of any other inputs.

As has been explained previously, the output from 75AXX is always the complement of the half-subtract result ($\overline{HS}$), and the output from 76AXX is always the half-subtract result (HS). The $\overline{HS}$ input to 78AXX, along with the enable "0" from the section borrow input, causes the output from 78AXX to be the half-subtract result (HS). In the explanation, no reference has been made to the logical signals on the request and enable lines of stage (XX) other than their uses in the half-subtract process. Whether stage (XX) can satisfy an intersection borrow request or not, or whether it is generating a borrow request of its own or not, is indicated by the Request and Enable signals. However, the outputs, as explained, remain the same under any of the aforementioned conditions. The effect of the Request and Enable signals of stage (XX) is covered in the explanation of stage (XX + 1).

Stage (XX + 1): Second Stage in a Section

Since there is a communication between stage (XX + 1) and stage (XX), and between stage (XX + 1) and preceding sections, additional circuits are required in this stage for proper functional and logical operation. Because any borrow request signal to stage (XX + 1) may be intrasection, intersection, or both, the analysis of stage (XX + 1) with regard to the borrow and no-borrow conditions for each case is covered individually.

Intersection Communication with Stage (XX)

An intersection borrow request to stage (XX + 1) is indicated (fig. 11-12) by a "1" input

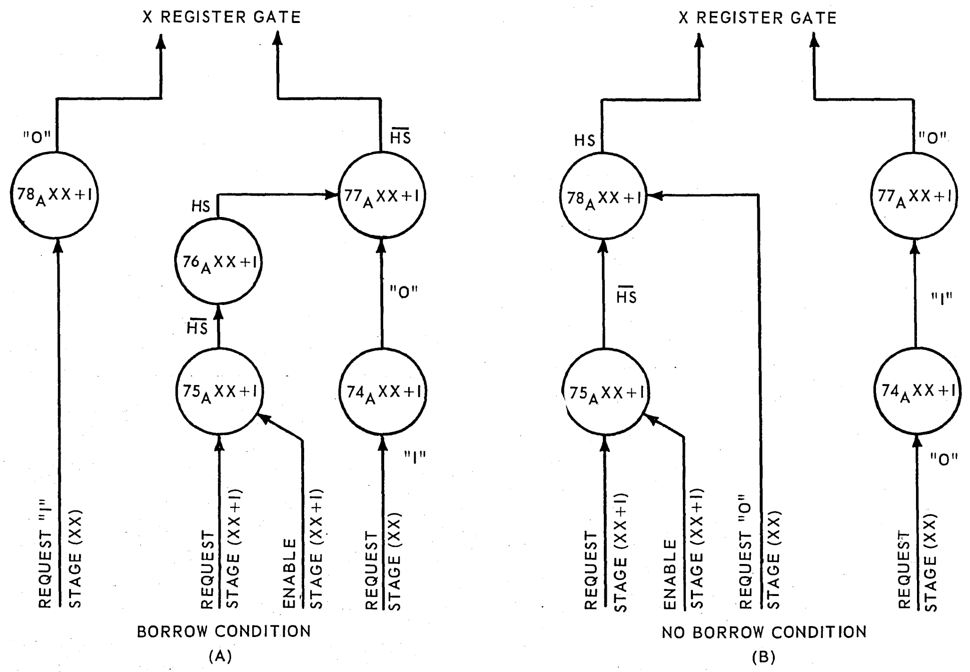
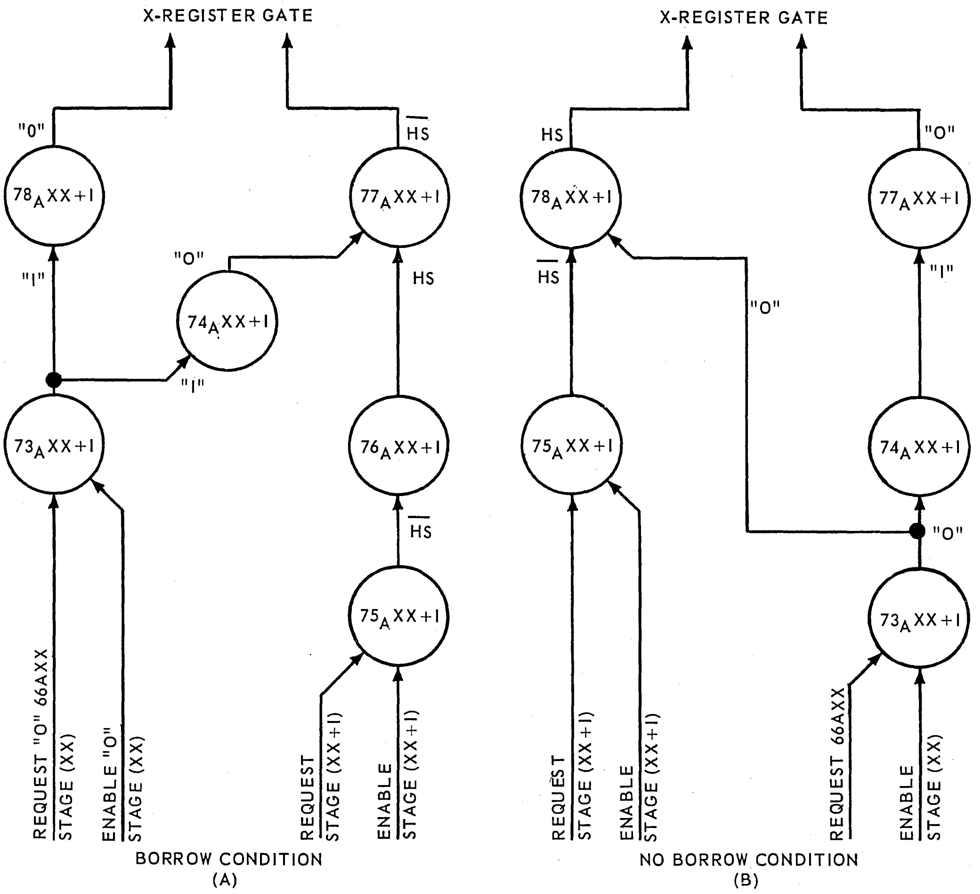


Figure 11-12.—Simplified schematic of stage (XX+1)—intrasection communications with stage (XX). 124.146

on the request line from stage (XX). With the "1" input, the outputs from 78A (XX + 1) and 74A (XX + 1) are forced to "0", regardless of any other inputs. The inputs on the enable and request lines from the input circuits of stage (XX + 1) are combined by 75A (XX + 1) to give an output that is the complement of the half-subtract result ($\overline{HS}$). This output in turn is inverted by 76A (XX + 1) to give an input to 77A (XX + 1) that is the half-subtract result ($\overline{HS}$). Then, the output from 77A (XX + 1) is $\overline{HS}$ with the HS input from 76A (XX + 1) and the enable "0" input from 74A (XX + 1). A no-borrow condition, or no-borrow request to stage (XX + 1) from stage (XX) is indicated by a "0" input on the request line from stage (XX). This "0" input to 74A (XX + 1) is inverted to a "1" output that forces the output from 77A (XX + 1) to a "0". The inputs on the request and enable lines from stage (XX + 1) are combined to again give $\overline{HS}$ at the output of 75A (XX + 1). The output from 78A (XX + 1) is HS with the $\overline{HS}$ input from 75A (XX + 1) and the enable input "0" from stage (XX).

Intersection Communication Through Stage (XX)

An intersection borrow request to stage (XX + 1) through stage (XX) is indicated (fig. 11-13) by a "0" input to 73A (XX + 1) from the section borrow input circuit, 66AXX, and by a "0" input on the enable line from stage (XX). In effect, there is an intersection borrow request indicated by the "0" input from 66AXX that cannot be satisfied in stage (XX) indicated by the "0" enable input from stage (XX). With "0" inputs, the "1" output from 73A (XX + 1) forces



124.147

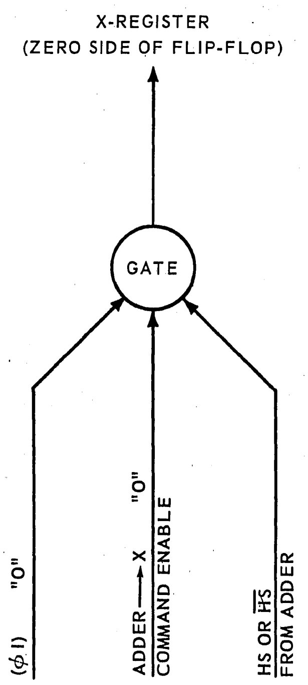
Figure 11-13.—Simplified schematic of stage (XX+1)—intersection communication through stage (XX).

the output from 78A (XX + 1) to "0". The 75A (XX + 1) and 76A (XX + 1) circuits again function so that the input to 77A (XX + 1) from 76A (XX + 1) is HS. The inputs to 77A (XX + 1) are HS and the enable "0" from 74A (XX + 1) to give the resultant output, $\overline{HS}$. A no-borrow condition, no intersection borrow request to stage (XX + 1) through stage (XX), is indicated by a "0" output from 73A (XX + 1). This indicates that either one or both inputs to 73A (XX + 1) are "1's". If there is a "1" input from 66AXX, there is no intersection borrow request to stage (XX + 1). If there is a "1" input on the enable line from stage (XX), any intersection borrow request has been satisfied by stage (XX). The "0" output from 73A (XX + 1) is inverted by 74A (XX + 1) to a "1" that forces the output of 77A (XX + 1) to a "0". The $\overline{HS}$ output from 75A (XX + 1) and the enable "0" from 73A (XX + 1) results in the HS output from 78A (XX + 1). At any time both an intrasection and an intersection-borrow request can be indicated to stage (XX + 1), or no-borrow requests of either type may be indicated. However, the outputs from 78A (XX + 1) and 77A (XX + 1) are the same as for the individual cases. The output from 78A (XX + 1) is HS for any no-borrow condition and "0" for any borrow condition. The output from 77A (XX + 1) is "0" for any no-borrow condition and $\overline{HS}$ for any borrow condition. The above conditions follow along with the solution of the half-subtract problem, wherein the HS is repeated when no borrow is needed from a bit position and complemented when a borrow is needed from a bit position.

Adder Output Gate

The output from the 30 stages of the Adder are gated into the corresponding stages in the X-register. Since the X-register is cleared to "1's" and the Adder output is gated into the zero side of the X-register flip-flop, actual transmission takes place only when HS or $\overline{HS}$ is a "0". Note in figure 11-14 that transmission takes place at phase 1 and on the Adder ⟶ X command enable (both "0's"). Then, for coincidence and for an output of "1" to change the state of the flip-flop from "1" to "0", HS or $\overline{HS}$ must be a "0". In effect, if HS or $\overline{HS}$ is a "1", the output "0" from the gate does not change the state of the flip-flop from "1" to "0".

All Adder operations, as described, are simultaneous. For practical explanations these operations must necessarily be analyzed separately; but for a true picture, they must be visualized as being accomplished together.



124.148
Figure 11-14.—Simplified schematic of adder output gate.

BASIC OPERATIONS

The basic operations performed in the arithmetic section are addition, subtraction, multiplication, division, and shifting. The following paragraphs present the principles involved and the manner in which the operations are performed. Any of the arithmetic processes can be done by a specific instruction. Addition and/or subtraction can be done separately or in combination, by several instructions. All addition and subtraction operations are done in the Adder. Since multiplication and division are essentially repeated additions or subtractions, they also make use of the Adder. The registers of the arithmetic section store the numbers being manipulated and provide temporary storage for the results.

ADDITION

The addition process makes use of the A-, D-, and X-registers and the Adder. The registers store and transmit data, and the Adder performs the arithmetic operation. The Adder is a subtractive device, but the following equation shows that addition can be performed by a subtractive operation, providing the sum does not exceed the modulus (maximum storage number of the registers involved).

$$A + Y = A \underset{\text{Subtractive operation of Adder}}{\overset{\text{Apparent complement in Adder}}{-}} D'$$

The contents of the A and D registers (see figure 11-1) are always applied to the Adder. The augend is in A (the A-register) and the addend is in D. The operation of the Adder is such that the contents of D appear to be complemented and subtracted from A, thus performing an addition.

A second example of the method of addition used in the computer is shown below.

Manual addition (using the subtractive process)

	00101100	44_{10}
	11101000	23_{10} complement
HS	11000100	
B	10000001	
HS	01000101	
B	00000010	Note: HS = Half Subtract
HS	01000111	B = Borrow
B	00000100	
	01000011	67_{10} final result

Computer Addition

A -	00101100	
D'-	11101000	
	11000100	HS
	10000111	ALL BORROWS
	01000011	

Note that in the Adder, D appears to be complemented and subtracted from A. Note that the output from the Adder is transmitted to X (under command control Adder ⟶ X, see fig. 11-1) A is cleared, and X may be transmitted to A.

The augend is initially placed in the A-register, and the addend in the D-register. Registers A and D furnish inputs to the Adder which adds the two numbers. Then the commands (command lines not shown) are issued to clear X, Adder ⟶ X (gate the Adder contents to X) and for most instructions clear A, and gate X ⟶ A. These commands are phase controlled so that the processes take place in the order given. The sum of the two numbers is in the A-register.

The result of the addition is utilized only if the command, Adder ⟶ X, is generated. The Adder always combines the contents of the A- and D- registers, but the Adder ⟶ X command is necessary to circulate the result. When f = 20, the Adder ⟶ X command is followed by the X ⟶ A command. This places the sum in the A register.

SUBTRACTION

Basically the operation of subtraction is the same as addition. The only difference is in the manipulation of the number contained in the D-register. The A-register holds the minuend and the D-register holds the complement of the subtrahend (see example below). The following example shows that the subtraction is performed, in the Adder, by subtracting the complement of the complemented subtrahend.

Y =	00010111	23_{10}
A-Register	00101100	44_{10}
D-Register	00010111	23_{10}
	11101000	1's comp. of D = D'
	00010111	1's comp. of the 1's comp. of D (D') which is used as the subtrahend in the adder.

Thus,

Minuend	00101100 =	44_{10} = A
Subtrahend	00010111 =	23_{10} = (D')
HS	00111011	
B	00100110	
HS	00011101	
B	00001000	
	00010101 =	21_{10} final result

The difference of A - Y is transmitted (fig. 11-1) from the Adder to X; A is cleared and X transmitted to A, thus circulating the results of the subtraction. The answer is placed in the A-register, as in addition.

In either addition or subtraction, the numbers to be manipulated are in A and D. The contents of A and D are applied to the Adder. The output of the Adder is gated into the X-register, A is cleared, and X is transmitted to A. Either operation can be performed by a specific instruction (f = 20 or 21) or as a part of another instruction.

MULTIPLICATION

Multiplication performed in the computer makes use of the shifting capability of the arithmetic section and the Adder$\longrightarrow$X operation. The operation of multiplication performed by instruction f = 22. Initially, the multiplier is placed in Q (fig. 11-1) and the multiplicand is placed in D. The double-length product is formed in AQ with A holding the most significant or higher-order bits.

An examination of ordinary pencil-and-paper multiplication will provide a basis for understanding the computer operation of multiplication. In the pencil-and-paper method, intermediate or partial products are formed by using the multiplicand and successive digits of the multiplier, beginning with the lower-order digits. These partial products, when set down, are shifted relative to one another so that their digits fall in the columns of proper order. They are then added to form the final product. See the following example:

Multiplicand		15	
Multiplier		X14	
	Partial $\longrightarrow$	60	Shift to place
	Products		$\longrightarrow$digits in pro-
		15	per order
	Product	210	columns

Working through this example shows that this pencil-and-paper method relies on previous knowledge of the decimal multiplication and addition tables, proper attention to carriers, and the order of partial product digits as well as the concepts of simple multiplication and addition.

Binary Multiplication

In binary multiplication no carries can occur because the results of binary multiplication are always single digits; they are either "0" or "1" (and "1" only when both multiplier and multiplcand are "1"). The previous example (in which decimal numbers are used) is worked out in binary multiplication as follows:

	1 1 1 1	15_{10}
	1 1 1 0	14_{10}
	(1)0 0 0 0	
	(1)	
Partial	(1)1 1 1 1	
	(1)	
Products	(1)1 1 1 1	
	(1)	
(1) - represents carry	(1)1 1 1 1	
to that order from		
next lower order.	1 1 0 1 0 0 1 0	210_{10}

Notice that in binary multiplication the partial products are either the multiplicand (when the current multiplier digit is "1") or zero (when the current multiplier digit is "0"). Notice also that the addition of several binary numbers at once (particularly where several 1 digits are considered) can be confusing and complex. This complexity would require corresponding complexity of design if this method were used in the computer.

Computer Multiplication

During computer multiplication, each partial product is accumulated in a running total as it is formed, so that only two binary numbers are being added at one time. To insure that each partial product is added with proper regard for the order of its digits, the running total is shifted to the right each time the partial is added. When the multiplier bit is "0", the partial product is also zero; thus, this step requires only a shift. In other words, multiplication by a "1" consists of adding the multiplicand and shifting. Multiplication by a "0" consists of merely shifting. This process, using the previous example, is shown below. (It is assumed in the example that both numbers are positive and no sign bit is used.)

		Multiplier Digits
Multiplicand	1 1 1 1	
Partial Product	0 0 0 0	0
(No Add)	- - - - -	
Right Shift	0 0 0 0 0	
Partial Product	1 1 1 1	1
Sum	1 1 1 1 0	
Right Shift	0 1 1 1 1 0	
Partial Product	1 1 1 1	1
Sum	1 0 1 1 0 1 0	
Right Shift	1 0 1 1 0 1 0	
Partial Product	1 1 1 1	1
Sum	1 1 0 1 0 0 1 0	
Right Shift	1 1 0 1 0 0 1 0	

Multiplication in the computer is accomplished in four steps: (1) initial sign correction, (2) multiply, (3) double-length product detection, and (4) final sign correction.

Initial Sign Correction

The process of multiplication can be done by the computer only if both multiplier and multiplicand are positive quantities. However, there is the possibility that either or both of the numbers involved in the multiplication process will be negative.

In order to understand why the computer performs initial sign correction, a review of the multiplication of signed numbers is necessary. A negative number multiplied by a positive number has a negative product, but the product of two negative numbers, or two positive numbers, is a positive number. In the computer, either the multiplier (held in Q) or the multiplicand (held in D), or both, can be negative. If either or both of these is a negative quantity, it must be changed to a positive quantity.

Negative Multiplicand

If the multiplicand (D) is negative, an inverter in the C sequence (not shown) senses this condition and initiates an operation called complement 2 (circuit not shown) for the purpose of correcting the sign of the final product. The output from another inverter in the C sequence causes the commands Clear X and transmit $D \longrightarrow X$. Subsequently, the commands are issued to clear D and transmit $X' \longrightarrow D$; where X' is the 1's complement of the original content of the D-register.

The computer has now changed the negative number (originally in D) to a positive number and has included (in the complement 2 operation) provisions for making final sign corrections. The product will now be a positive number, but since the original problem contained a positive and a negative number, the product must be complemented so that it is a negative number. This final sign correction of the product occurs after the completion of the multiply step.

Negative Multiplier

A negative multiplier (in the Q-register) is sensed by an inverter ($^{63}N^{05}$) in the B sequence (see figure 10-15). The $^{63}N^{05}$ output enables the complement 1 operation (circuit not shown) for complementing Q and for later use in the final sign correction. A flip-flop (not shown) enables a complement AQ sequence. This operation normally complements the contents of the A- and Q-registers, for final sign correction after the multiply (or divide) operation. However, in the C sequence (at the same time that the multiply operation is initiated) an inverter (not shown) causes the A-register to be cleared. The final result is that only (Q) has been complemented. The final product is corrected for proper sign after the multiply step.

Negative Multiplier and Multiplicand

If both numbers are negative, the two previously described sequences (for negative multiplicand and negative multiplier) occur. This results in the complementing of both numbers and the activation of both complementing operations (1 and 2). The two positive numbers are now multiplied and yield a positive product. Since the product of two negative numbers is a positive number, no final sign correction is necessary. The final sign correction (complement AQ) is disabled when the complement operations 1 and 2 are both set. The complement 1 and 2 operations were both set during the initial sign correction (for correcting the sign of the multiplicand and multiplier separately) if the multiplicand and multiplier were both negative.

Positive Multiplier and Multiplicand

If both numbers are positive, there is no need for initial sign correction. The product will be positive, thus there is no need for final sign correction. The final sign correction (Complement AQ) is disabled when Complements 1 and 2 are not set, (as is the case when multiplicand and multiplier are positive). Table 11-4 shows the four conditions that can exist between Q and D and the need for initial and/or final sign correction.

Table 11-4.—Sign Correction For Multiplication.

Initial Sign Correction		Comp - 1	Comp - 2	Final Sign Correction
D Neg.	YES (C Seq)		SET	YES
Q Pos.	NO			
Q Neg.	YES (B Seq)	SET		YES
D Pos.	NO			
Q neg.	YES (B Seq)	SET		NO
D Neg.	YES (C Seq)		SET	
Q Pos.	NO			NO
D Pos.	NO			

Multiply Step

The A, B, and C sequences are involved in interpretation and initial sign correction (correction of sign of either register separately) for a multiply instruction if necessary. (See figure 5-9 for basic principles of computer multiplication.)

The multiply step is performed systematically following a specific set of conditions. In the multiplication process a right shift is accomplished for each bit of the multiplier (in the Q-register); and there may or may not be an addition of the multiplicand to the partial result (see figure 5-9C), depending on the current multiplier bit (the current multiplier bit, Q00, is the lowest order bit in the Q register). If the current multiplier bit is "0", there is no addition, only a right shift. If the current multiplier bit is a "1", there is an addition of the multiplicand to the previously obtained partial result and then a right shift.

When the lower five bits of the multiplier (Q04-00) are zero, there will be no additions for the next five multiplier bits. Rather than make five one-place shifts there are circuits (not shown) to sense the condition wherein a shift of five-shift positions is necessary. The four-place right shift is executed only when the sensing circuit input conditions are satisfied. The one-place shift is unconditional and consequently the computer accomplishes a shift of the Q-register contents right by five-bit positions.

In a multiply instruction, K^{15} (fig. 11-4) is unconditionally set when the multiply sequence is initiated (C sequence). The commands that affect the K-registers for multiply are the same as for divide. (The divide operation is treated later.) The only difference is in the sum of K2 and K0. In the multiply step the circuits which sense the lower five bits of Q are operative. If the lower five bits equal zero, again it means there can be five shifts with no additions needed. When Q04-00 (the lower five bit in the multiplier which is in the Q register) equals zero, there is a four-place right shift and a one-place right shift and the shift count must be advanced by five. This is accomplished by setting K^{60} when Q04-00 equals zero, and when K0 is added to K2 the result is as follows:

K2	00000	(Q04-00=0)
K0	101	(if Q04-00=0) (K^{15}=1, K^{30}=0, K^{60}=1)
K adder$\longrightarrow$K1	00101	(Represents shifts to be done)

Clear K2, K1$\longrightarrow$K2

The overall count is increased by five when K^{60} is set (K^{60} causes a four-place right shift and K^{15} remains set through the entire operation and causes a one-place shift. After multiplication by all 1 bits in the multiplier have been executed and only 0's remain in the multiplier, the shift operations involving K^{60} (just described) continue until the shift count in K equals 30. At this time an inverter (not shown) has an output that disables and stops the multiply sequence.

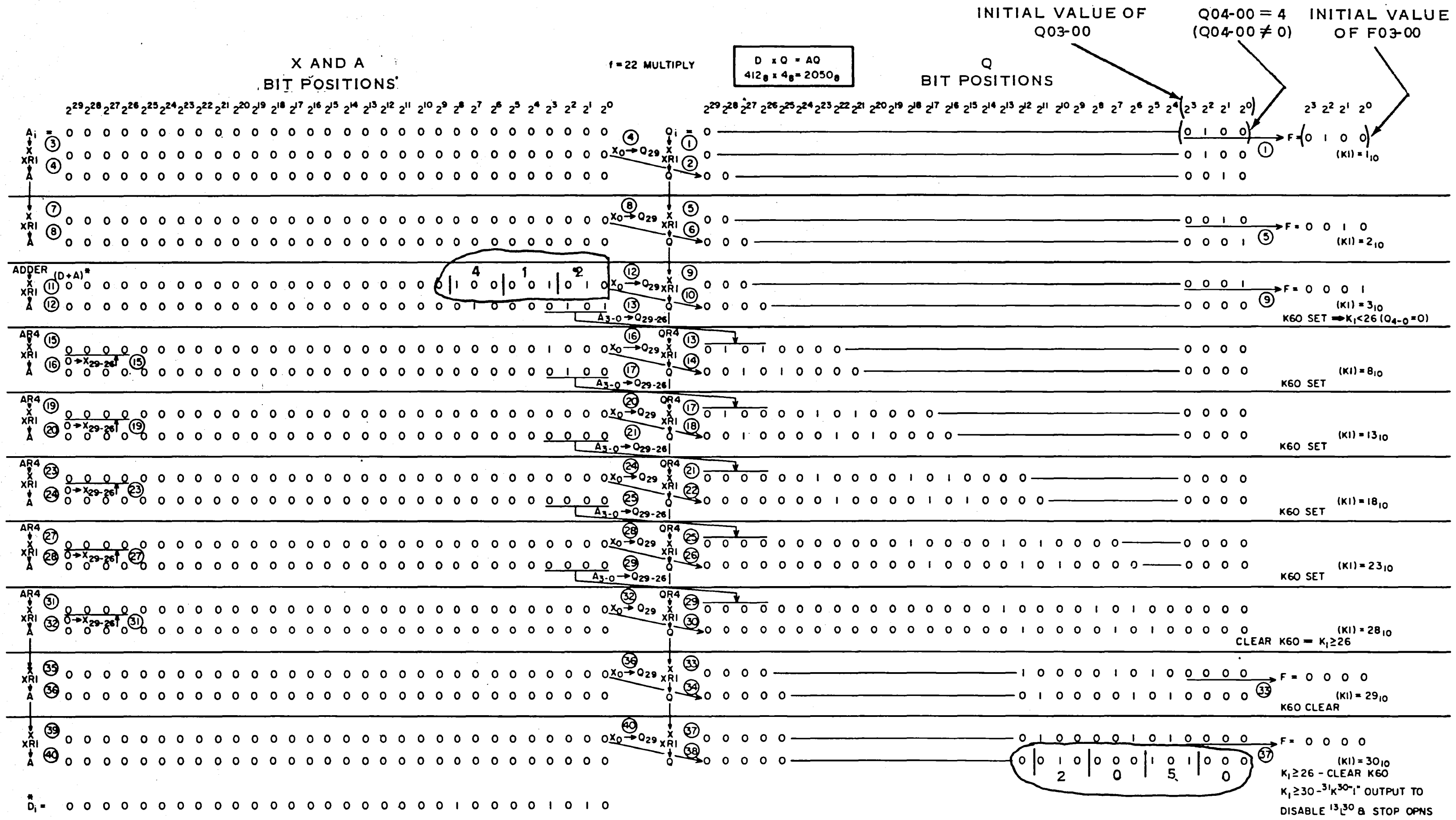
When the shift count reaches 26, only four shifts remain. For this reason it is necessary to disable the circuits that can initiate a right shift of four places. This is accomplished by inverters (not shown) which sense the shift count. When the shift count reaches 26, these inverters disable the circuit which cause K^{60} to be set thus preventing K^{60} from being set, and the remaining four shifts occur one at a time.

Shift Count

The shift count is held in the K1 register (fig. 11-4) and is advanced by one for each shift. Initially the K^{15} stage of K0 is set during the C sequence, and at this time K2 = 0 and K1 = 0 (all registers previously cleared). In going through the first multiply step the computer transmits K adder$\longrightarrow$K1 (this is the sum of K2 and K0 transmitted to K1). This makes K1 = 1 after the first multiply step. Then K1$\longrightarrow$K2 and clear K1 occurs. The next time through the multiply step, K2(1) is again added to K0(1) and is transmitted to K1; now K1 equals 2. This procedure continues (accomplishing shifts of 1 or 5 as discussed) until K1 = 30. When K1 = 30, the multiply step operation is stopped.

Shifting and Adding

Figure 11-15 illustrates data flow in the A-, X-, Q-, and F-registers for the multiply example 412_8 x 4_8 = 2050_8. The circled number by each data transmission shows sequence of events in the overall multiply operation. (Numbers are omitted where instructions are repeated.)



1. DURING A MULTIPLY OPERATION, D (MULTIPLICAND) MUST BE POSITIVE. TO INSURE THIS:
   IF D_i IS POSITIVE – THE "C" SEQUENCE WILL: D→X, CLEAR D, X→D
   IF D_i IS NEGATIVE – THE "C" SEQUENCE WILL: D→X, CLEAR D, X'→D
2. SKIP EVALUATION (DOUBLE-LENGTH PRODUCT DETECTION) IS DONE PRIOR TO FINAL SIGN CORRECTION.
3. AT THE END OF THE MULTIPLY SEQUENCE, THE COMPLEMENT AQ SEQUENCE IS INITIATED. IF COMPLEMENT 1 OR 2 FLIP FLOP IS SET, (NOT BOTH) AQ WILL BE COMPLEMENTED (MULTIPLIER OR MULTIPLICAND NEGATIVE).

124.149

Figure 11-15.—f = 22, multiply.

Before the actual multiplication process begins, the registers contain the following data:

A = 0 (bit positions A29 thru A00)
D = Multiplicand 412_8 (100 001 010)
X = 0 (bit positions X29 thru X00)
Q = Multiplier (000 100)
F = 00000 (bit positions F04-F00)

Before the specific example of multiplication using figure 11-5, consider the following general discussion.

In a multiply operation the contents of Q are first transferred to X (not shown) in preparation for right-shifting Q. (The Q-register content, the multiplier, is right-shifted after consideration of each bit so that the current multiplier bit is always in the Q^{00} bit position.) The X-register content (the original content of Q) is right-shifted (generally one bit position) and transferred back to Q. During this process, the lower four bits of the initial value of Q (Q^{03}-Q^{00}) are placed in the F-register bit position F03-F00. The F-register, if f00=1, in turn, initiates the addition (D + A). This value represents the product of two numbers only if the multiplier is one. In all other cases where the multiplier is greater than one, the first addition in A generates a partial sum.

If the lowest order bit if F (F00) is 0, no addition of D + A takes place. If F00 = 1 the multiplicand is added to the A-register content.

The A-register content (which accumulates the product) is also right-shifted after each consideration of the current multiplier bit so that as the multiplier is shifted out of Q, the product is shifted into Q. The right shift of A is accomplished by transferring A$\longrightarrow$X and later right-shifting X and transferring to A. In the example of fig. 11-15; the following events occur:

1. Q$\longrightarrow$X, Q03-00$\longrightarrow$F03-00 (A04-00 $\neq$ 0)

Note: If the lowest order bit in Q (Q^{00}) were 1, the transfer of the lower four bits in Q to the F-register would result in a 2 bit in the F00-bit position. The F-register would therefore cause the multiplicand to be added to the A-register content (all bits in A initially 0's) before the second highest bit in 0 is considered.

2. XR1$\longrightarrow$Q, K adder$\longrightarrow$K1, (K1 = 1)
(Note — not shown in illustration — see fig. 11-4)
3. A$\longrightarrow$X (The current multiplier bit in F00 is "0", thus no addition is done.)
4. XR1$\longrightarrow$A, X00$\longrightarrow$Q29 (places A00 in Q29)

The computer has now shifted AQ right one place and it recycles to do the same steps (steps 1 thru 4) because Q04-00 $\neq$ 0, and the lowest order bit in F (shifted right 1 bit position) is 0. At the end of the second cycle, AQ has been shifted right two places and K_1 = 2. At the beginning of the third cycle (step 9) the current multiplier bit (F00) equals "1" which means add the contents of the D-register (the multiplicand) to A.

9. Q$\longrightarrow$X, Q03-00$\longrightarrow$F03-00 (Q04-00 $\neq$ 0)
10. XR1$\longrightarrow$Q, $K_2$$\longrightarrow$$K_1$, ($K_1$ = 3)
11. Adder$\longrightarrow$X (current multiplier bit (F00) is "1" so add A to D and transmit to X)
12. XR1$\longrightarrow$A, X00$\longrightarrow$Q29 (Q04-00 = 0 thus set K60)
13. QR4$\longrightarrow$X, A03-00$\longrightarrow$X29-26
14. XR1$\longrightarrow$Q, K Adder$\longrightarrow$$K_1$ (K_1 = 8)
15. AR4$\longrightarrow$X, 0's$\longrightarrow$X29-26
16. XR1$\longrightarrow$A, X00$\longrightarrow$Q29

The example shown above illustrates that the computer has completed the necessary multiplication and must now finish the 30 shifts. (The entire 30 bits of the multiplier in Q must be considered singly or in groups of four and one for each multiply operation to ensure proper computer action for each multiply bit in Q. This action is indicated in steps 17-4Q.

The A- and Q- registers combined (AQ) is shifted in increments of one or four and one until K_1 = 30. When the shift count reaches 26, the remaining shifts must be done one at a time. This is done because the four shift (initiated by K60) is followed by the unconditional one shift caused by K15. This could result in a total shift of more than 30 bit positions and hence an incorrect product. An over-shift condition is prevented by disabling circuits (not shown) when the shift count is greater than 25. As long as the shift count is less than, or equal to 25, the setting of K60 is dependent on the lower five bits of Q. When the count is 25 or greater the disabling circuits prevent the setting of K60. When K_1 = 30 the multiply step stops and the product is found in AQ.

Double-Length Product Detection

If the quantities in Q and D are sufficiently small, the product may be entirely in the Q register. However, if there are large numbers involved, there may be a double-length product in AQ with A holding the higher-order bits. It is desirable to have an indication at the completion of the instruction as to whether the product is single- or double-length, because the two cases must necessarily use different procedures for product storage.

Double-length product detection is done prior to final sign correction. The size of the product is determined in the following manner. The multiply instruction can be programmed with j = 2 (skip the next instruction if Q is positive) and a skip evaluation then checks to see if Q29 = 0. If Q29 = 0, skip the next instruction and add 0 to A and check for A = 0. If both A and Q29 are "0", the entire product and sign bit are in Q and the computer proceeds, assuming that the answer is a single-length product. If, however, A does not equal 0, the computer treats AQ as the double-length product. Also, if Q29 = 1 the computer retains A because in this case, Q29 is not necessarily the sign, but it may be a significant digit of the product. Double-length product detection is accomplished by the programmer who uses different j values in a series of instructions following a multiply instruction. The following is a series of instructions which could accomplish double-length product detection. (See table at the end of chapter 10 for interpretation of designator values.)

```
22 200 -----  Multiply, skip if Q29 = 0
60 100 (----) Jump to double-length storage
                  routine (not shown)
20 400 00000  Add 0 to A, skip if A = 0
60 100 (---)  Jump to double-length storage
                  routine
14 030 -----  Store Q (single-length product
                  storage)
```

Double-length product detection can be achieved by using a j of 3 (skip the next instruction if Q is negative) and a routine which functions accordingly.

DIVISION

The operation of division is performed by the instruction: f = 23 (Divide). In division, the 60-bit dividend is stored initially in AQ, the divisor in D, the quotient is formed in Q and the remainder, if any, is in A. The X-register is used in the execution of the left shift and for complementing. Division is accomplished by making use of the shifting properties of the arithmetic section and by the subtraction ability of the Adder. A general understanding of the basic principle of the computer method of division can be obtained by first reviewing the pencil-and-paper method of division.

In division, the quotient indicates how many times the divisor is contained in the dividend; i.e., how many times the divisor can be subtracted from the dividend. The quotient is formed one bit at a time by a series of partial divisions as follows:

```
     251
  2 /503
     -2 }  subtracting 2 twice from partial
     -2 }  dividend
     --
     10    —Partial Dividend
     -2 \
     -2 |
     -2 >  subtracting 2 five times from
     -2 |  partial dividend
     -2 /
     --
     03    —Partial Dividend
     -2}   subtracting 2 once from partial
     --    dividend
      1    remainder
```

The most significant digit of the quotient, 2, is obtained by counting the number of times the divisor can be subtracted from the first partial dividend, 5. The second partial dividend, 10, is formed from the remainder of the previous partial division and the second digit of the initial dividend which is taken as the lowest digit. Since two is contained in 10 five times, the second digit of the quotient is a 5. This procedure is continued until each digit of the initial dividend has been used in a partial dividend.

Computer Division

Computer division is similar to pencil-and-paper division. In certain aspects, however, the two methods differ. The major differences are described in the following paragraphs.

Numbers used in computer division are expressed in binary rather than decimal notation (as is commonly used with the pencil-and-paper method). Where binary numbers are used, the divisor can be subtracted only once, if at all, from each partial dividend. In division of decimal numbers, the divisor can be subtracted as many as nine times from the partial dividend. Thus the use of binary numbers simplifies the process considerably from this standpoint.

Second, in division by the pencil-and-paper method, there is no fixed limit to the size of the divisor, dividend, or quotient. If one wishes to spend the time, he can divide a dividend of 100 digits by a divisor of two digits and get a quotient of perhaps 50 digits. In the computer, however, there is a fixed limit to the size of the three numbers. The divisor cannot exceed 30 bits (29 data bits and the sign bit) and the dividend cannot exceed 60 bits (59 data bits and the sign bit). Furthermore, the relative sizes of the divisor and dividend must be such that there are no more than 30 bits (29 data bits and the sign bit) in the quotient. For example, in the computer a dividend of 30 or more significant digits cannot be divided by a divisor of one digit, because in this case the quotient would exceed the maximum.

A third point to be noted is the stair-step arrangement of the pencil-and-paper example shown above. There is a step corresponding to each partial division. The divisor is shifted one place to the right before each subtraction. The same effect is obtained by shifting the dividend one place to the left prior to each subtraction.

Finally, it should be noted that there is a partial dividend for each digit in the quotient. Regardless of the actual size of the numbers, the computer division procedure always assumes and operates as though they were the maximum size, namely 30 bits for the divisor and quotient, and 60 bits for the dividend. Briefly, this means that the zeros to the left of the most significant bit are not disregarded as they are in the pencil-and-paper method. The computer always performs 30 partial divisions in order to form a 30-bit quotient.

Initial Sign Correction

Initial sign correction must be done in division because the dividend AQ must be positive and the divisor (D) must be negative. The divisor (D) must be negative because in division, the Adder performs the subtraction function. Thus, the divisor must be complemented in the computer before it is applied to the Adder.

Initial sign correction for AQ takes place in the B sequence. Circuit $^{63}N^{04}$ (fig. 10-15) senses the sign bit of AQ (A29); if it is negative, Complement 1 is set for final sign correction and the Complement AQ sequence is initiated, making AQ a positive number.

The initial sign correction for D takes place in the C sequence (circuit not shown). If the division, D_i (D initial) is negative, Complement 2 is set, $D_i \longrightarrow X$, Clear D, $X \longrightarrow D_f$ (D final). D remains a negative quantity and Complement 2 is set for final sign correction. If the divisor is positive the C sequence senses this and transmits $D_i \longrightarrow X$, clears D, $X' \longrightarrow D_f$ (where X' is the complement of D_i). The divisor is not a negative quantity but no final sign correction is necessary if the original divisor and dividend were the same sign. Initial sign correction and the setting of Complement 1 and/or Complement 2 is done before the actual divide sequence occurs. When the divide sequence starts, D always holds a negative quantity, AQ is positive, and the condition of Complement 1 and Complement 2 flip-flops (set or clear) depends upon the initial sign of the numbers. Table 11-5 shows the conditions for initial and final sign correction.

Divide Step

The divide operation utilizes the subtractive adder and the A-, Q-, D-, and X-registers. In the Adder, D is apparently complemented and subtracted from A. Therefore, if A is greater than or equal to D', a subtraction is done with no end-around borrow. If A is less than D', an end-around borrow is initiated. The generation of an end-around borrow is used in the divide sequence to indicate that the divisor is larger than the dividend, and prevents executing Adder$\longrightarrow$X (a step which permits the divisor to be subtracted from the dividend). If D' can be subtracted from A (no end-around borrow), a partial division is performed and a "1" is entered in Q^{00} as a digit of the quotient. If D' cannot be subtracted from A (end-around borrow requested), a "0" is entered in Q^{00} indicating that a partial division was attempted, but not completed because the divisor was larger than the partial dividend. AQ (the dividend) is then shifted left one place thus entering a "0" in the lowest bit position of Q.

In the divide sequence the computer senses if A is greater than or equal to D', and if true, a "1" is transferred to Q^{00}. If A is less than D' a "0" is transferred to Q^{00}. AQ is shifted left one place for each subtraction or attempted subtraction. This sense and shift procedure is performed 30 times regardless of the size of the divisor. When the procedure stops, the quotient is in Q, and the remainder is in A.

The shift count is maintained in much the same way as in a multiply instruction. In the C sequence, K^{15} (fig. 11-4) is set, K2 and K1 are cleared. With each shift the following operation occurs: $(K0 + K2) \longrightarrow K1$; $K1 \longrightarrow K2$, clear K1. As in multiplication, this operation results in the shift count increasing by one for each shift. When the shift count equals 30, the divide operation stops.

The divide operation is explained with the aid of figure 11-16. It is assumed that the program has advanced through the A and B sequences into the C sequence and that the divide step has been initiated. Further, it is assumed that all initial sign corrections have been made and that K^{15} is set.

In the first step (not shown by number) of the example, where Ai is less than D' (D is subtracted from Ai in the adder), an end-around borrow is generated. Since there is an end-around borrow, a "1" is not transmitted to the lowest bit position of X in preparation for entrance into Q as a bit in the quotient.

After the attempted subtraction (Ai - D') the command is issued $Ai \longrightarrow X$ (step 1) followed by step 2, $XL1 \longrightarrow A$ (read, left shift X one bit position and transmit results to A). This action shifts Ai left one bit position. After the shifting of A, the commands $Q \longrightarrow X$ (step 3) and $XL1 \longrightarrow Q$ (step 4) are executed. During the left shift Q operation, and if a 1 bit was generated by the Ai - D' operation, this 1 is entered in the lowest bit position of X and transmitted to the lowest bit position of Q where it will represent a significant bit in the final quotient. If the Ai - D' operation does not produce a 1 (which represents the condition in this case where $Ai - D' < 1$) a "0" will be entered in the lowest bit position of X during the $Q \longrightarrow X$ and $XL1 \longrightarrow Q$ process, thus transmitting this "0" to Q where it represents a significant digit in the final quotient. The computer has now completed the first partial division, entered the first digit ("0") in the partial quotient, and shifted AQ left one bit position.

At this time the shift count equals one (K1 = 1), (see figure 11-4). This same series of steps (fig. 11-12) will be done until the shift count equals 26 (K1 = 26) at which time, for the first time, A is greater than, or equal to, D'. Under this condition a slightly different series of steps is followed. In the next series of steps, the computer $A \longrightarrow X$, $XL1 \longrightarrow A$; $Q \longrightarrow X$, $XL1 \longrightarrow Q$; but now since $A = 5_8$ and the divisor $= 5_8$, a subtraction can be performed and a "1" is entered in Q^{00} to indicate this. On the next series of steps, since A = D', there is no end-around borrow indicated in the Adder and $Adder \longrightarrow X$ occurs. $Adder \longrightarrow X$ takes place because A = D' and it performs the partial division and transmits the next partial dividend to X, followed by $XL1 \longrightarrow A$ and $Q \longrightarrow X$, $XL1 \longrightarrow Q$. The shift count now equals 28. There are now two more partial division attempts to be made, and, since A is less than D, these two steps are the same as the first step.

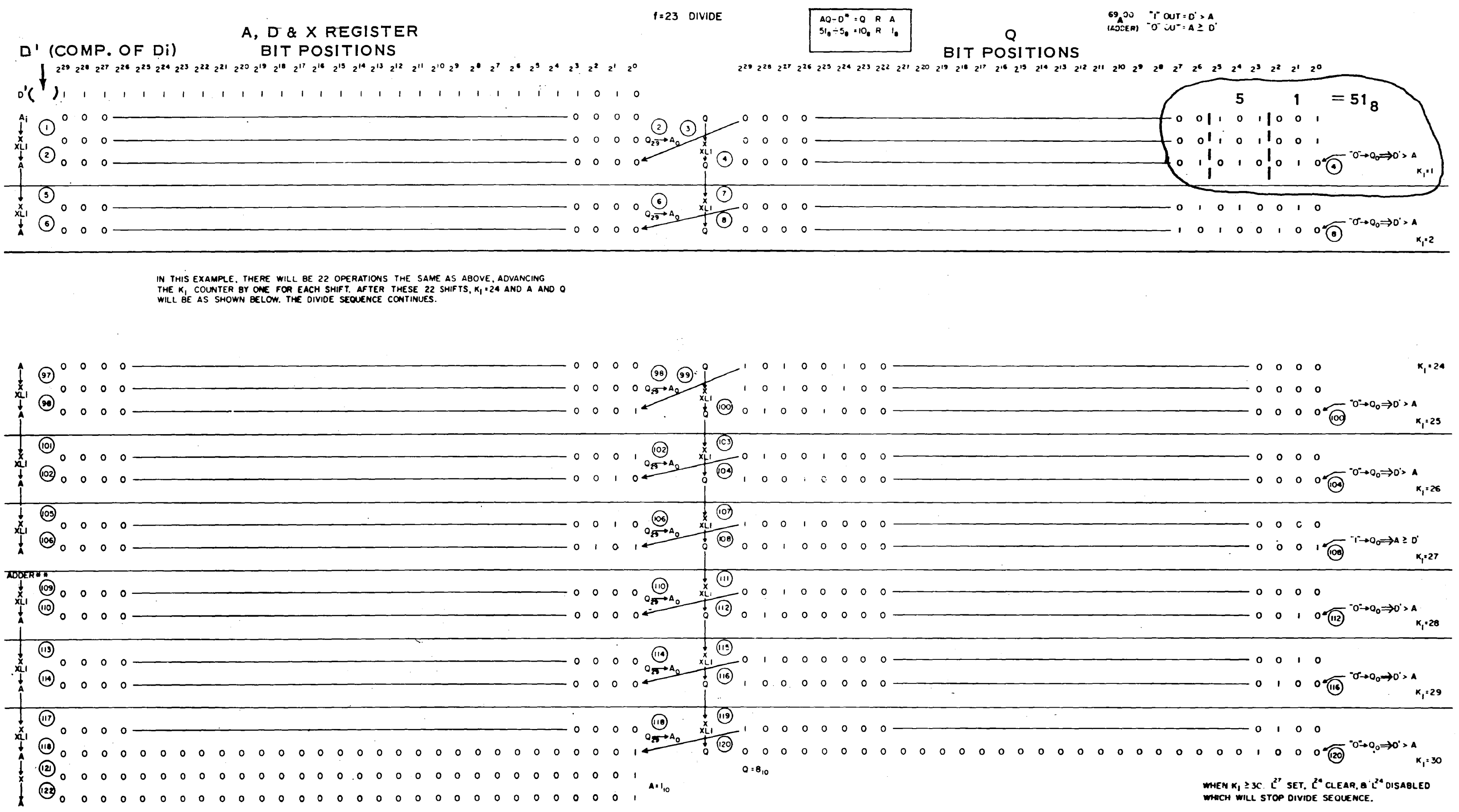
Table 11-5.—Sign Correction For Division.

Initial Sign Correction		Comp - 1	Comp - 2	Final Sign Correction
D_i Neg. AQ Pos.	NO NO		SET	YES
AX Neg. D_i Pos.	YES (B Seq) YES (C Seq)	SET		YES
AQ Neg. D_i Neg.	YES (B Seq) NO	SET	SET	NO
AQ Pos. D_i Pos.	NO YES (C Seq)			NO

When the shift count equals 30, an inverter (not shown) in the K2 register has a "1" output that prevents recycling of the divide step. The divide step has now sensed for A greater than or equal to D', 30 times; shifted AQ left 30 places and the operation is stopped. The result of the division has eight in Q (quotient) and one in A (remainder) which is the correct answer for $41 \div 5$.

Divide Faults

In the computer process of division, there are two possible fault conditions. The first occurs when the divisor is zero and the second occurs when the quotient is more than 30 bits in length including the sign bit. The computer will carry out the division regardless of whether one of these fault conditions exists.



IN THIS EXAMPLE, THERE WILL BE 22 OPERATIONS THE SAME AS ABOVE, ADVANCING THE K_I COUNTER BY ONE FOR EACH SHIFT. AFTER THESE 22 SHIFTS, K_I=24 AND A AND Q WILL BE AS SHOWN BELOW. THE DIVIDE SEQUENCE CONTINUES.

AT THE END OF THE DIVIDE SEQUENCE, THE COMPLEMENT AQ SEQUENCE IS INITIATED. IF COMPLEMENT 1 <u>OR</u> COMPLEMENT 2 FLIP FLOP (NOT BOTH) IS SET, AQ WILL BE COMPLEMENTED (DIVIDEND OR DIVISOR NEGATIVE).

AT THE COMPLETION OF THE DIVIDE SEQUENCE (j=2 OR 3) AND BEFORE FINAL SIGN CORRECTION IS ACCOMPLISHED, A CHECK IS MADE FOR POSSIBLE DIVIDE FAULTS. IF Q_{29}="1" A DIVIDE FAULT HAS OCCURRED.

* THE DIVISOR MUST BE A NEGATIVE VALUE. IF D_i IS POSITIVE-THE "C" SEQUENCE WILL: D⟶X, CLEAR D, X'⟶D, IF D IS NEGATIVE C SEQUENCE WILL: D⟶X, CLEAR D, X⟶D.
** WHEN A≥D', D' IS SUBTRACTED FROM A IN THE ADDER AND THE RESULT IS TRANSMITTED TO X, THEN TO A.

124.150

Figure 11-16.—f = 23, divide.

However, the divide instruction does provide for the detection of such a fault after the division has been performed. Two divide faults (divisor equal to zero and quotient overflow) are explained below.

Divisor Equal To Zero

The fault resulting from a divisor that is equal to zero can be detected by programming the divide instruction with j = 3,—skip the next instruction if (Q) is negative (see table 10-2). Before the final sign correction, the skip evaluation is made to determine whether the skip condition has been satisfied. When the divisor is zero, the quotient is always negative before the final sign correction. The only other condition that will cause Q to be negative at this time is the occurrence of the quotient overflow fault. Therefore, the fault condition of the divisor being equal to zero can be detected by skipping the next instruction if Q is negative.

Consider what the quotient is before final sign correction if the divisor is equal to zero. Since the divisor is zero, the condition, (A is greater than or equal to D') is satisfied for each of the partial divisions. For each partial division in which this condition is met, the partial quotient entered in Q^{00} is a "1". Therefore, when the divisor is zero each bit of Q (including sign bit) is a "1", at the end of the division phase. Thus if the divisor is zero, Q is negative when the skip evaluation is made. With j = 3 in the divide instruction, the next instruction is skipped when this divide fault is detected. The instruction following the one that is skipped could jump to a subroutine designed to remedy the fault.

Quotient Overflow

The second fault condition where a quotient of more than 30 bits is produced (commonly called overflow), occurs when the proper quotient for a division contains more than 29 bits and a sign bit. The division procedure in the computer provides for only a 30-bit quotient. When the divisor is excessively small in comparison with the dividend, a quotient with more than 30 bits can occur.

Normally, suitable steps are taken in the program to avoid this quotient overflow condition. Sometimes however, the condition occurs because the exact size of the divisor and the dividend are not known in advance. For this reason it is desirable to have a method of detection if an overflow has resulted during division.

An examination of the quotient resulting from a division when an overflow does not occur will aid in understanding the method used to detect and overflow condition. The first partial division produces the first partial quotient bit that will be, finally, the sign bit when the 30 partial divisions have been performed. Thus, the first partial quotient is the sign bit of the final quotient. Initial sign correction produces a dividend and divisor that gives a positive quotient before final sign correction, if there is no divide fault. Therefore, the first partial quotient should be a "0".

In order for the first partial quotient to be "0", the first partial dividend must be less than the divisor. An overflow fault occurs when the first partial dividend is larger than or equal to the divisor (A is greater than or equal to D'). As a consequence, the first partial quotient is a "1". Thus, before final sign correction, Q29 contains a "1" (Q is negative). Before the final sign correction, Q is positive if an overflow condition does not exist, while Q is negative if an overflow condition does exist. Since the evaluation is made to determine if the skip condition is satisfied before the final sign correction, an overflow fault can be detected by programming the divide instruction with j = 3 (skip the next instruction if Q is negative). Divide faults can also be detected by using a j and a corresponding routine.

COMPARE

The operation of comparison is performed in two different ways by two types of instructions: f = 04 Compare, and F = 43 Masked Comparison (defined later). Since the two methods of comparison vary greatly, they are discussed separately.

COMPARE INSTRUCTION (f = 04)

Instruction f = 04 compares (Y) the operand with (A) and/or (Q). The j designator specifies the comparison condition which, if satisfied, results in the next sequential instruction being skipped.

Although the contents of A and Q are altered during the compare sequence, steps are taken to restore the original contents of A and/or Q. This instruction has no net effect on the contents of any operational register.

The j designator is interpreted in the following manner for specification of the conditions for skipping the next instruction.

j = 0	Do not skip
j = 1	Skip next instruction
j = 2	Skip if (Y) $\leq$ (Q)
j = 3	Skip if (Y) > (Q)
j = 4	Skip if (Q) $\geq$ (Y) A
j = 5	Skip if (Q) < (Y) or (Y) A
j = 6	Skip if (Y) $\leq$ (A)
j = 7	Skip if (Y) > (A)

A Skip evaluation circuit (now shown) senses the j designator and an Adder output to check for the skip conditions. During the B sequence Y is read up from memory and placed in the D register as follows: (Y)$\longrightarrow$X, X'$\longrightarrow$D, and sense j is initiated. The C sequence initiates sense j (for A) and the transmission of A$\longrightarrow$X; A is cleared and Q$\longrightarrow$A. The Adder now manipulates A and D, and senses j. Then the C sequence clears A, X$\longrightarrow$A. This places the original contents of A back in the A register. The Compare instruction has been accomplished and the contents of Q and A have not been changed. A final = A initial and Q final = Q initial (A_f = A_i and Q_f = Q_i).

MASKED COMPARE INSTRUCTION (f = 43)

The Masked Comparison instruction (see table at end of Chapter 10) provides for the comparison of a portion of Y with (A) or a portion of (Q) with (A). If the conditions specified by the j designator are satisfied, the next instruction is skipped.

Comparing (Y) With (A)

The case where a portion of Y is compared with (A) is discussed first. The selection of the portion of Y that is to be compared with (A) is accomplished by masking the portion which is not to be used in the comparison. During a previous instruction, the mask is entered in the Q register. The mask (in Q) consists of bits that are "1" for the portion of Y to be compared and "0" for the portion that is not to be used in the comparison. The masking of the portion of Y that is not to be used is accomplished by the logical product method as follows: The B sequence initiates the read operation that places Y in the D register. The C sequence then causes the simultaneous transmission of (D) and (Q) to X. The X register now holds the logical product of Y and (Q).

The following is an example of a logical product as used for masking. Note: Original valve of X not shown.

EXAMPLE	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Y (D)	0	1	1	0	1	1	0	1	1
Q (MASK)	0	0	0	1	1	1	1	0	0
X'	0	0	0	0	1	1	0	0	0

In this example, the bit positions 2^2 through 2^5 are the only portions of Y that are to be compared with (A). All other bits have been masked. The C sequence then initiates the transmission of X' to D. With the unmasked portion of Y in the D-register, the comparison is now made with (A). The computer now senses j and skips if the conditions are satisfied.

Comparing (Q) With (A)

The second masked comparison is that of comparing a portion of (Q) with (A). Now the mask is Y so only the desired portion of (Q) is used in comparison. As in the first example, "1's" are used for the portion to be compared and "0's" provide the mask. The operation of masking the unused portion of (Q) is also done via the logical product method. The B sequence enters Y in the D-register and the C sequence forms the logical product in the X-register. The following shows an example of this operation. Note: Original valve of X not shown.

	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Y (D-MASK)	0	1	1	1	1	0	0	0	0
Q	0	1	0	0	1	1	1	0	1
X'	0	1	0	0	1	0	0	0	0

In this example the bit positions 2^4 through 2^7 are the only portions of (Q) that are compared with (A). All other bits have been masked. The C sequence initiates the transmission of X' to the D-register and the skip evaluation is made.

CHAPTER 12

NTDS COMPUTER MEMORY SECTION

The five principal parts of the memory section (fig. 12-1) of the CP-642A computer are (1) the memory stack which contains the storage element of the system; (2) the address selection circuits, which function to select the specified memory address; (3) the data control circuits, which control the flow of information into and out of memory; (4) the timing control circuits, which establish the timing relationships during cycles of operation, and (5) the wired memory which provides a permanently wired bootstrap load routine in an auxiliary memory. The entire memory section is contained in five identical memory chassis and a control chassis (not shown). The control chassis contains the S-register and translator, the Z-register, and the memory timing chain. The actions of these circuits to control the memory section operations are discussed later.

GENERAL DESCRIPTION

Small ferrite magnetic cores are used as the basic storage elements in the computer memory section.

Each of the five memory chassis contains a portion of the magnetic core storage system. The memory is designed so that the speed of operation is compatible with the other computer sections (high-speed); data may be referenced in a sequential or nonsequential manner (random-access); and the memory elements will retain data when power is removed from the computer (nonvolatile).

There is a total of 32,768 locations (addresses) on the five memory chassis for the storage of 30-bit words. Each chassis can store a six-bit segment of the 30-bit word. Every storage location is assigned a separate address (00000_8 through 77777_8). A 30-bit word in storage can be divided into two 15-bit words; the upper 15 bits (M_U) and the lower 15 bits (M_L). By means of programming, and the use of the k-designator (discussed in chapter 10), each 15-bit word can be handled separately.

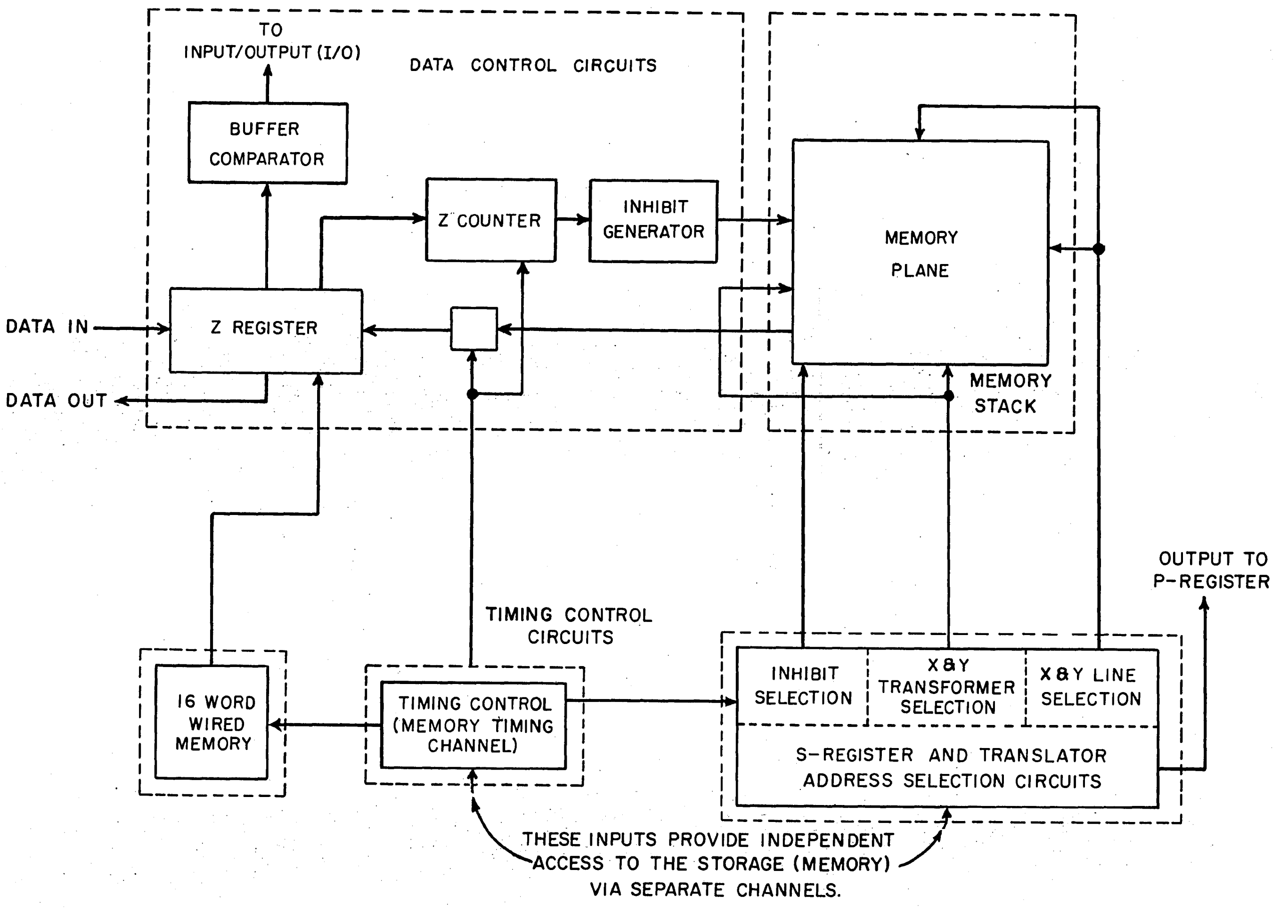
When a specific storage location in memory is referenced by a specific instruction in the program, the S-register will contain a 15-bit address word that specifies one of the 32,768 storage locations. The data transmission into or out of the selected storage location (address) is channeled through the Z-register.

The Control and Input/Output sections of the computer have independent access to the storage registers through the use of the S-register and translator or through the Memory Timing circuits and the Z-register. One locks the other out when it has control.

The time required for one memory reference (basic memory cycle time) is 8.0 microseconds. After a given function (dictated by the program or fed in from the console) initiates memory, it is approximately 2.0 microseconds before the delivery of data from storage (readout time). All timing relationships in the memory section are established by the Memory Timing chain.

The memory section of the computer is a current-operated, magnetic core memory. As previously stated, it used the permanent magnetic properties of ferrite cores to store binary information (the theory of operation of magnetic cores is treated in chapter 6). The cores are assembled in square matrices on a memory board. There are 12 of these boards or 6 planes (2 boards per plane) on each memory chassis. Figure 12-2A shows a simplified memory board containing a 16 x 16 array of cores, although the actual boards used in the computer each contain a 128 x 128 array (fig. 12-2B) arranged in four 64 x 64 matrices. This latter array provides 16,384 one bit storage addresses.

Selectable conductors (drive lines fig. 12-2A) are threaded through each core in each row and each column. Any one of the magnetic cores in



124.151

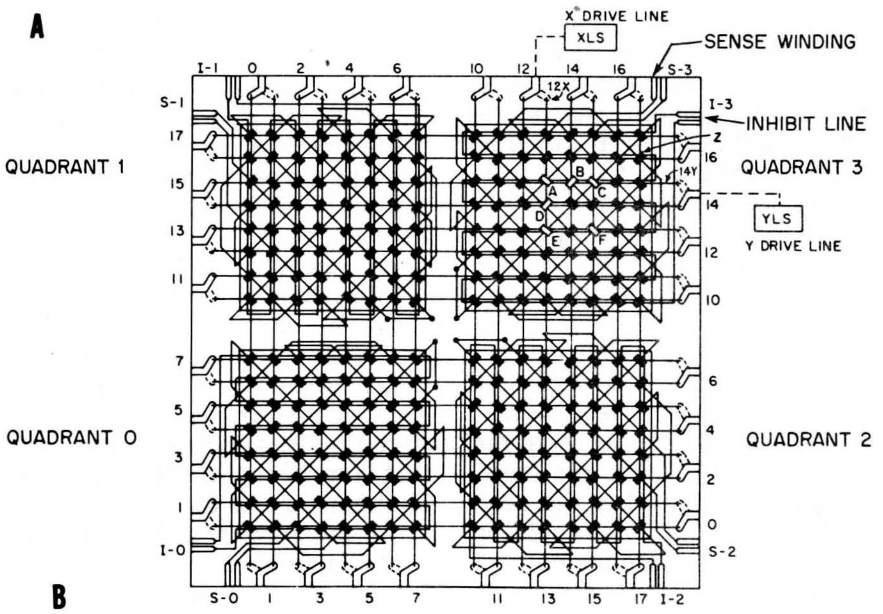
Figure 12-1.—Memory section block diagram.

the matrix may be selected (addressed) by passing a half current pulse through a given horizontal row and another current through a given vertical column, thus causing the following sequence of events to take place: A coincident half-amplitude current pulse is generated in each selected row and each selected column, the core at the intersection of the row and column (selected core) receives a net full-amplitude current pulse and is therefore selected. For example, assume core A (fig. 12-2A) is to be selected or addressed. Drive lines 12X and 14Y are pulsed with half-amplitude current pulses. Core A at the intersection of lines 12X and 14Y receives a net full-amplitude current pulse. All other cores in the 12X and 14Y lines receive half-amplitude current pulses (cores B, C, D, and E, etc.) which is an insufficient current amplitude to switch or alter data stored in these cores. The cores which receive only half-amplitude current are referred to as "half-selected". The remaining cores, neither half-selected nor fully selected, are referred to as unselected cores (core F).

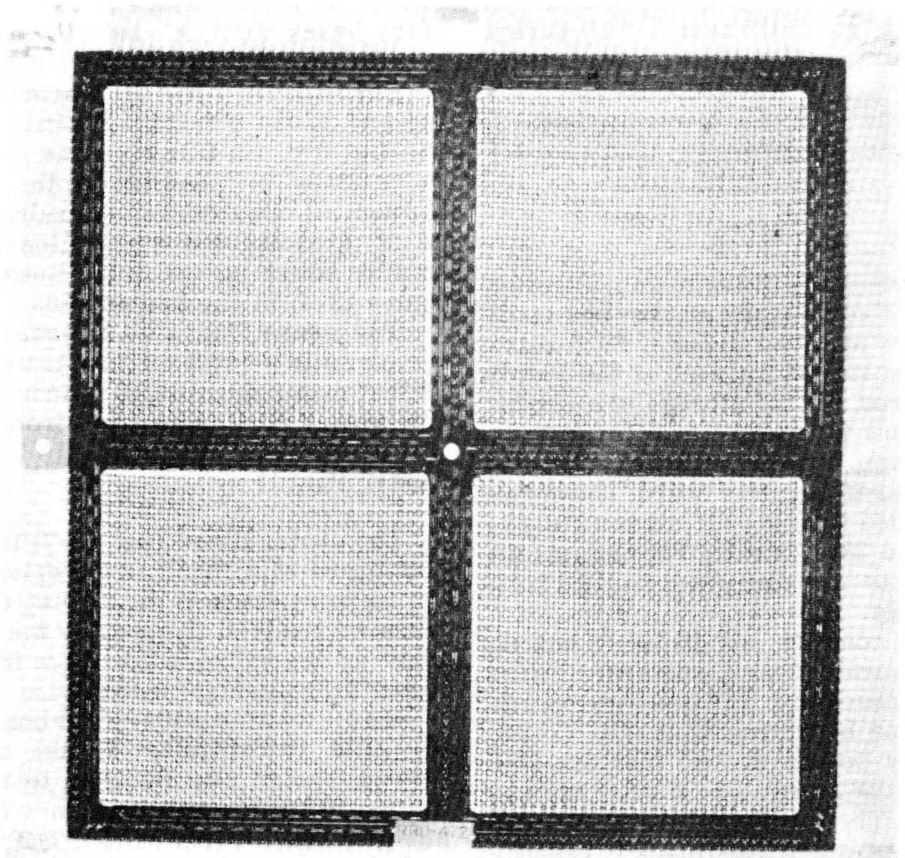
The binary information ("0" or "1") stored in a core is determined by the polarity, or the direction of magnetization.

Information is extracted from a selected core when two coincident half-amplitude read current pulses, are applied in an attempt to drive the core to the "0" state. If the core was in the "1" state, it will shift to the zero state inducing current into the sense winding. If the core was in the zero state no shift will occur. A voltage (55 millivolts nominal) is induced in the output line (sense winding) when the core is shifted. When the selected core is already in the "0" state, no shift occurs so that an insignificant voltage (10 millivolts maximum) is induced in the sense winding when the Read pulses are applied. The small induced voltage indicates no flux reversal.

A



B



124.58

Figure 12-2.—Magnetic core matrices. A. Simplified magnetic core board 16 by 16 array. B. 128 by 128 array, top view.

When operating on a selected core as explained in chapter 6), a Read pulse is followed by a Write pulse of the opposite current direction through on the drive lines. For this reason the drive lines are also called read/write lines or R/W lines. When a core contains a ''1'', a full-amplitude Read pulse switches the core to ''0'' (destructive readout). The following full-amplitude Write pulse switches the core back to a ''1''. If it is desired to leave ''0'' in the selected core, an Inhibit pulse is applied on an inhibit line simultaneously with the Write pulse. The Inhibit pulse is applied in the read direction and is equal to a half-amplitude Read pulse. Thus, if it is desired to maintain core A in the ''0'' state after reading a ''1'' from this core, an Inhibit pulse is applied on line Z in the read direction. The net effect on core A by applying a full-amplitude Write pulse to the core coincident with an Inhibit pulse is a half-amplitude Write pulse in core A. This pulse is insufficient to switch the core from ''0'' back to ''1'' during the write portion.

The principle of operation described here for the selection of one core will also apply cores of a selected address (30-bits) for each memory reference.

MEMORY STACK

The storage elements of the memory section are contained in five memory stacks; one stack on each of the five memory chassis. Each stack contains six bit positions for each of the 32,768 addresses required for the storage of a six-bit segment of a 30-bit word.

A stack contains 12 memory boards, two end board assemblies, and a center board. The memory stack is divided into two half-stacks (fig. 12-3A) which store 16,384 bits each. The half-stacks are referred to as the front stack and the back stack, respectively.

A half-stack contains six memory boards above, or below a center board, and is terminated by an end board assembly. The memory boards are interconnected by spring clips (fig. 12-3B, only two memory boards shown) that are also used to connect the center board or to the end board assembly. The front and back stacks are often referred to as stack 0 and stack 1, respectively.

Memory Board

The memory board is the basic unit of the memory stack. It has 16,384 magnetic cores, each of which is at the intersection of horizontal and vertical conductors. Two memory boards are required to store all the possible addresses (32,768) for one bit. (See figure 6-4 for a simplified view of drive line intersection.) The conductors threaded through cores along the (X) axis are called X drive lines. The conductors threaded through cores along the (Y) axis are called Y drive lines.

The X and Y drive lines terminate at tabs along the edges of the board (fig. 12-4A). A drive line connected to an upper tab on one edge of a board terminates on a lower tab at the opposite edge. A memory board has four inhibit windings (fig. 12-4B) and four sense windings (fig. 12-4C) that are brought out to solder terminals at the four corners of the board.

A memory board is divided into four quadrants numbered from the lower left (fig. 12-5). Quadrants 0 through 3 are in the front stack (stack ''0''); quadrants 4 through 7 are in the back stack (stack ''1''). A quadrant on an actual board contains a 64 x 64 array of cores (fig. 12-6).

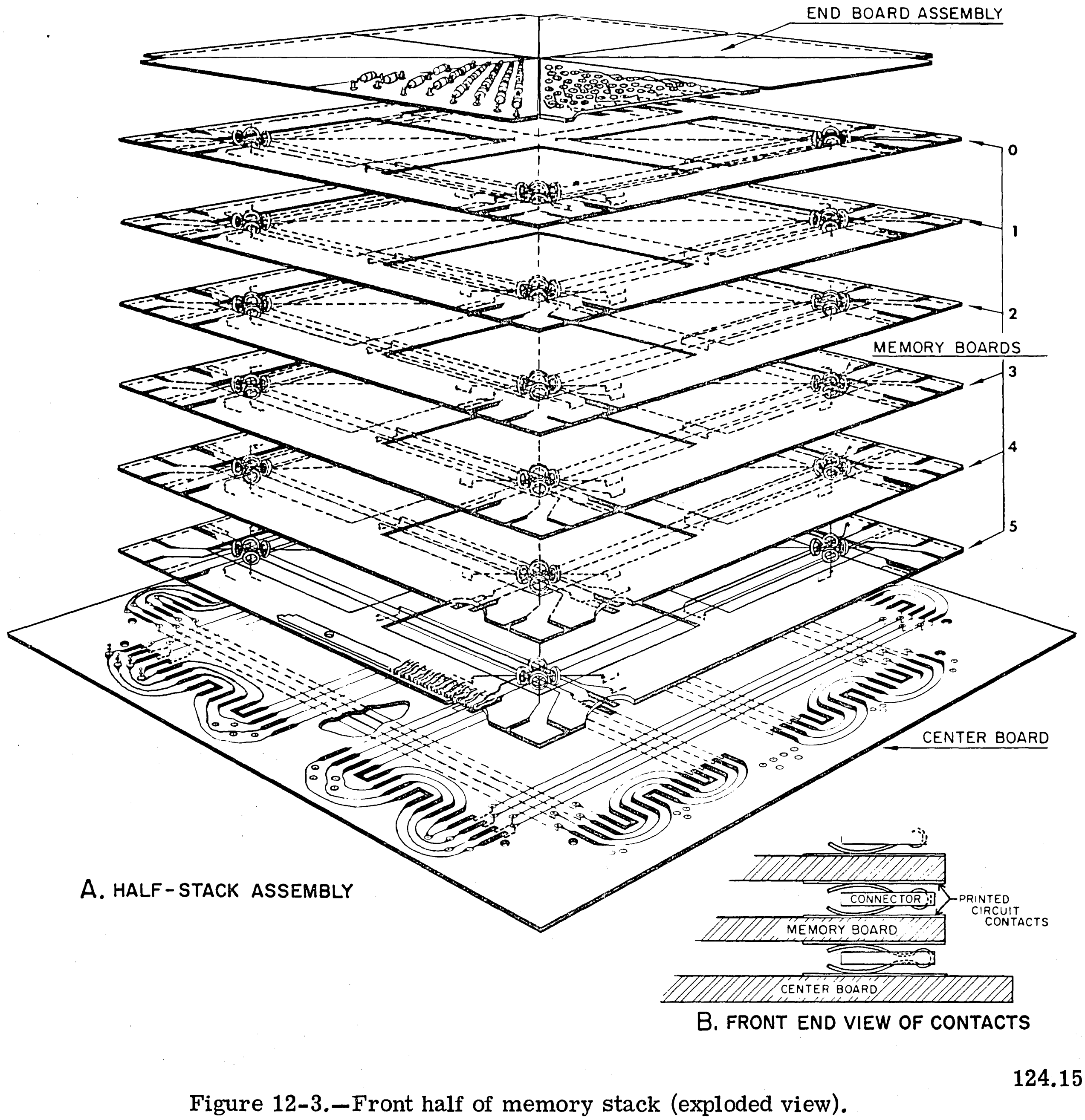
Each quadrant has its own inhibit (INH) winding and sense windings (S windings). An inhibit winding (fig. 12-4) is threaded parallel to the X drive lines or parallel to the Y drive lines through all of the cores in a quadrant. This method of threading the inhibit lines equalizes the loading effect on the drive lines (not shown in figure 12-4) by the inhibit lines.

The sense line, also threaded through all of the cores in a quadrant, is oriented within a core so that maximum voltage is induced in the sense line when a core is switched from a ''1'' to ''0''.

Center Board

The center board (fig. 12-7) is a printed circuit board that connects the driven elements of the memory stack to the driving elements on the memory chassis. It connects the X and Y drive lines on the memory boards to the X and Y selector transformer secondaries (shown later).

On the top side of the center board (fig. 12-7A) the tabs (rectangular areas) are connected through spring clip (similar to those shown in fig. 12-3B) to the drive line tabs on the adjacent memory board (front stack). The tabs on the underside of the center board (fig. 12-7B) are connected through spring clips to the drive line tabs on the adjacent memory board (back stack). In fig. 12-7A, the left side vertical wiring and tabs connect to the X drive lines in quadrants 0 and 1. The right-side vertical wiring and tabs connect

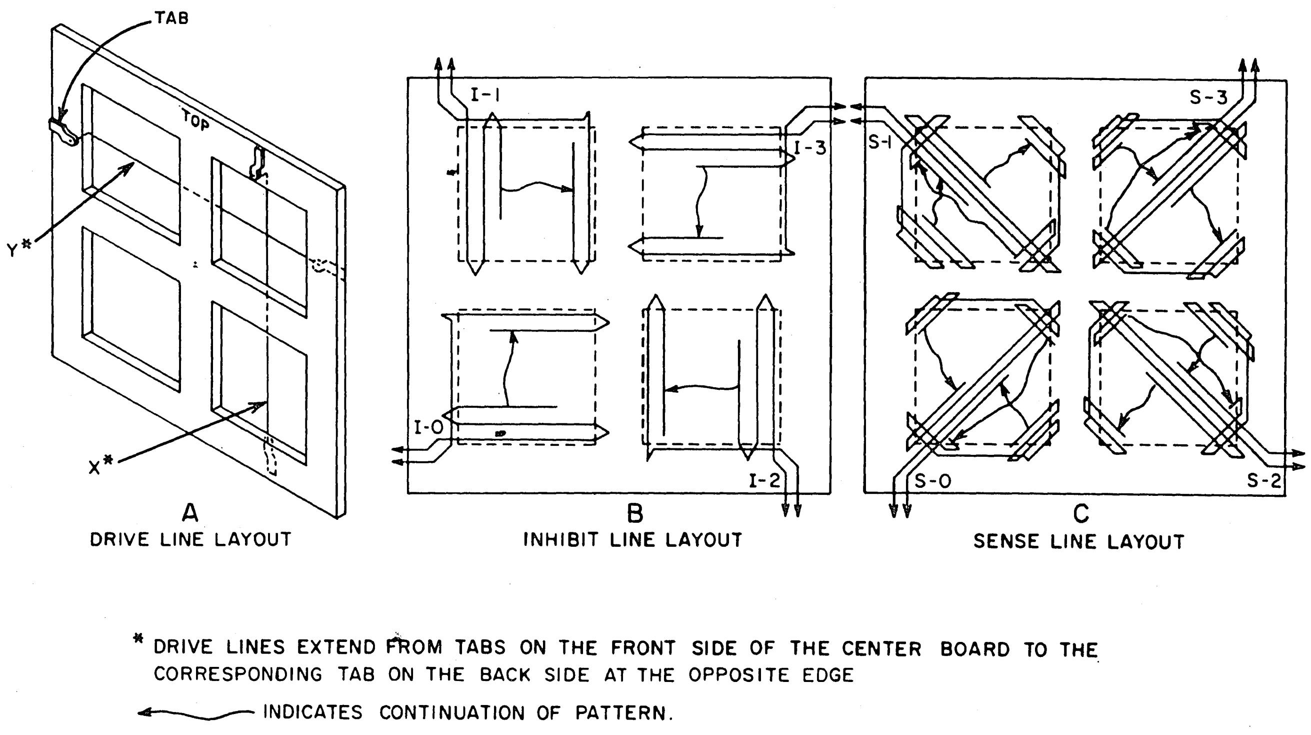


124.153

Figure 12-3.—Front half of memory stack (exploded view).

X drive lines in quadrants 2 and 3. The lower horizontal wire tabs (not shown) connect to the Y drive lines in quadrants 0 and 2. The interconnecting wiring for the horizontal tabs is located on the underside of the board. The tabs on the underside of the board are connected to the X and Y drive lines in corresponding quadrants (4, 5, 6, and 7) in the back stack.

In any group of wiring on the center board there are 32 tabs at opposite edges of the board. The printed wiring connects the tabs together in such a manner that there are eight groups of eight interconnected tabs. This provides the 64 drive line connections for two quadrants, e.g., quadrants 0 and 1 (X drive lines) or quadrant 0 and 2 (Y drive lines).


TAB
TOP
Y*
X*
A
DRIVE LINE LAYOUT
I-1
I-3
I-0
I-2
B
INHIBIT LINE LAYOUT
S-1
S-3
S-0
S-2
C
SENSE LINE LAYOUT
* DRIVE LINES EXTEND FROM TABS ON THE FRONT SIDE OF THE CENTER BOARD TO THE CORRESPONDING TAB ON THE BACK SIDE AT THE OPPOSITE EDGE
INDICATES CONTINUATION OF PATTERN.

124.154

Figure 12-4.—Memory board wiring layout.

End Board

An end board assembly, one of which terminates each half-stack (only one shown in fig. 12-3), consists of eight printed wiring boards and the connected diodes. Each end board is also referred to as a diode board. The eight diode boards in an assembly are divided into four pairs, each pair consisting of a right- and left-hand board (fig. 12-8). A pair of diode boards is needed to terminate the 64 drive lines in any two corresponding quadrants, e.g., quadrants 0 and 1 (X drive lines). The end-board assembly is connected to the adjacent memory board by spring clips that make the interconnections between tabs on each board. In this manner, printed wiring is used to connect each of 32 drive lines to two diodes on each diode board.

ADDRESS SELECTION

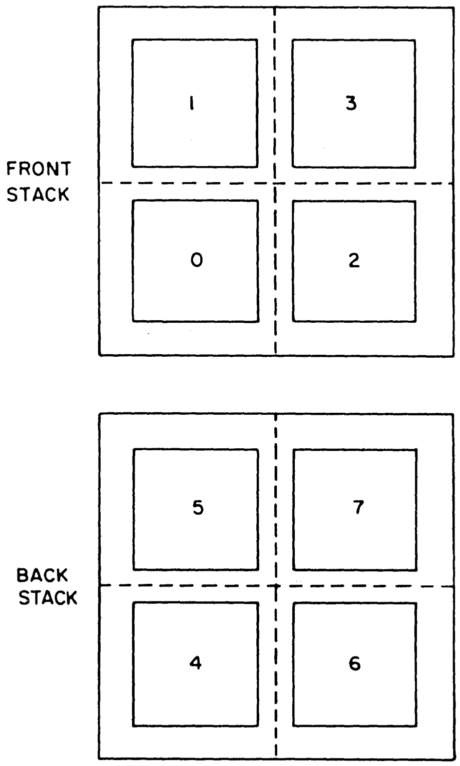
Address selection is the translation of the contents of the S-register (storage address register) to enable the desired X and Y drive lines. The 30 cores at the intersection of 30 selected drive lines represent the specified memory register. Since the Memory section of the computer can be considered as $32,768_{10}$ 30-bit memory registers, the S register is used to identify each of these registers by a unique address. The addresses range from 00000_8 through 77777_8.

At the beginning of a memory reference, the address of a specific memory register is transmitted to S. (This process is explained in chapter 4.) The contents of S are translated as described in the following paragraph to select the unique group of 30 cores that represents the specified memory register.

The three basic problems of address selection are: (1) selection of the front or back stack on each memory chassis, (2) selection of an X and Y drive line in each stack, and (3) selection of the inhibit line that is in the same quadrant as the selected drive lines. All of the three selections are made by outputs from the S translator.

S-REGISTER AND TRANSLATOR

Any computer function requiring access to memory must first transmit the address of the



124.155
Figure 12-5.—Quadrant designations.

memory register into S. During the memory reference, outputs from the S translator enable the proper X and Y drive lines and the proper inhibit lines.

Initially, the S-register (fig. 12-1) is cleared by an enable from the memory timing chain (shown later) or by the master clear circuits (not shown). After clearing, a clock phase enables the input circuits to S thereby permitting the S-register to be loaded (receive the input data).

The inputs to S (see figure 8-1) are from any of, the following sources: (1) the P-adder which is command timed (receives timed command enable signals) from the sequence (A-, -B, C-, or D), (2) from Z_L (the lower 15-bits of the Z-register), or (3) from an address in memory (via Z_L, U_L and the U-adder) which specifies the address to be used by an input/output (I/O) channel.

During the A-sequence, the address of the next instruction to be executed is generally transmitted to S from the P-adder (P-adder $\longrightarrow$ S). During an interrupt A sequence (discussed in chapter 10), initiated by an interrupt instruction, the interrupt address is fed via U_L and the U-adder is S (interrupt address $\longrightarrow$ S). Normally, the address of the operand is transmitted to S from the U-register (U-adder $\longrightarrow$ S) during the B sequence. For various store and replace operations the D sequence initiated the following transmissions for U_L and the U-adder: U-adder $\longrightarrow$ S (for storing and replacing data from the U-register), and $\hat{j} \longrightarrow$ S (where $\hat{j}$ represents a selected input/output channel from which the storing and replacing of data is involved).

During buffer operations a $Z_L \longrightarrow$ S transmission takes place, thereby permitting the address received from the buffer via the Z-register to control the S register memory selection circuits.

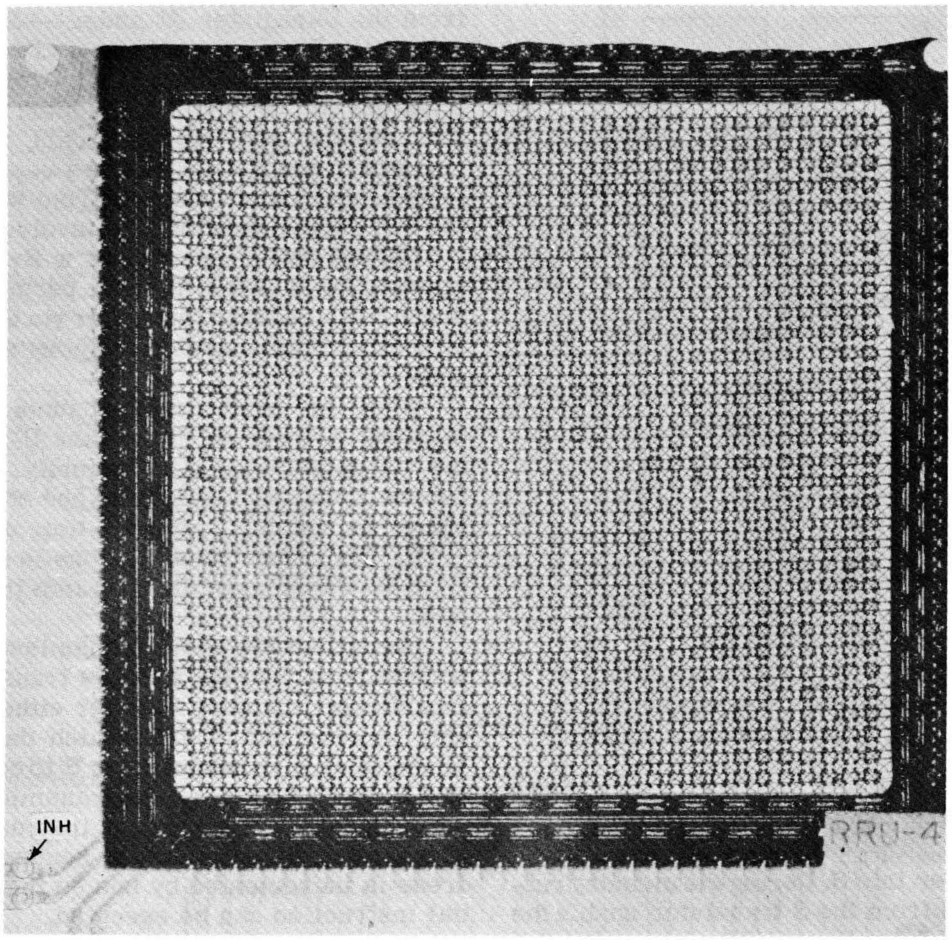
When an input/output channel is making a memory reference, U_L and the U-adder causes the transmissions I/O priority address $\longrightarrow$ S (where "priority" indicates one of 14 I/O channel to be selected) or real-time clock address $\longrightarrow$ S. The latter transmissions is not gated into S but is transferred directly into the proper bits of S.

For all of the above transmissions to S, the outputs from S to the address translator specify a particular memory register either from which data is read out or into which data is stored. There is also an output from S to corresponding bits in the P register. The transmission returns the address of the present instruction to P in preparation for the next A sequence. This address is incremented by one so the next sequential instruction can be executed.

SELECTOR TRANSFORMER

Note in figure 12-7 that there are 32 tabs in each group of wiring on the centerboard. The printed wiring connects the tabs together so that there are eight groups of eight interconnected tabs. This provides the 64 connections as necessary for completing circuit paths to the drive lines for two quadrants, e.g., the X drive lines for quadrants 0 and 1, or the Y drive lines for quadrants 0 and 2.

Selector transformers (fig. 12-9) are used to enable the X and Y drive lines. Each of two selector transformer primaries feed four dual wound secondaries. There are eight X selector primaries (only two shown) and eight Y selector primaries on each of the five memory chassis. Each primary has four secondaries.



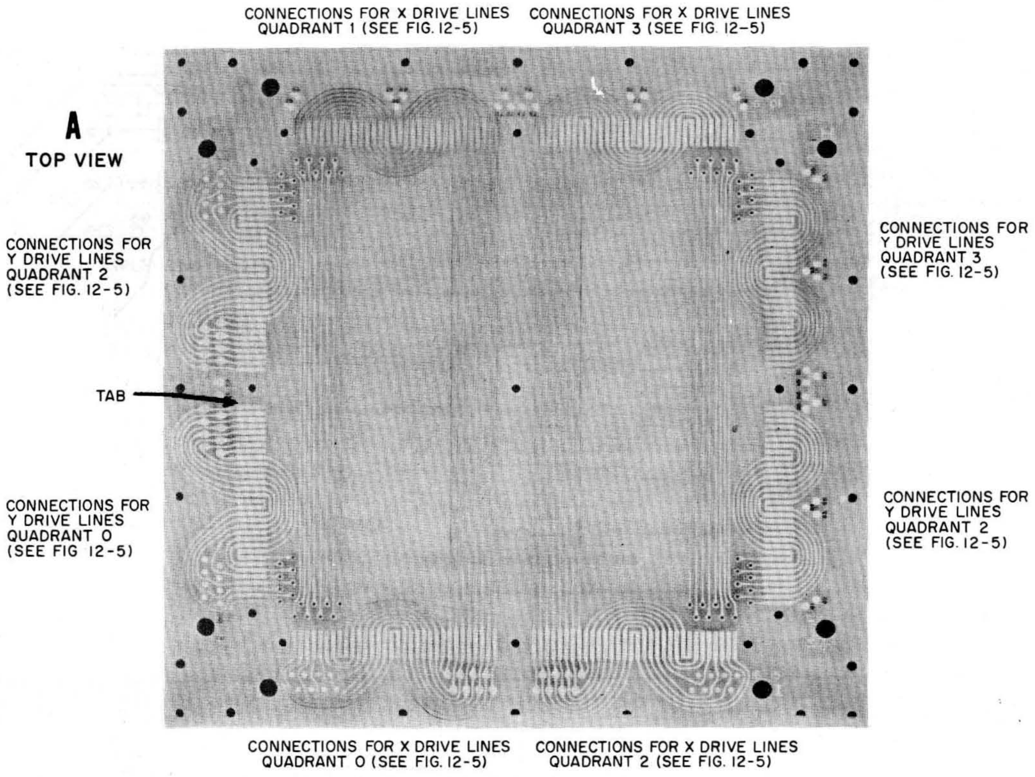
124.58

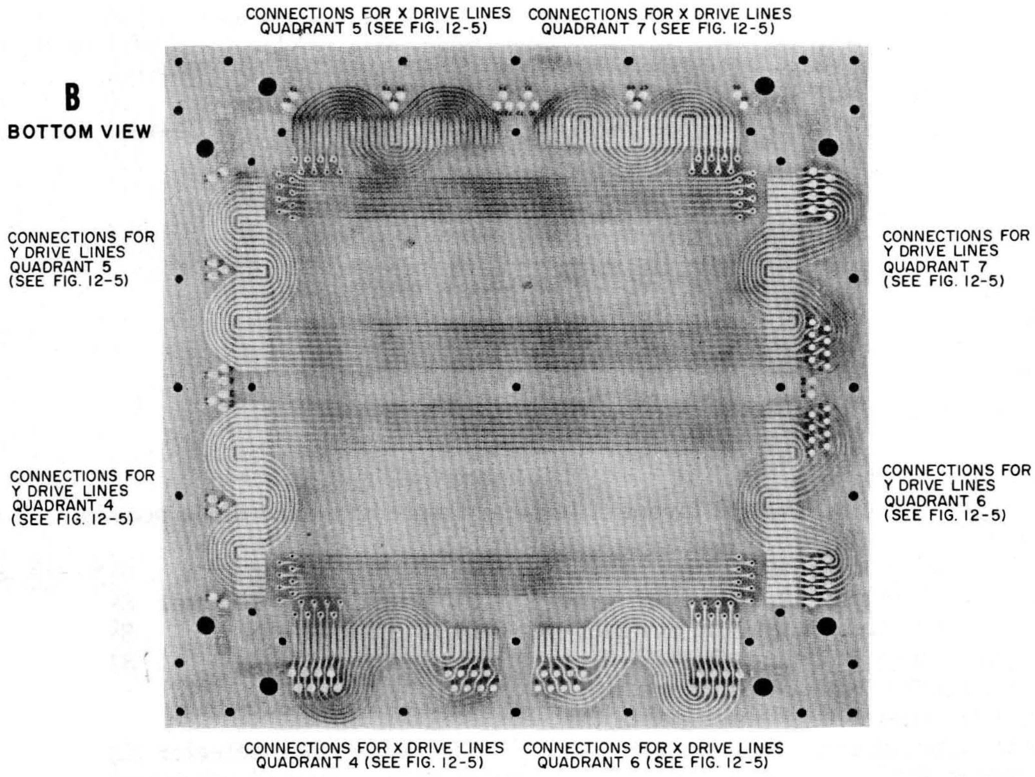
Figure 12-6.—Quadrant 64 by 64 array, top view.

The primaries can be connected to provide read or write power to the top or bottom four secondaries. For simplicity the method of completing desired circuit paths is shown here using single-pole single-throw switches, although in the actual computer circuits the selecting function is accomplished by transistor circuitry (shown later). The desired primary winding is selected by completing the circuit path through the appropriate primary selector and timing control inputs. Any one of the eight secondary windings is enabled by completing the desired input path through the secondary selector and the line selector. The eight secondary windings from the two selector transformers associated with the X drive lines for, say quadrants 0 and 1 (fig. 12-5) are connected to these terminals. Since each terminal is connected to eight drive lines (fig. 12-9), a secondary of each selector transformer connects to these same eight lines. The remaining connections for the X and Y drive lines are made in a similar manner to the other groups of numbered terminals.

ADDRESS TRANSLATION

The S translator (fig. 8-1) translates the outputs from the S register to specify the unique enables that select the X and Y drive lines and the inhibit lines. Ths S translator is divided into


CONNECTIONS FOR X DRIVE LINES QUADRANT 1 (SEE FIG. 12-5)
CONNECTIONS FOR X DRIVE LINES QUADRANT 3 (SEE FIG. 12-5)
A
TOP VIEW
CONNECTIONS FOR Y DRIVE LINES QUADRANT 2 (SEE FIG. 12-5)
CONNECTIONS FOR Y DRIVE LINES QUADRANT 3 (SEE FIG. 12-5)
TAB
CONNECTIONS FOR Y DRIVE LINES QUADRANT 0 (SEE FIG 12-5)
CONNECTIONS FOR Y DRIVE LINES QUADRANT 2 (SEE FIG. 12-5)
CONNECTIONS FOR X DRIVE LINES QUADRANT 0 (SEE FIG. 12-5)
CONNECTIONS FOR X DRIVE LINES QUADRANT 2 (SEE FIG. 12-5)


CONNECTIONS FOR X DRIVE LINES QUADRANT 5 (SEE FIG. 12-5)
CONNECTIONS FOR X DRIVE LINES QUADRANT 7 (SEE FIG. 12-5)
B
BOTTOM VIEW
CONNECTIONS FOR Y DRIVE LINES QUADRANT 5 (SEE FIG. 12-5)
CONNECTIONS FOR Y DRIVE LINES QUADRANT 7 (SEE FIG. 12-5)
CONNECTIONS FOR Y DRIVE LINES QUADRANT 4 (SEE FIG. 12-5)
CONNECTIONS FOR Y DRIVE LINES QUADRANT 6 (SEE FIG. 12-5)
CONNECTIONS FOR X DRIVE LINES QUADRANT 4 (SEE FIG. 12-5)
CONNECTIONS FOR X DRIVE LINES QUADRANT 6 (SEE FIG. 12-5)

124.153

Figure 12-7.—Centerboard.

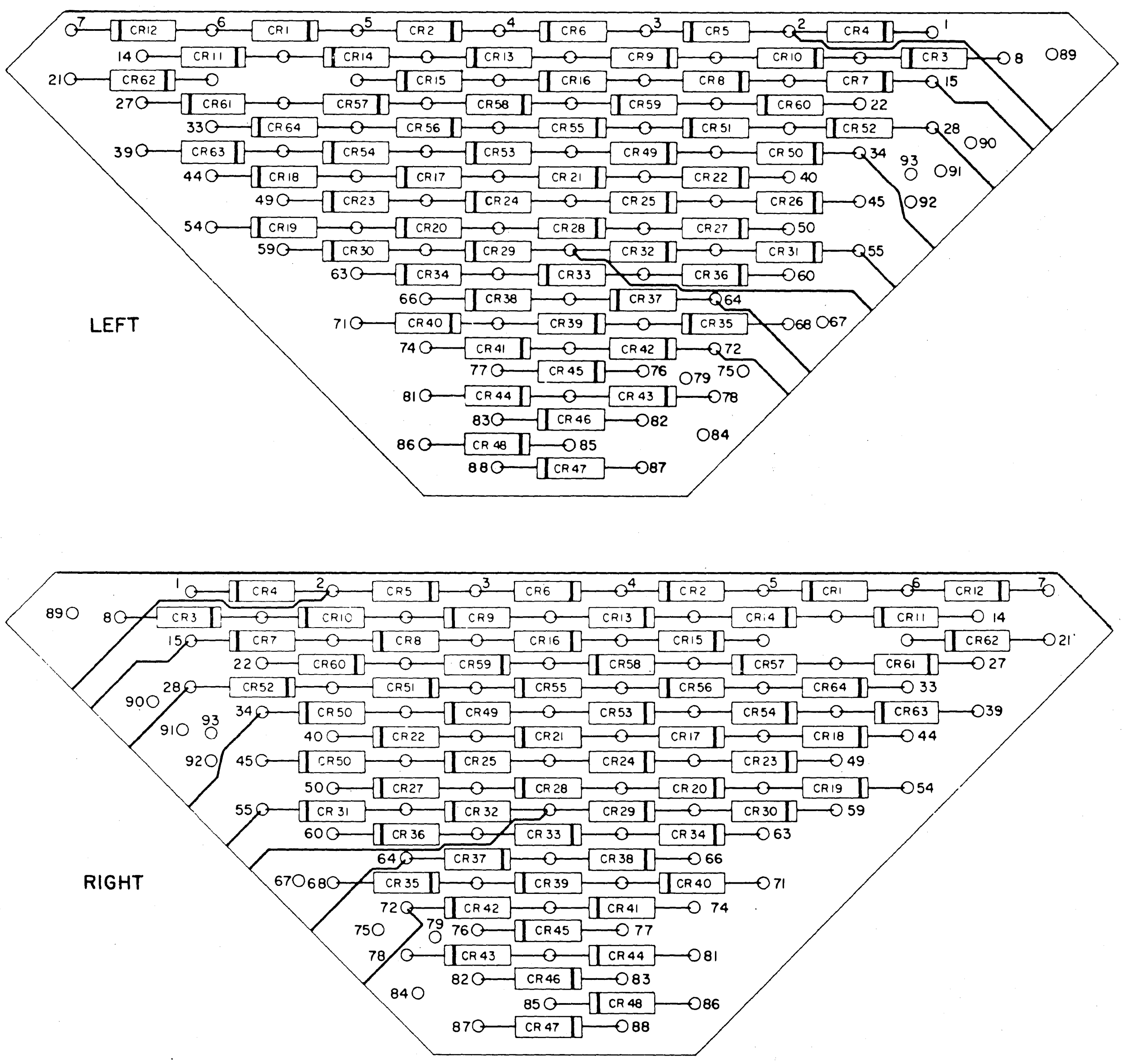


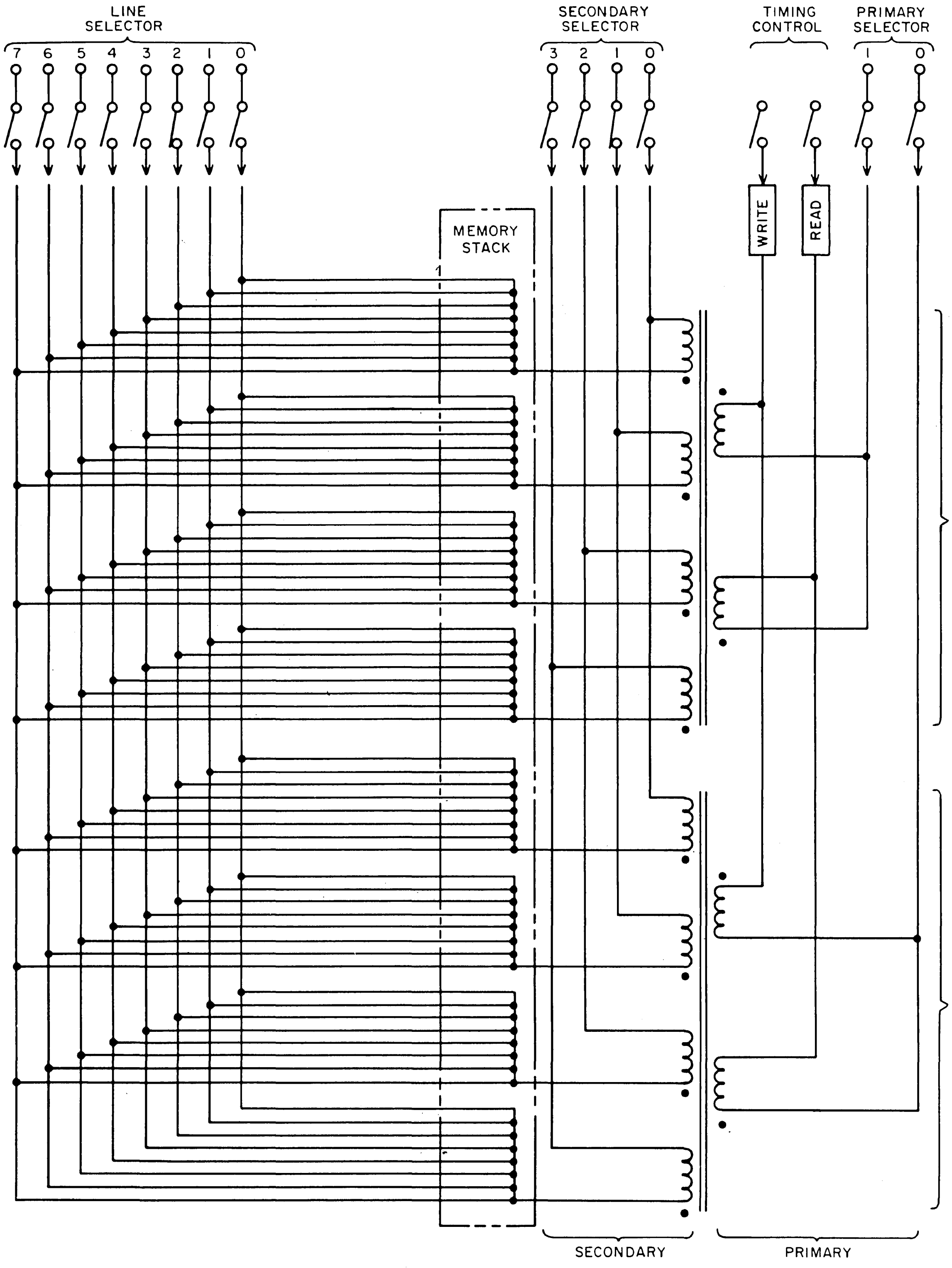
Figure 12-8.—Physical layout of left-hand and right-hand diode boards.

seven separate translator circuit groups (similar to the one shown in figure 12-10A), each with a distinct function and each controlled by specific stages of S (as shown in figure 12-10B). The following is a listing of the translations performed by the S translator, and the stages of the S-register which control the translations.

1) X primary selector	S^{14}, S^{13}, S^{11}
2) X secondary selector	S^{10}, S^{09}
3) X line selector	S^{08}, S^{07}, S^{06}
4) Y primary selector	S^{14}, S^{12}, S^{05}
5) Y secondary selector	S^{04}, S^{03}
6) Y line selector	S^{02}, S^{01}, S^{00}
7) Inhibit selector	S^{14}, S^{13}, S^{12}

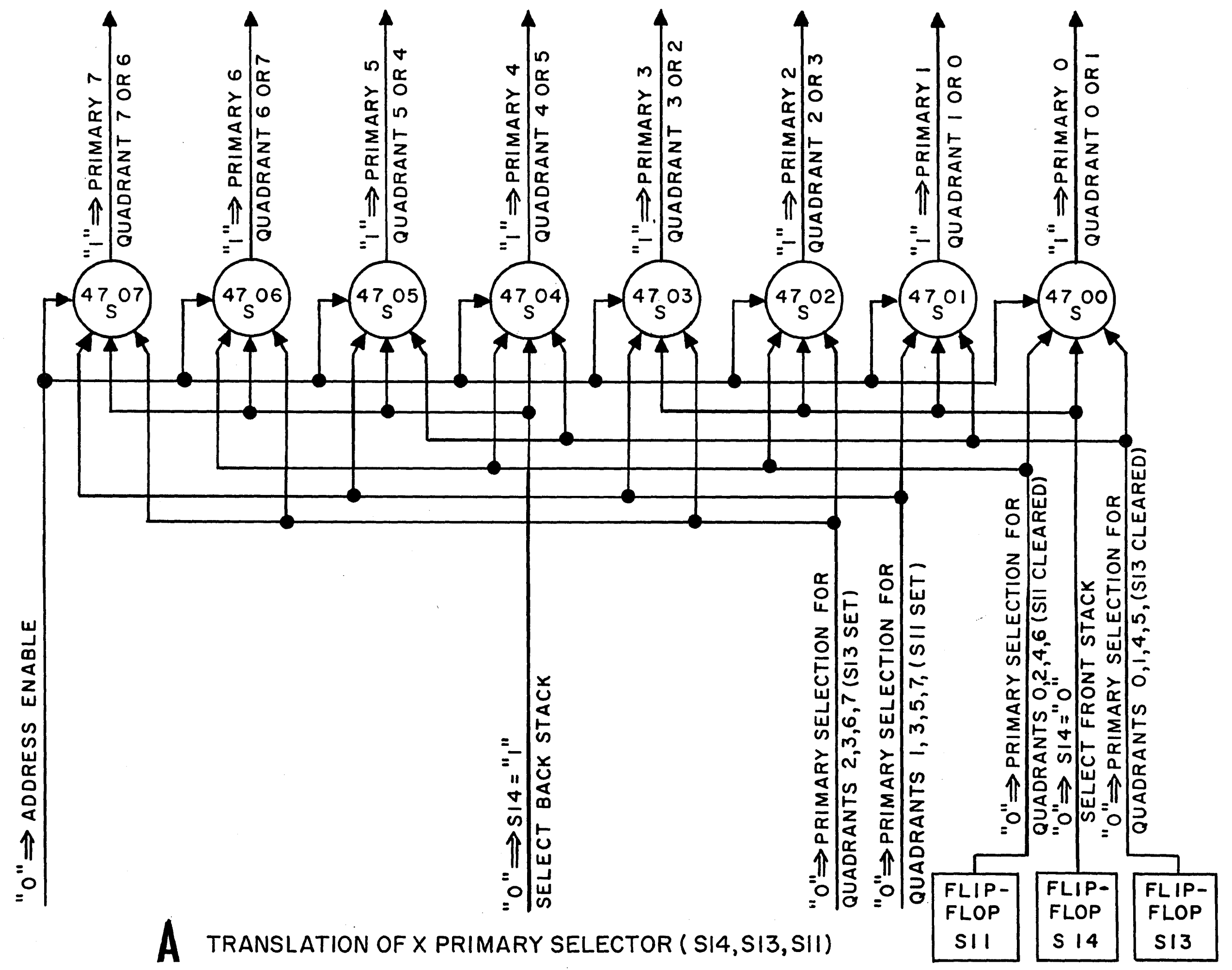
X PRIMARY SELECTOR

The X primary selector (fig. 12-10A) is the group of circuits that selects and enables the primary winding of one of the eight X selector (drive current) transformers on a memory chassis.

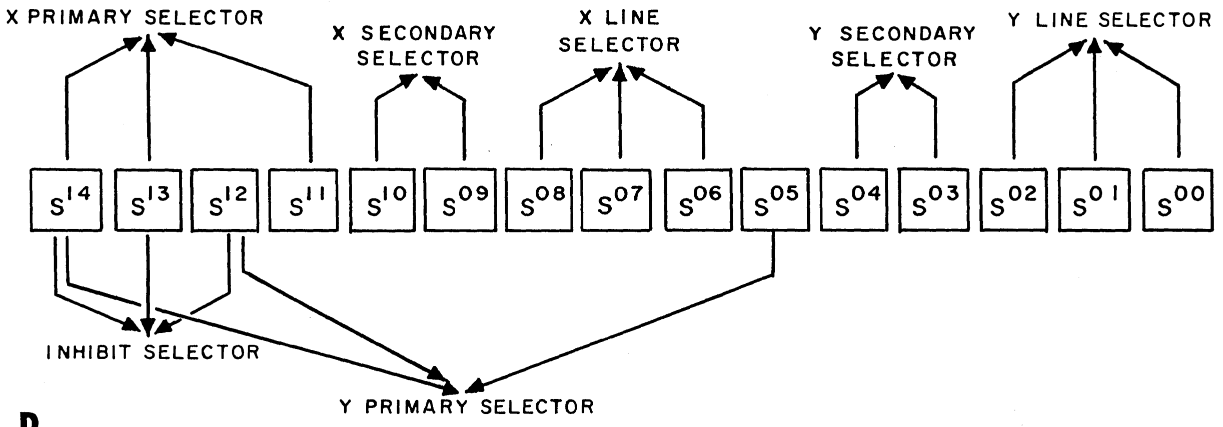


124.159

Figure 12-9.—Connections of transformers used for drive line translations.


"1" ⇒ PRIMARY 7 QUADRANT 7 OR 6
"1" ⇒ PRIMARY 6 QUADRANT 6 OR 7
"1" ⇒ PRIMARY 5 QUADRANT 5 OR 4
"1" ⇒ PRIMARY 4 QUADRANT 4 OR 5
"1" ⇒ PRIMARY 3 QUADRANT 3 OR 2
"1" ⇒ PRIMARY 2 QUADRANT 2 OR 3
"1" ⇒ PRIMARY 1 QUADRANT 1 OR 0
"1" ⇒ PRIMARY 0 QUADRANT 0 OR 1
47 07 S
47 06 S
47 05 S
47 04 S
47 03 S
47 02 S
47 01 S
47 00 S
"0" ⇒ ADDRESS ENABLE
"0" ⇒ S14 = "1" SELECT BACK STACK
"0" ⇒ PRIMARY SELECTION FOR QUADRANTS 2,3,6,7 (S13 SET)
"0" ⇒ PRIMARY SELECTION FOR QUADRANTS 1,3,5,7,(S11 SET)
"0" ⇒ PRIMARY SELECTION FOR QUADRANTS 0,2,4,6 (S11 CLEARED)
"0" ⇒ S14 = "0" SELECT FRONT STACK
"0" ⇒ PRIMARY SELECTION FOR QUADRANTS 0,1,4,5,(S13 CLEARED)
FLIP-FLOP S11
FLIP-FLOP S14
FLIP-FLOP S13

A TRANSLATION OF X PRIMARY SELECTOR (S14,S13,S11)



B CONTROLLING STAGES OF THE S REGISTER FOR EACH OF THE SEVEN S TRANSLATOR CIRCUITS

124.160

Figure 12-10.—Translation circuits for X-transformer primary.

For simplicity, assume that the eight selector transformer primaries (not shown) associated with the X drive lines are numbered 0 through 7 (fig. 12-10A). The translation circuits $^{47}S^{00}$ through $^{47}S^{07}$ produce outputs which specify the selection of one of the eight X transformer primaries. The S-register flip-flops, S^{00} - S^{14} (fig. 12-10B) store the number of the address to be referenced in memory (as discussed earlier). The outputs of the S^{14}, S^{13}, and S^{11} flip-flops only are used in determining which of the eight X selector transformer primaries will be selected. For this reason, the S^{14}, S^{13}, and S^{11} flip-flops are called the primary selector stages.

For purposes of explanation, assume that the address 13700_8 has been loaded into the S-register and thus initiates a request to reference this address in memory. The extent of the operations to reference this address is limited at this time to the selection of the X selector transformer primary only. The binary-coded octal representation of address 13700 and the contents of the S-register flip-flops are illustrated in table 12-1.

The contents of the primary selector stages are S^{14} = "0", S^{13} = "0", and S^{11} = "0". The outputs of these stages are applied (in various combinations) to the translation circuits $^{47}S^{00}$ through $^{47}S^{07}$ (fig. 12-10A). A command enable (called the address enable) is applied to all of the translation circuits. The application of the "0" address enable signal signifies that an X selector transformer is to be selected. The coincidence of the "0" address enable input and "0" inputs from S^{14}, S^{13}, and S^{11} produce a "1" output from $^{47}S^{00}$, thus indicating that the primary of transformer 0 (used to enable the X selector drive lines in quadrants 0 to 1) is to be selected. In like manner, any combinations of inputs from S^{14}, S^{13}, and S^{11}, will be interpreted by one of the translator circuits ($^{47}S^{00}$ through $^{47}S^{07}$) to indicate the selection of a specific X transformer primary.

The "1" output from $^{47}S^{00}$ (fig. 12-10A) is applied to a memory control driver $^{50}Y^{32}$ (fig. 12-11). Corresponding memory control driver circuits receive the output from each of the translation circuits (fig. 12-10A). However, for simplicity only those circuits associated with the selection of the "0" X-transformer primary are discussed.

The output of $^{50}Y^{32}$ (fig. 12-11 inverted circuits) is a "0" which is applied to the cathode of diode CR1 on transformer enables circuit $^{53}Y^{10}$. The diode cuts off and the positive potential thus applied to the base of Q5 causes this transistor to conduct. The collector potential of Q5 decreases in positive value toward ground. This negative-going voltage is applied to the base of Q1 and Q3, partially enabling these circuits.

The emitter of Q3 is connected to the emitter of the write current diverter Q6, (a part of the write circuit) while the inverter of $^{53}Y^{10}$ is tied to the emitter of a similar inverter (not shown in the read current diverter, $^{52}Y^{00}$. For every memory reference, either write circuit ($^{52}Y^{10}$) or the read circuit ($^{52}Y^{00}$) will be enabled. (The read circuit is not completely shown in the diagram. The basic circuit operation is the same as that for the write circuit which is discussed in the following paragraphs.)

When the read or write enable is generated on either the $^{51}Y^{01}$ or $^{51}Y^{00}$ input lines, the associated current diverter is turned off. Assuming a write input, Q6 switches from a conducting condition to cut off. Note that a 6 mh inductance and 470 ohm resistance comprising L3 is connected from the emitters of Q6 of the write circuit and Q3 of the transformer enabler to ground. When the write current diverter, Q6, is cut off L3 causes a large self-induced voltage (positive to ground) to be developed across the 6 mh coil.

The self-induced voltage thus generated is applied to the emitter of Q3 ($^{53}Y^{10}$). The coincident application of this input along with the

Table 12-1. — S-Register Content (Address 13700).

Octal Number	1			3			7			0			0		
Binary-coded Octal Representation	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0
S-register flip-flop outputs bit positions of S	S^{14}	S^{13}	S^{12}	S^{11}	S^{10}	S^{09}	S^{08}	S^{07}	S^{06}	S^{05}	S^{04}	S^{03}	S^{02}	S^{01}	S^{00}

negative-going base input to the Q3 base from Q5 (as discussed) causes the Q3 collector current to flow through the 15-14 primary winding of driver current transformer #0. Transistor, Q1, ($^{53}Y^{10}$) operates in a similar manner when a read enable input is received at $^{52}Y^{00}$. Thus when a read or write pulse is generated at the output of Q1 or Q3 ($^{53}Y^{10}$), the primary of the selector transformer can drive (provide a line current) as long as Q1 or Q3 is conducting depending of course on which of the two circuits is enabled. The pulse is coupled into the secondary windings and applied into the secondary windings and applied to the groups of interconnected drive lines. The selection of any X-selector transformer primary is accomplished in like manner as determined by the "1" output from the translator (fig. 12-10A).

X-SECONDARY SELECTOR

The X-secondary selector is the group of circuits that selects and enables 2 of 8 the secondary windings fed from a given X-selector primary (see figure 12-9). The translation circuits, $^{44}S^{00}$ through $^{44}S^{03}$ (fig. 12-12), designate by their output a specific transformer secondary winding in the group of four such secondary windings associated with each primary.

The stages of the S-register (see table 12-1 and figure 12-10B) concerned with the translation for the selection of the required X-secondary are S^{10} and S^{09}. The contents of these stages are S^{10}= "1" and S^{09}= "1" (address being selected is 13700). These "1" inputs are inverted (inverters not shown) and applied to the translation circuits. $^{44}S^{03}$ through $^{44}S^{00}$ (fig. 12-12). The circuit operation is similar to that described for X-primary selection as discussed with the aid of figure 12-10.

In the translation, the output from $^{44}S^{03}$ (fig. 12-12) is "1" to indicate the selection of secondary 3. This "1" output is inverted to a "0" output by $^{50}Y^{23}$ (an inverter in figure 12-11 whose output is negative to ground, called a negative enabler as discussed later), and $^{56}Y^{33}$ (an inverter whose output is positive to ground, called a positive enabler). The negative and positive enabler circuits used to select the X-secondary ($^{57}Y^{33}$ and $^{56}Y^{33}$ shown in block form) are identical to $^{57}Y^{20}$ and $^{56}Y^{20}$ shown in schematic form and used to select the desired diode pair from the memory stack, A3. Thus, the following discussion of the circuit operation of $^{52}Y^{20}$ and $^{56}Y^{20}$ is applicable to all enabler circuits treated in this chapter.

When the enablers conduct, the collectors of both circuits (i.e., Q2 of $^{57}Y^{20}$ and Q1 of $^{56}Y^{20}$) are essentially at ground potential. This near ground potential is established as a result of low voltage drop across the forward conducting resistance of the transistors in series with a 1-ohm emitter-to-ground resistance. (When the transistors are not conducting, a relatively large voltage positive to ground across R6 in $^{56}Y^{20}$ and negative to ground across R4 in $^{57}Y^{20}$, is developed from collector-to-ground. The low transistor resistance in series with the low emitter resistance to ground in each of the circuits are essentially a short-circuit across the collector resistors R6 and R4, respectively, when the transistors are conducting.)

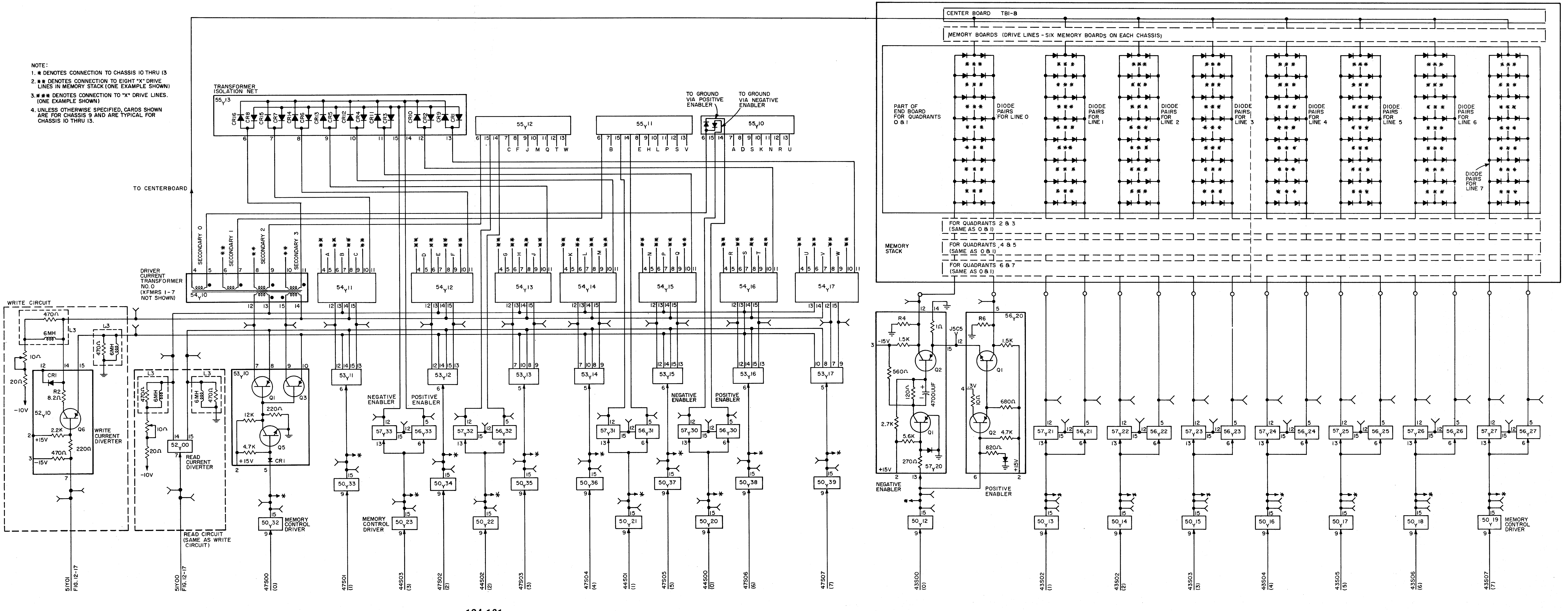
If these low-resistance-to-ground paths are established at the output of $^{57}Y^{33}$ and $^{56}Y^{33}$ (as they will be when X secondary 3 is to be selected), the ground from the output transistor of $^{57}Y^{33}$ will be applied to the anodes of CR1 through CR8 on $^{55}Y^{13}$, and the ground from the collector of the output transistor of $^{56}Y^{33}$ will be applied to the cathodes of CR9 through CR16. Because secondary 3 on $^{54}Y^{10}$ is to be selected (for address 13700) and also because only the diodes on $^{55}Y^{13}$ receive the ground enable, only diodes CR8 and CR16 conduct. The alternating voltage induced in the 10-11 winding of $^{54}Y^{10}$ (the secondary 3 winding) causes a current from ground (at the emitter of the output transistors of $^{57}Y^{33}$ and $^{56}Y^{33}$) through the forward biased diode (either CR8 or CR16) through the secondary 3 transformer winding through the centerboard (via a connection similar to that shown for the 0 secondary) to the appropriate memory boards. A subsequent signal (discussed later) selects the appropriate X line for the address 13700.

The enabling of the two diodes on $^{55}Y^{13}$ is required to allow current flow in either direction for read/write cycles. The diodes which are not conducting at any given instant are back biased by the voltage developed across the driver transformer to isolate the selected and nondriven secondary windings from the selected and driven secondary windings. The selection of any one of the four secondary windings is accomplished in like manner by the "1" output from the translator.

Because of the interconnections provided by the centerboard (discussed earlier) a group of eight interconnected drive lines is selected by the transformer secondary, and current flow will occur through certain memory cores.

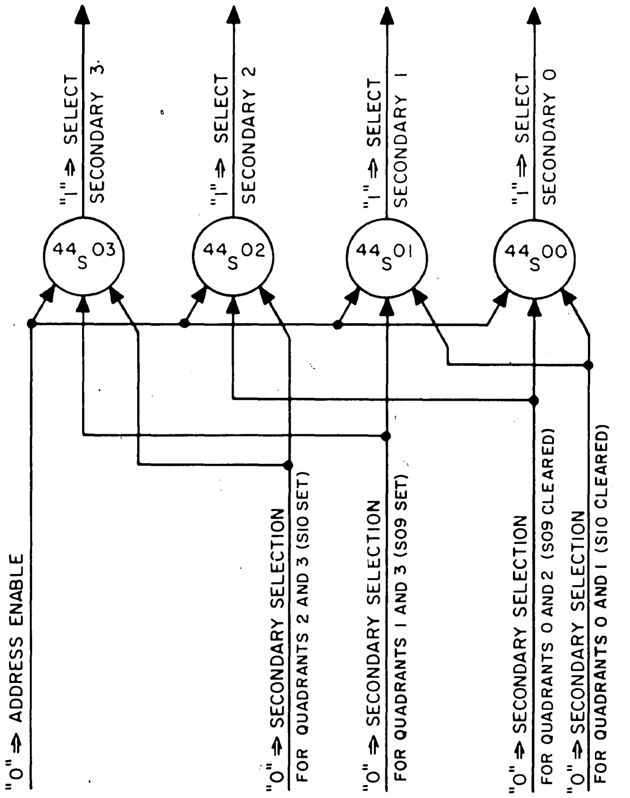
X LINE SELECTOR

The X line selector is the group of circuits that selects and enables one of the eight drive


NOTE:
1. * DENOTES CONNECTION TO CHASSIS 10 THRU 13
2. ** DENOTES CONNECTION TO EIGHT "X" DRIVE LINES IN MEMORY STACK (ONE EXAMPLE SHOWN)
3. *** DENOTES CONNECTION TO "X" DRIVE LINES. (ONE EXAMPLE SHOWN)
4. UNLESS OTHERWISE SPECIFIED, CARDS SHOWN ARE FOR CHASSIS 9 AND ARE TYPICAL FOR CHASSIS 10 THRU 13.
TRANSFORMER ISOLATION NET
TO GROUND VIA POSITIVE ENABLER
TO GROUND VIA NEGATIVE ENABLER
CENTER BOARD TB1-8
MEMORY BOARDS (DRIVE LINES - SIX MEMORY BOARDS ON EACH CHASSIS)
PART OF END BOARD FOR QUADRANTS 0 & 1
DIODE PAIRS FOR LINE 0
DIODE PAIRS FOR LINE 7
FOR QUADRANTS 2 & 3 (SAME AS 0 & 1)
FOR QUADRANTS 4 & 5 (SAME AS 0 & 1)
FOR QUADRANTS 6 & 7 (SAME AS 0 & 1)
MEMORY STACK
TO CENTERBOARD
DRIVER CURRENT TRANSFORMER NO. 0 (XFMRS 1-7 NOT SHOWN)
WRITE CIRCUIT
WRITE CURRENT DIVERTER
READ CURRENT DIVERTER
READ CIRCUIT (SAME AS WRITE CIRCUIT)
NEGATIVE ENABLER
POSITIVE ENABLER
MEMORY CONTROL DRIVER
51Y01 FIG. 12-17
51Y00 FIG. 12-17
47S00 (0)
47S01 (1)
44S03 (3)
47S02 (2)
44S02 (2)
47S03 (3)
47S04 (4)
44S01 (1)
47S05 (5)
44S00 (0)
47S06 (6)
47S07 (7)
43S00 (0)
43S02 (1)
43S02 (2)
43S03 (3)
43S04 (4)
43S05 (5)
43S06 (6)
43S07 (7)

124.161

Figure 12-11.—X-drive line translation.



"1" → SELECT SECONDARY 3
"1" → SELECT SECONDARY 2
"1" → SELECT SECONDARY 1
"1" → SELECT SECONDARY 0
44S03
44S02
44S01
44S00
"0" → ADDRESS ENABLE
"0" → SECONDARY SELECTION FOR QUADRANTS 2 AND 3 (S10 SET)
"0" → SECONDARY SELECTION FOR QUADRANTS 1 AND 3 (S09 SET)
"0" → SECONDARY SELECTION FOR QUADRANTS 0 AND 2 (S09 CLEARED)
"0" → SECONDARY SELECTION FOR QUADRANTS 0 AND 1 (S10 CLEARED)

124.162
Figure 12-12.—X-secondary selector.

lines from the group selected by the secondary selector. The translation circuits, $^{43}S00$ through $^{43}S07$ (fig. 12-13), designate by their outputs a particular X drive line in a group of eight (see connections of secondary to drive lines in figure 12-9).

The stages of S concerned with the X drive lines translation are S^{08}, S^{07}, and S^{06} (see table 12-1 and figure 12-10B). The contents of the X line selector stages ($S08$, S^{07} and S^{06}) are S^{08} = "1", S^{07} = "1", and S^{06} = "1" (for selection of address 13700). These outputs from S are inverted and applied to the translator (fig. 12-13). The output from $^{43}S07$ is a "1" to indicate the selection of drive line 7. The "1" output is inverted to "0" output by $^{50}Y19$ (fig. 12-11), the corresponding Memory control driver. The "0" output from $^{50}Y19$ drives the positive and negative enabler circuits, $^{56}Y27$ and $^{57}Y27$. The operation of the enabler is similar to that described for $^{56}Y33$ and $^{57}Y33$ (used for X-secondary selection).

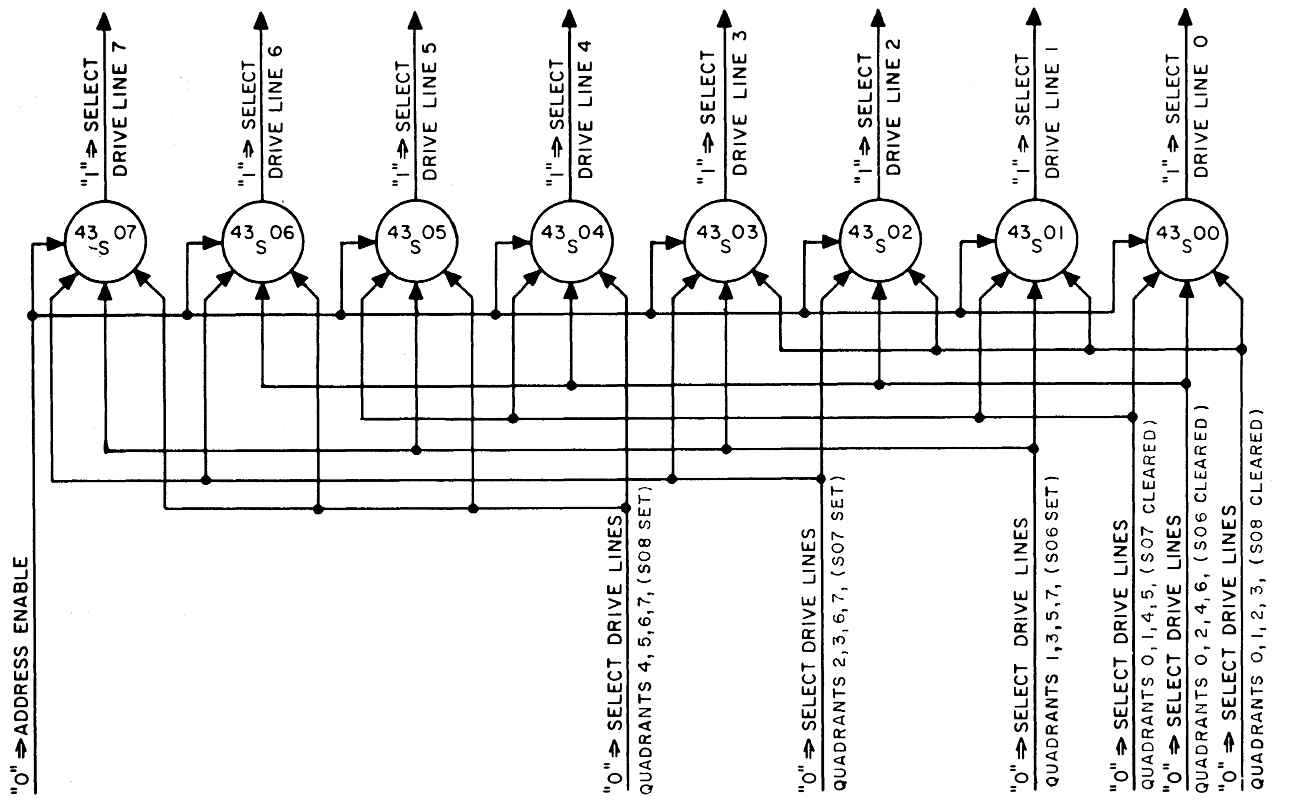
The outputs from the enablers are connected to 32 diode pairs on the end board assembly (one diode pair for each X secondary). Recall that each secondary connects to eight drive lines. The diode pairs are enabled in the same manner as the diodes on $^{55}Y13$ used for X-secondary selection (discussed earlier). One of the 32 diode pairs connects (through six memory boards) to one of eight drive lines in the group enabled by the selected secondary. Through this action the desired X-drive lines are selected, which pass through memory cores.

In the example cited, line 7 of the group is enabled. Enablers $^{57}Y27$ and $^{56}Y33$ provide circuit continuity for the READ operation and $^{56}Y27$ and $^{57}Y33$ provide circuit continuity for the WRITE operation.

Summarizing the selection of the X drive lines when referencing address 13700 in memory is thus accomplished by first selecting the X primary winding from one of eight X primaries, 0 through 7. The combination of inputs from S^{11}, S^{13}, and S^{14} (see table 12-1 and figure 12-10B) are correct at the X primary translating circuits $^{47}S00$ through $^{47}S07$ (fig. 12-10A) to select primary 0 which drives secondaries associated with quadrants 0 and 1. Thus it is determined (by the translator circuits) that the address to be referenced lies in either quadrant 1 or quadrant 2. Second, the desired secondary is selected by X-secondary selector circuits which receive inputs from S^{10} and S^{9} (table 12-1 and figure 12-10B). These inputs are translated by the $^{44}S00$ through $^{44}S03$ X secondary translation circuits, one of which produces a 1 output (for any given reference) to indicate the selection of a particular X secondary. The "1" output from $^{44}S03$ thus enables secondary 3.

Finally, the particular X drive line which must be referenced to enable read or write from address 13700 is selected by sampling the condition of stages S^{08}, S^{07}, and S^{06}, and translating the sampled inputs in the X line selector translator circuits $^{43}S00$ through $^{43}S07$ (fig. 12-13). This process establishes a condition which enables one of the eight drive lines connected to the selected secondary. The line terminates in a diode pair. The diode pair provides an easy conduction path for the line selected (in this case line 7) while it simultaneously presents a high impedance path for unselected secondaries.

It is now possible to note the complete conduction path from secondary through drive lines to the diode pairs on the end board. Assume that the 4-5 winding (secondary 0) on $^{54}Y10$ is selected. The selection of this (or any) secondary is accomplished by the low-impedance-to-ground


"1" → SELECT DRIVE LINE 7
"1" → SELECT DRIVE LINE 6
"1" → SELECT DRIVE LINE 5
"1" → SELECT DRIVE LINE 4
"1" → SELECT DRIVE LINE 3
"1" → SELECT DRIVE LINE 2
"1" → SELECT DRIVE LINE 1
"1" → SELECT DRIVE LINE 0
43S07
43S06
43S05
43S04
43S03
43S02
43S01
43S00
"0" → ADDRESS ENABLE
"0" → SELECT DRIVE LINES QUADRANTS 4, 5, 6, 7, (S08 SET)
"0" → SELECT DRIVE LINES QUADRANTS 2, 3, 6, 7, (S07 SET)
"0" → SELECT DRIVE LINES QUADRANTS 1, 3, 5, 7, (S06 SET)
"0" → SELECT DRIVE LINES QUADRANTS 0, 1, 4, 5, (S07 CLEARED)
"0" → SELECT DRIVE LINES QUADRANTS 0, 2, 4, 6, (S06 CLEARED)
"0" → SELECT DRIVE LINES QUADRANTS 0, 1, 2, 3, (S08 CLEARED)

124.163

Figure 12-13.—Translation of X-line selector.

path supplied by one of the negative and positive enabler circuits as discussed. These enablers are made conductive following the selection of the desired X-transformer primary. Thus all four secondaries produce a secondary voltage as a result of mutual inductance.

If secondary 0 is to be selected (as assumed) enablers 57Y30 and 56Y30 receive inputs which permits the low-impedance-to-ground condition to exist at the output of these circuits. This produces a low impedance path to ground from one side of secondary 0 (at terminal 5) via the diodes on 55Y10 (ground not shown).

Pending on X drive line specification input to any of the circuits 50Y12 through 50Y19, the associated negative and positive enabler circuits will be activated. If the enablers for line 0 are activated, the circuit path from terminal 4 on secondary 0 will be completed via the centerboard (which connects to all eight drive lines in a quadrant) through the memory board drive lines (six memory boards on a chassis) through the diode pairs for line 0 to ground (at enabler outputs). Thus the eight X drive lines associated with secondary 0 are enabled.

X DRIVE LINE CONTROL

When the primary, secondary, and line selector enables are present, a memory timing control circuit (discussed later) initiates the read or write pulses in the primary winding of 54Y10. The pulses are coupled via the selected secondary winding to the centerboard. The secondary current pulses flow from the secondary through the centerboard, and through the X drive line enabled by the line selector.

Appropriate Y drive lines must be enabled in order to reference a specific address. Like the X drive lines, the selected Y drive lines are threaded through cores to be selected on six of the twelve memory boards on any one chassis. For example, for the selection of address 13700, the X drive lines enable cores in quadrants 0 and

1 on the six memory boards as discussed. Before the final selection of a specific core in any one quadrant on a memory board can be made, the Y drive lines concerned must also be enabled.

Y SELECTOR

The Y primary selector, secondary selector, and line selector circuits (not shown) function in essentially the same manner as the corresponding X selector circuit just described. The primary difference is the stages of S (see table 12-1 and figure 12-10B) concerned with the Y selector translations. The Y primary selector circuits receive inputs from S^{14}, S^{12}, and S^{05}; the Y-secondary selector received inputs from stages S^{04} and S^{03}; and the Y line selector inputs are from stages S^{02}, S^{01}, and S^{00}.

One of the Y drive lines enabled on a given quadrant will intersect an X drive line enable on that same quadrant. For example, in the previous discussion regarding the selection of address 13700, X drive lines on quadrants 0 and 1 were enabled. By performing similar translations for the Y selectors for the selection of address 13700 (as must be done) the Y drive lines enabled will be found to be threaded through cores in quadrants 1 and 3. (The process of selecting the quadrants and drive lines for Y selection is not explained.) Because, in this case, both the X and Y drive lines enabled are in quadrant 1, their intersection is in quadrant 1.

From the above example it must be understood that on any one memory chassis the intersection of the X and Y drive lines would take place on six memory boards in quadrant 1. This method selects six bits of the specified 30 bit word on each memory chassis. On say chassis 1, bits 0 through 5 are selected; on chassis 2, bits 6 through 11 are selected; on chassis 3, bits 12 through 17 are selected; on chassis 4, bits 18 through 23 are selected; and on chassis 5, bits 24 through 29 are selected.

Any address from 00000_8 through 77777_8 can be translated in the manner explained to enable the specified X and Y selectors and to indicate the quadrant of intersection on each memory board.

INHIBIT SELECTOR

An inhibit input (fig. 12-2A) is applied on an inhibit line to a core which is already receiving X and Y inputs to prevent that core from changing state. The inhibit selector circuits (shown later) which receive inputs from stages S^{14}, S^{13}, and S^{12} (see table 12-1 and figure 12-10B) are a group of circuits that select and enable specific inhibit windings in the 30 selected quadrants on 30-memory boards. The translation circuits, $^{46}S^{00}$ through $^{46}S^{07}$ (fig. 12-14) designate by their outputs, an inhibit winding in a specific quadrant.

For example, for selection of address 13700, the contents of the inhibit selector stages (see table 12-1 and fig. 12-10B) are S^{14} = "0", S^{13} = "0", and S^{12} = "1". Supplied with these inputs, the translator output (fig. 12-14) from $^{46}S^{01}$ is a "1". This "1" output indicates an inhibit winding in quadrant 1 is to be enabled.

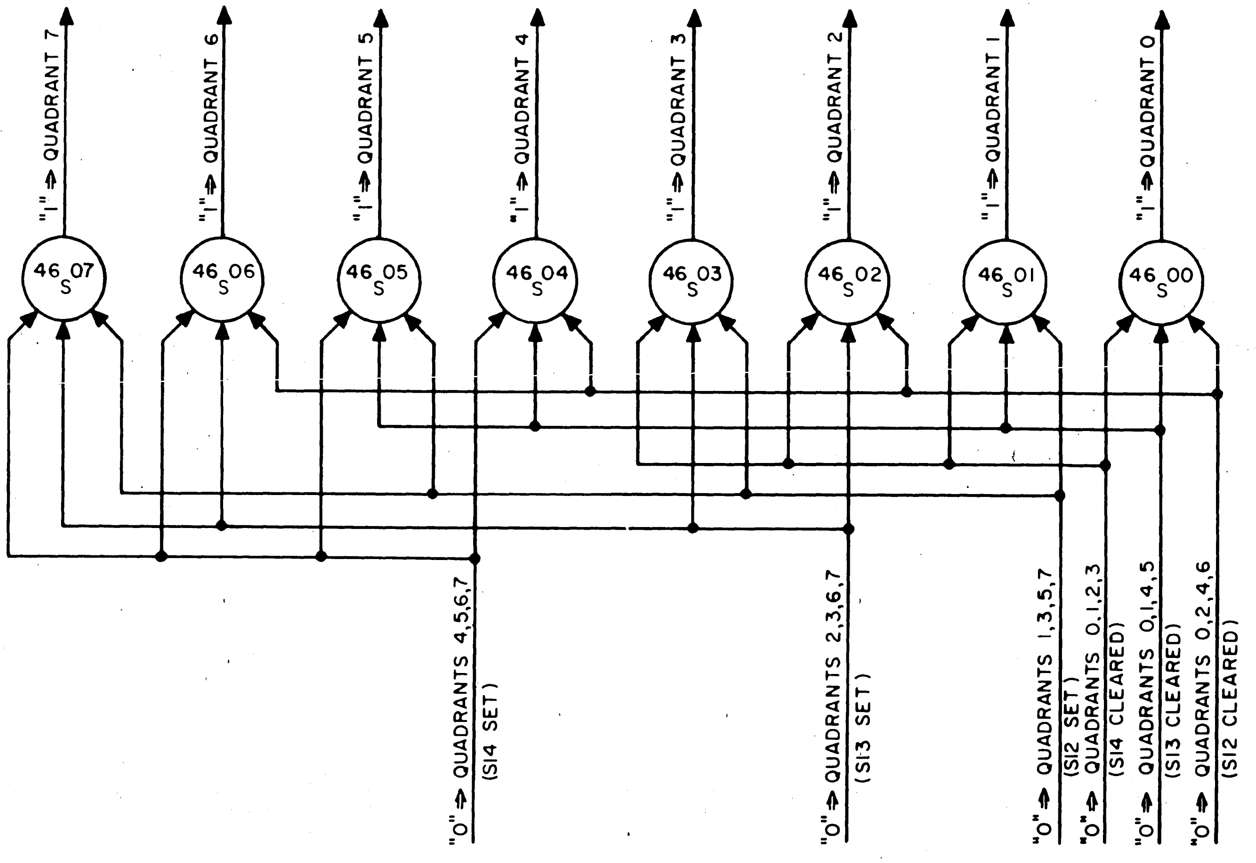
Recall that the intersection of the X and Y drive lines for address 13700 takes place in quadrant 1. If an inhibit pulse is required in any given quadrant during the write cycle, a current is established in the inhibit line in that quadrant.

There are eight inhibit enablers (enabler circuits not shown) used on each chassis. Each one is used to enable the corresponding inhibit line in each quadrant on each memory board.

The procedure to establish an inhibit line in an enabled state does not produce an inhibit pulse on the enabled line. The same inhibit line is enabled in a given quadrant on all memory boards on each of the five memory chassis a total of 30 lines. The generation of the inhibit pulse is a function of the Ƶ-register (described later). If "0" is to be written into memory in a given bit position, bit position Ƶ$^{--}$, representing that position, will produce a "0", which, in turn, allows an inhibit pulse on the associated previously enabled inhibit line. Although an inhibit pulse may not be required, the 30 inhibit lines are enabled during address selection prior to the read/write cycle.

DATA CONTROL

Communication between the Z-register (fig. 12-1) and the addressed memory cell is regulated by data control. Information is detected in a given bit of the selected memory register by sampling the remanent magnetization of the selected core. A net full amplitude Read pulse, impressed on the intersecting drive lines, attemps to switch the core to "0". If the core contained a "1", the flux reversal, caused by the Read pulse polarity, induces a relatively large voltage on the sense winding (see fig. 12-2A). When a read memory reference is specified, this



"1" → QUADRANT 7
"1" → QUADRANT 6
"1" → QUADRANT 5
"1" → QUADRANT 4
"1" → QUADRANT 3
"1" → QUADRANT 2
"1" → QUADRANT 1
"1" → QUADRANT 0
46 07 S
46 06 S
46 05 S
46 04 S
46 03 S
46 02 S
46 01 S
46 00 S
"0" → QUADRANTS 4,5,6,7 (S14 SET)
"0" → QUADRANTS 2,3,6,7 (S13 SET)
"0" → QUADRANTS 1,3,5,7 (S12 SET)
"0" → QUADRANTS 0,1,2,3 (S14 CLEARED)
"0" → QUADRANTS 0,1,4,5 (S13 CLEARED)
"0" → QUADRANTS 0,2,4,6 (S12 CLEARED)

124.164

Figure 12-14.—Translation of inhibit line selector.

induced voltage pulse is amplified and strobed (gated at the time of maximum voltage amplitude) into the corresponding stage of the Z-register where it is retained for future use. Since the contents of the selected memory register are strobed unconditionally into Z during a read memory reference, the Z-register must be cleared at the start of every memory reference.

During a write memory reference (where data is to be written into memory) the data is loaded into the previously cleared Z-register for subsequent storage in a selected memory register. The Write pulse attempts to write "1" in every core in the selected memory register. The contents of corresponding stages of the Z-register are used to determine if an inhibit is generated for a given bit to provent the "1" from being written.

The output from each stage of the Z-register (fig. 12-15) is connected to one side of the eight inhibit windings in an associated memory plane (connections to inhibit lines are not shown). The desired inhibit winding in a given quadrant of each memory plane is enabled by the inhibit translation of the upper three bits of the S-register as explained in the discussion of the inhibit selector circuit. If the content of any stage of the Z-register is a "0", an inhibit pulse is allowed on the selected inhibit winding only to nullify the effect of the write pulse. If Z contains a "1", the inhibit pulse is inhibited allowing the full effect of the write pulse.

Data control, being involved in the transfer of information out of and into the selected memory register, serves its function (memory reference) in any communication between memory and the other computer sections. A memory reference is initiated to sample a word (read) to store a word (write), or to perform a combination of these functions (read/write). The same cycle of operation is used within memory for each of the three cases - a Read pulse followed by a

Write pulse. The difference is in the manner in which the Z-register is used; whether it receives data from memory for use and rewriting a memory or from other sources for writing into memory.

Z-REGISTER

The Z-register functions as a buffer between the memory section and the arithmetic, control, and input/output sections of the computer. All information, going to or coming from memory, passes through this register. The Z-register is a 30-bit register that functions at the bit-plane level; i.e., specific stages in Z control, specific planes in memory. For some commands the Z-register is divided logically into the upper 15-bits (Z_u) and the lower 15-bits (Z_L). The division into Z_u and Z_L is controlled by the k or $\hat{k}$ designator.

Z-Register Inputs

All stages of the Z-register (fig. 12-15) have inputs from memory (main storage), wired memory, the D-register, and 14 I/O channels (discussed in chapters 13 and 14). In addition, Z_L (the lower 15 bit positions of Z) has inputs from the P-adder. The various inputs are numbered 1 through 7 in figure 12-15).

The inputs from main memory is controlled by a strobe pulse (discussed later) that is initiated by a memory timing chain. The inputs from the D-register, I/O channels, and the P-adder are controlled by command enables generated in their respective circuits.

Each stage in Z is associated with eight sense windings (fig. 12-16), one winding for each of eight quadrants in a given plane. The signal from the activated sense winding (only one winding in each of 30 bit planes is sensed for each memory reference) is amplified in a sense amplifier circuit. Because only one reference is made from a bit plane at any given instant, the lines which carry the amplified sense signals for the same bit plane are tied together at the output of the sense amplifiers.

The amplified sense signals are applied to NOR elements (not shown) in the sense output circuits. A strobe pulse and either Z_L and/or Z_u command enable signals are also applied to the input of the sense output circuits. If the three input signals are "0" in coincidence, the sense output circuits produce a "1" output which is applied to the corresponding stage in Z causing the SET condition to be established in tha stage. Note that the input to Z is from memory (the number 4 input shown in figure 12-15).

If the wired memory mode is active, each stage in Z receives an input from the wired memory addresses (the first 16_{10} addresses in memory). The wired memory contains the basic instructions which are necessary to permit other instruction (in a program to be executed) to be read in (automatically) into the main memory. Thus, the wired memory is, in itself, a program or routine. The use of wired memory to load instructions into computer memory is sometimes called "bootstrap" loading.

The inputs to Z from the D-register can be a half-word length (15 bits) or a full word length (30 bits) operand, as determined by the value of the k-designator. The command enables $D_L \longrightarrow Z_L$ and/or $Du \longrightarrow Zu$ are generated during the D sequence.

The inputs to Z from any of the I/O channels are 30-bit words. The command enables are generated in an I/O timing chain (not shown).

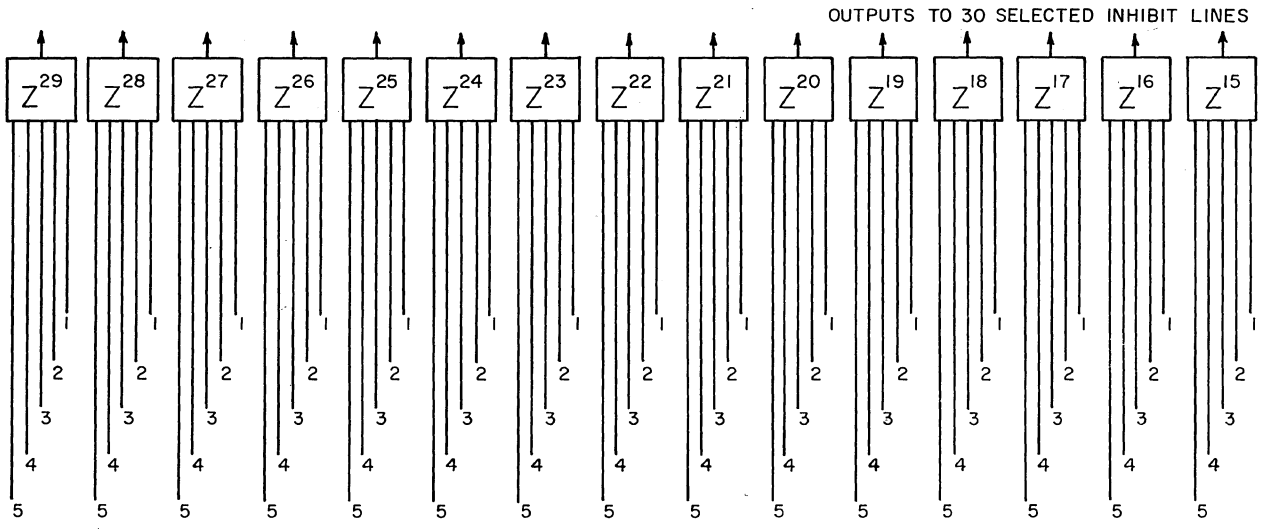
The input to Z from the P-adder is used during a return jump instruction to store P (the address of the present instruction) + p (the number by which P is to be advanced) in M_L (the lower 15 bits of the specified memory address) if the jump condition is satisfied. The command enable for the store P + p operation is generated during the D sequence.

Z-Register Outputs

The Z-register (see fig. 8-1) has outputs to all computer sections. In additions to the outputs to memory, each stage of Z has an output to the C_0, and C_1 registers (via gate circuits), to the U, X, and S registers and to the buffer comparator.

The two C-registers (C_0 and C_1) are used in the execution of data transfers from the computer to external equipments. The C register transmits data to other computers and the C_0-register transmits data to external equipments.

The Z output (30 bit words) to C_0 and C_1 are used during normal output buffer operations as controlled by the I/O timing chain (discussed in a later chapter). The transmissions are controlled by the D sequence.



INPUTS FROM
1. INPUT/OUTPUT 4. MEMORY (MAIN STORAGE)
2. INPUT/OUTPUT 5. WIRED MEMORY
3. D-REGISTER 6. P-ADDER
7. P-ADDER

A Z $\longrightarrow$ U operation occurs during the A sequence. This operation loads an instruction (30 bits) into the U-register. Afterwards, the instruction (having been previously read from memory) is executed by the computer to carry out a specified function. The instruction read from memory in a current A sequence is the instruction to be executed in the normal sequence of operation, i.e., if no skip, jump or other conditions occur.

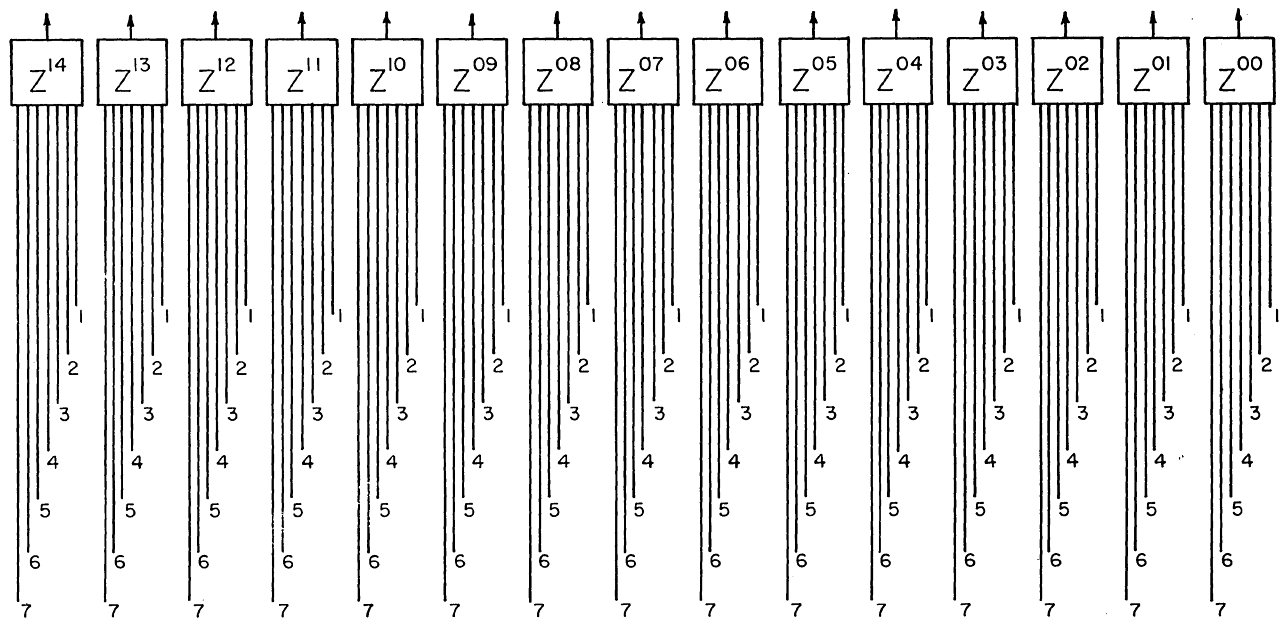
A Z $\longrightarrow$ X operation transfers the operand into the X-register where the operand is complemented, shifted, etc. (if any of these operations are to be performed) before it is transferred into the D-register. The Z $\longrightarrow$ X transmission (which occurs during the B sequence) transfers 30 bits into X. Any modification to the operand is made during the X $\longrightarrow$ D transmission.

The data transmission from Z into S is only a 15-bit transfer ($Z_L \longrightarrow S$). This transfer is used during a specific input/output operation to specify the storage address for the current buffer operation. The $Z_L \longrightarrow S$ transmission is disabled during the real-time clock (RTC) mode wherein the RTC address is transferred directly into S.

Z-Register Counters

Two 15-bit Z-register counters, (see fig. 8-1) one for Z_u and one for Z_L, are used to count and increment the memory addresses when a buffer operation to input data from an I/O channel is in progress. (Buffers and buffer controlled operations are discussed in chapter 7.)

The actual incrementing of the Z-register counters is performed under the control of the I/O section. When I/O initiates a memory reference during a normal buffer operation, the Z_L counter is enabled. This causes the number of the selected memory register (address) to be incremented by one after the storage of each data word from the I/O channel. The action increases the special buffer storage address by one in anticipation of storing the next data word. The Z counters are also used for updating the Real time clock address.



124.165

Figure 12-15.— -Z-register, simplified diagram.

When the X-register is used to transfer data into storage from any computer section, the Z-register counters are disabled and the data in Z is transferred into memory without alteration.

TIMING CONTROL

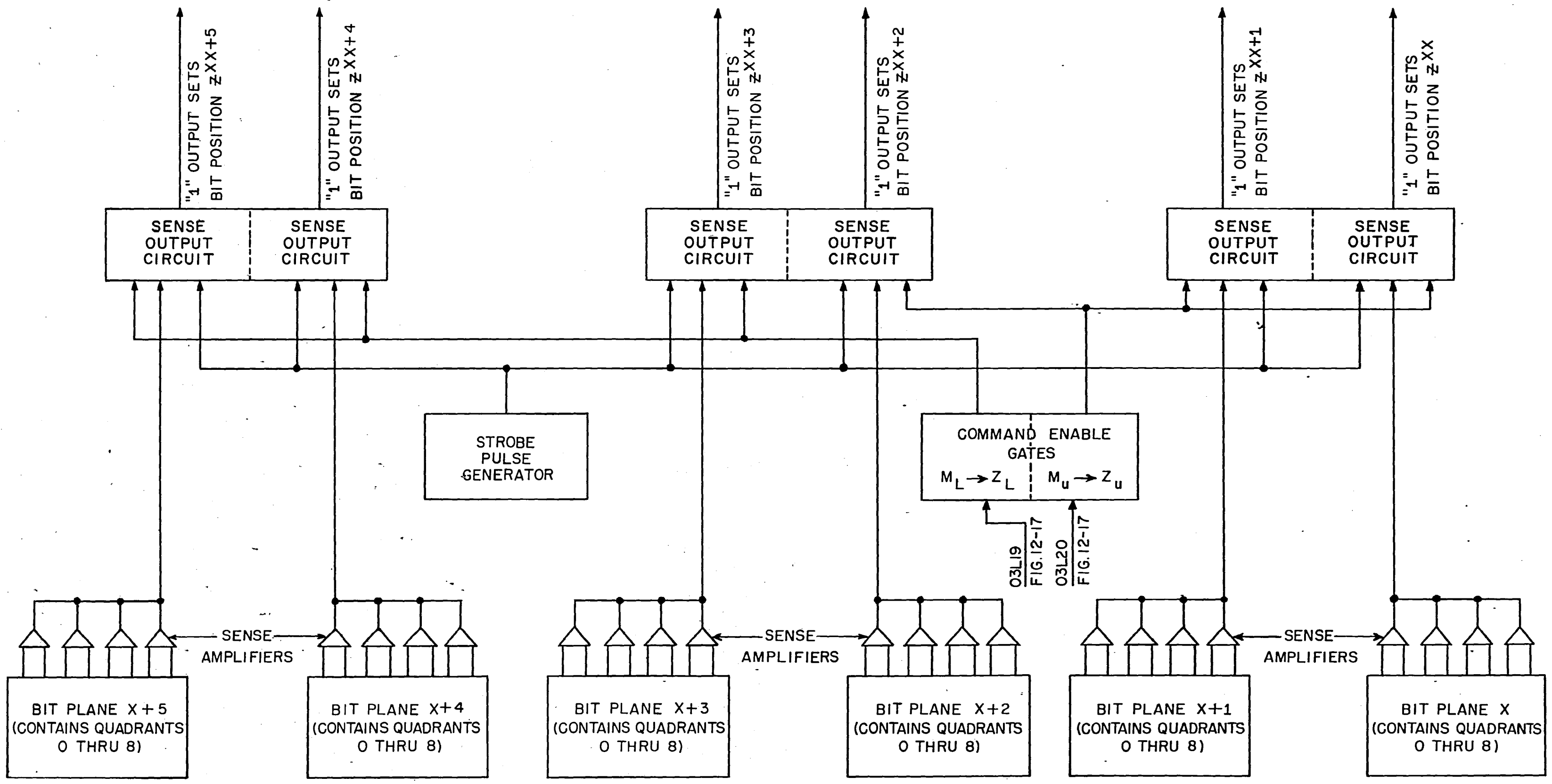
The generation of control signals for the operation of the memory section must occur in a fixed time relationship. This timing relationship is controlled by the Memory timing chain (fig. 12-17). The timing chain provides enable signals for the clearing of the S-register, for the data transmission from memory to Z (read cycle), and for the data transmission from Z to memory (write cycle).

The basic memory cycle time is 8.0 microseconds (fig. 12-18). This is the minimum time between consecutive memory references.

One clock cycle is subdivided into seven sections each of which is made up of the four clock phases (see figure 10-4). Thus to reference a given instant in a cycle (fig. 12-18) it is necessary to know the phase time (1, 2, 3, or 4) and the portion of the cycle time (times 0 through 6). The numbers used to indicate phase time should be interpreted to mean that a particular phase is enabled, such as phase 1, phase 2, etc. The actual time or period of each phase is 0.4 μsec as discussed in chapter 10.

In referencing specific times within a cycle we will adhere (as is generally done) to the following procedure. Phase 2, time 0 is denoted by the upper left-hand pulse in figure 12-18. Phase 4, time 0 is denoted by the leading edge of the left-most pulse on the "clear S" line. Phase 2, time 5 is denoted by the leading edge of the right-most pulse on the "initiate memory" line. In writing, these intervals are denoted as; Ø2.0, Ø4.0, and Ø2.5, respectively. The fifth time Ø2 comes up it is designated as Ø2.5.

A memory reference is initiated when one (or more) of the signals applied to ^{14}L11 (fig. 12-17) is "1". The "0" output (from ^{14}L11) is inverted to "1" by ^{17}L11 which serves as an enable signal to cause the S-register to be cleared and to prevent the setting of the I/O flip-flop. The latter action prevents memory


"1" OUTPUT SETS BIT POSITION ZXX+5
"1" OUTPUT SETS BIT POSITION ZXX+4
"1" OUTPUT SETS BIT POSITION ZXX+3
"1" OUTPUT SETS BIT POSITION ZXX+2
"1" OUTPUT SETS BIT POSITION ZXX+1
"1" OUTPUT SETS BIT POSITION ZXX
SENSE OUTPUT CIRCUIT
STROBE PULSE GENERATOR
COMMAND ENABLE GATES
$M_L \rightarrow Z_L$
$M_u \rightarrow Z_u$
03L19 FIG. 12-17
03L20 FIG. 12-17
SENSE AMPLIFIERS
BIT PLANE X+5 (CONTAINS QUADRANTS 0 THRU 8)
BIT PLANE X+4 (CONTAINS QUADRANTS 0 THRU 8)
BIT PLANE X+3 (CONTAINS QUADRANTS 0 THRU 8)
BIT PLANE X+2 (CONTAINS QUADRANTS 0 THRU 8)
BIT PLANE X+1 (CONTAINS QUADRANTS 0 THRU 8)
BIT PLANE X (CONTAINS QUADRANTS 0 THRU 8)

124.166

Figure 12-16.—Memory control and sense circuits.

references which may be present from an input/output channel from interfering with memory references already initiated. The "0" output from $^{14}L^{11}$ is also used (via $^{13}L^{11}$) as an enable signal to flip-flop L^{11} to start the memory timing chain.

The functions enabled by the timing chain with respect to clock cycles are listed as follows:

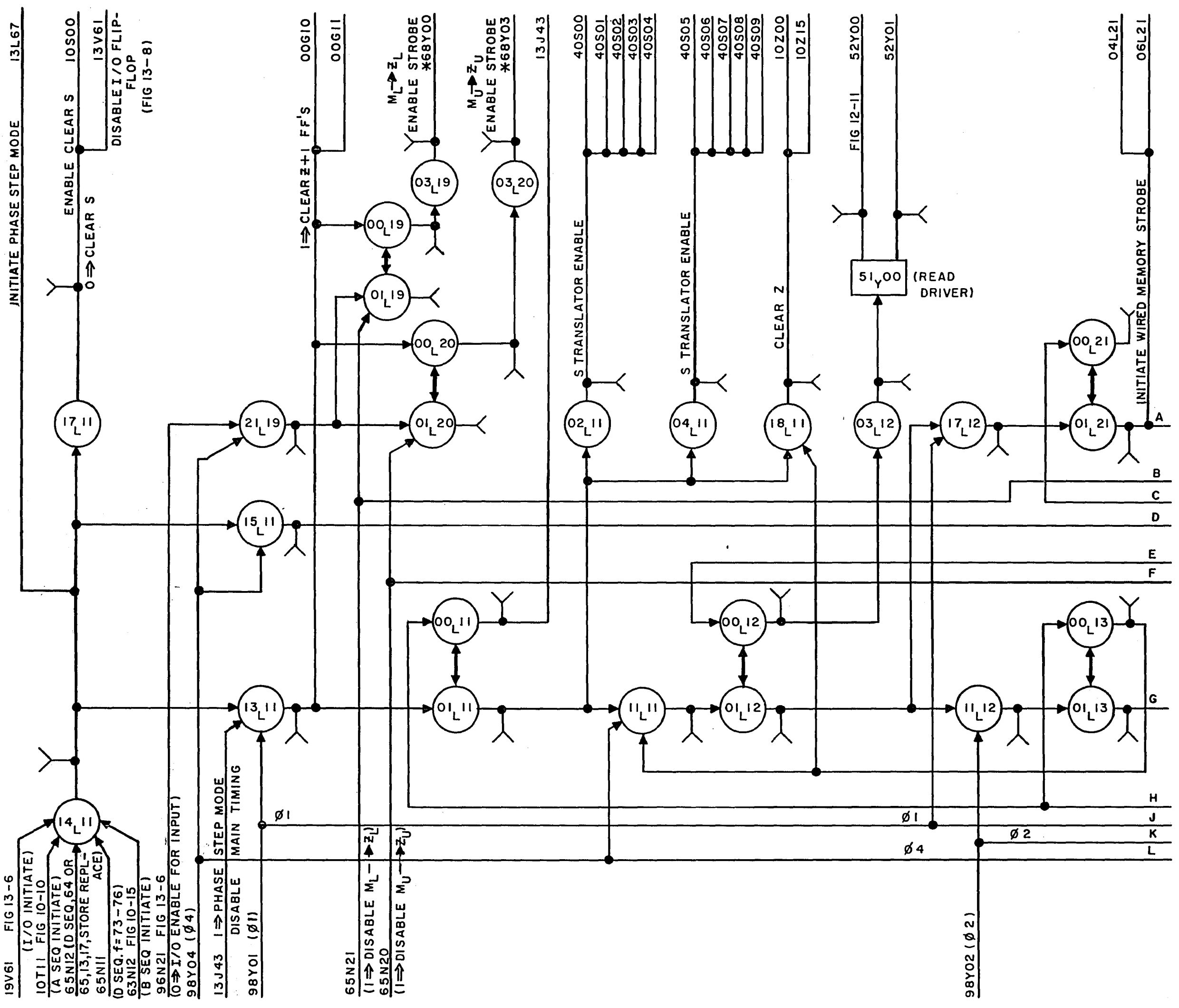
Ø2.0	Initiate memory	-Enable clear S
Ø4.0	Clear L^{17} (via $^{15}L^{11}$)	-Set memory lockout (eliminates possibility of any circuit referencing memory other than via the memory timing chain).
Ø1.1	Set L^{11} (via $^{13}L^{11}$)	-Enable S translator in preparation for a memory reference. Also enables clear Z.
Unconditional commands	Clear L^{19} and L^{20} (via $^{13}L^{11}$)	-Enable strobe M $\rightarrow$ Z
	Clear G^{10} and G^{11} (not shown) via $^{13}L^{11}$	-Disable Z counter
Conditional commands	Set L^{70} Set L^{71}	-Disable $M_L \rightarrow Z_L$ -Disable $M_u \rightarrow Z_u$ } D sequence
Ø4.1	Set L^{12}	-Enable read (M $\rightarrow$ Z)
Ø1.2	Set L^{21}	-Initiate Strobe
Ø4.3	Set L^{15}	-Enable inhibit
Ø1.4	Set L^{16}	-Enable write (Z $\rightarrow$ M)
Ø3.4	Set L^{17}	-Memory available
Ø2.5	Set L^{18}	-Enable disturb
	Initiate Memory	

PHASE STEP MODE

The computer must be in high speed operation before a memory reference can be made. The memory cycle (read/write) must operate at high speed to function properly. If memory is initiated when the computer is in the PHASE STEP MODE (explained in chapter 10), the memory timing chain will not be enabled ("1" into $^{13}L^{11}$ disables timing chain). All of the functions enabled by the timing chain are also disabled. None of the memory enables required for the read/write cycle are generated. Thus, if desired, the contents of any selected memory register must be entered manually into the Z-register.

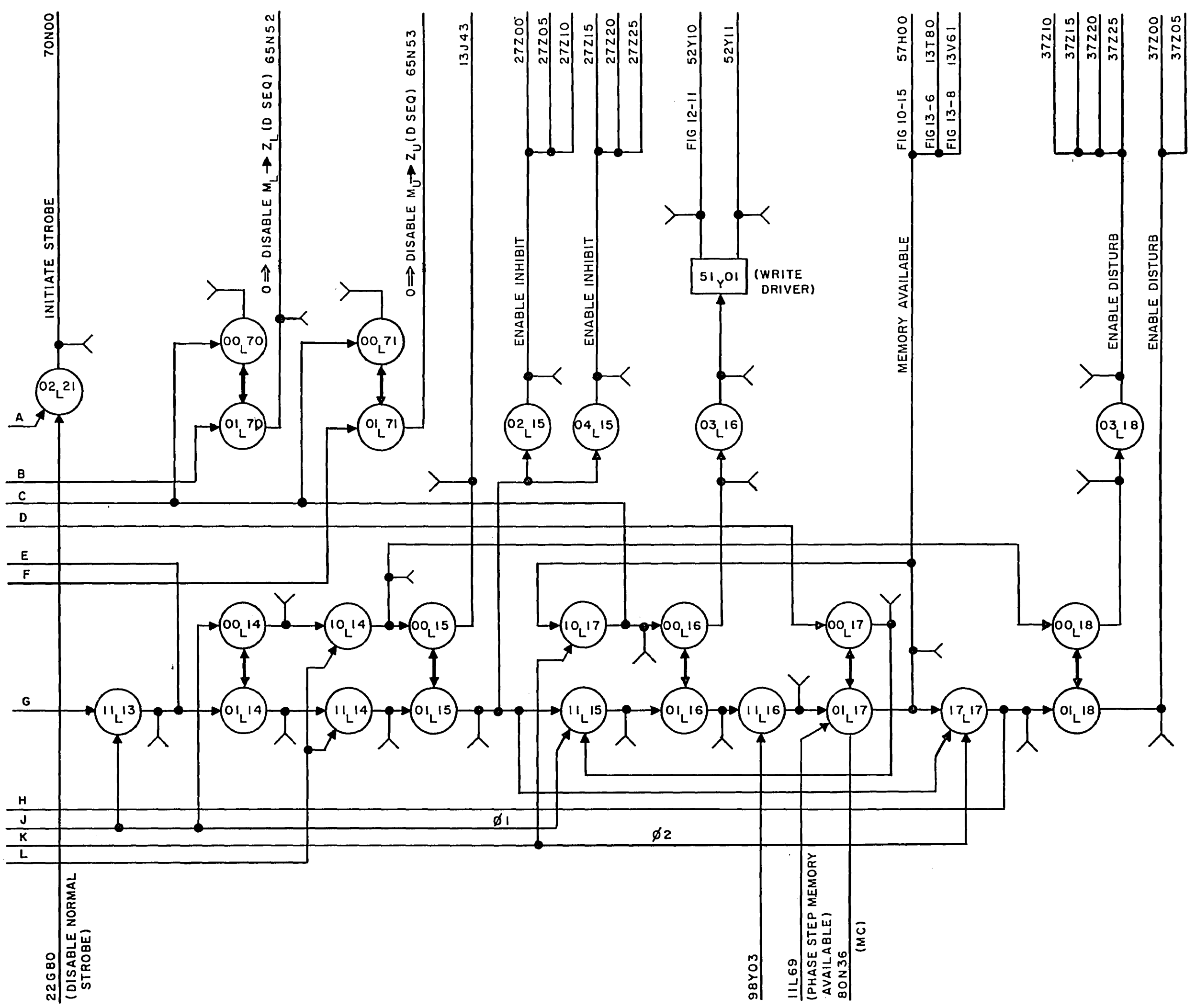
MEMORY INITIATION

Memory can be initiated by the A, B, and D sequences and by an I/O channel. During the A sequence the next instruction to be executed (or the first instruction of an interrupt subroutine) is read out of memory. Once memory has been



initiated, the Memory timing chain (fig. 12-17) starts to cycle. The address in memory to be referenced is transmitted to S, either P adder → S or Interrupt address → S. During the read cycle the instruction located at the specified address is read out of memory into Z. It is then transmitted from Z to the U-register for execution. During the write cycle of the timing chain (which automatically occurs after each read cycle to restore the original contents of the register from which the data is read) the instruction is written back into memory, unaltered, at the same address from which it was read.

During the B sequence the operand for all instructions is read out of memory if k≠ 0, 4, or 7 (see table 12-2 for interpretation of k-designator). Memory is initiated by a "1" output from $^{63}N^{12}$ in the B sequence (fig. 10-15) to $^{14}L^{11}$. The (fig. 12-17) U-adder S transmission specifies the memory address from which data must be obtained in order to execute the present instruction. The output from that address is read



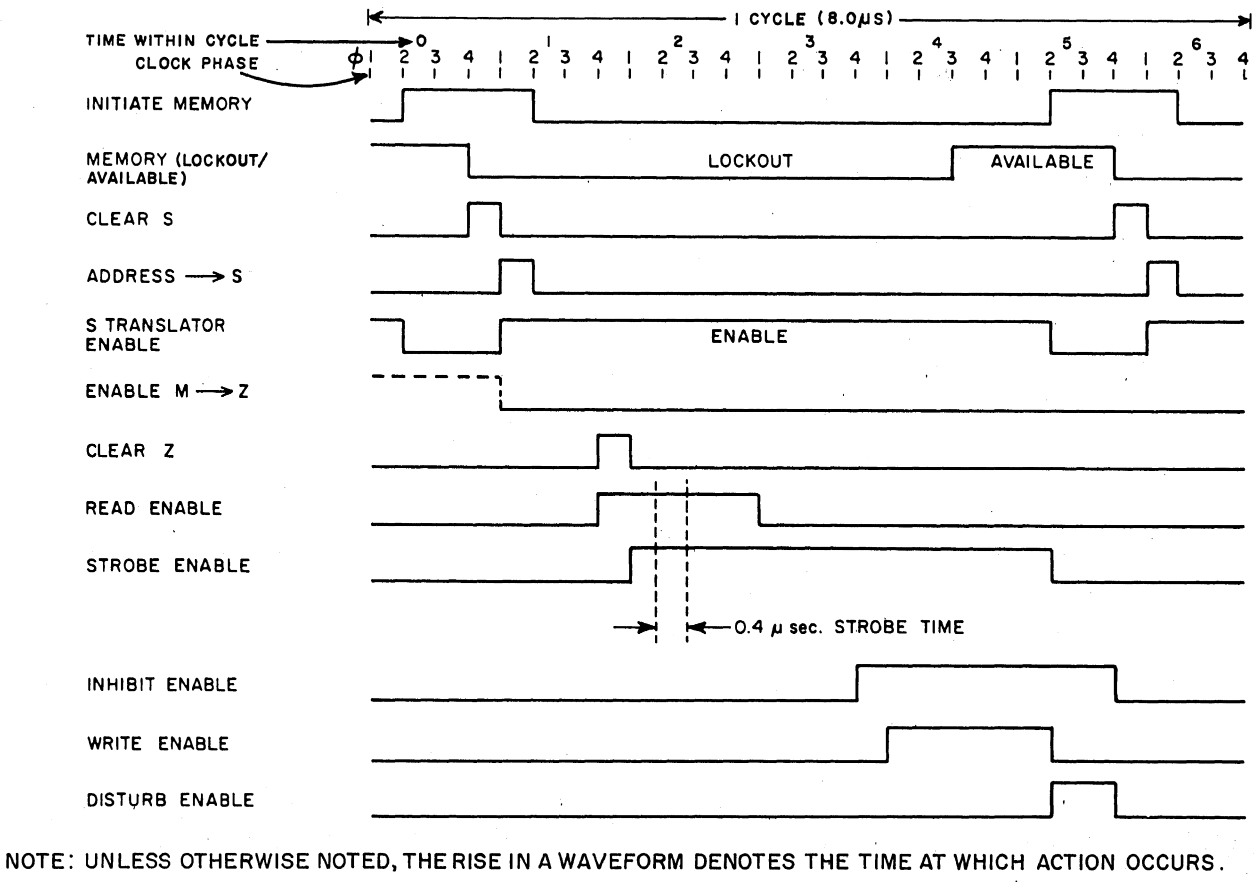
124.167

Figure 12-17.—Memory timing chain.

into the Z-register. It is then transmitted to the X-register and then to the D-register, for modification as specified by the k-designator.

During the D sequence the manner in which memory is used, is conditioned by the function code and the value of the k or $\hat{k}$ designator. During the Read cycle (see table 12-2) the information read out of memory into Z is conditioned by the value of k. If k = 1 or 5, L^{70} (fig. 12-17) is set and the transmission of $M_L \rightarrow Z_L$ is disabled, allowing only $M_u \rightarrow Z_u$ to be executed ($Z_L = 0$, $Z_u = M_u$). If k = 2 or 6, L^{71} is set and the transmission of $M_u \rightarrow Z_u$ is disabled, allowing only $M_L \rightarrow Z_L$ to be executed ($Z_L = M_L$, $Z_u = 0$). If k = 3 or 7, both L^{70} and L^{71} are set and the transmission of both $M_u \rightarrow Z_u$ and $M_L \rightarrow Z_L$ is disabled ($Z_L = 0, Z_u = 0$). In all cases all memory is Read but various paths are disabled.

For all the instructions except f = 17 (see table 10-2) the information to be stored is held in the D-register. Whenever a transmission from



124.168

Figure 12-18.—Memory timing sequences.

memory to Z is disabled, a transmission from D → Z is enabled.

During the Write (Store) cycle the information in Z is written back into memory. If k = 1 or 5, M_u is left unchanged and new information is entered into M_L ($D_L \rightarrow Z_L$) ($Z_L \rightarrow M_L$). If k = 2 or 6, M_L is left unchanged and new information is entered into M_u ($D_u \rightarrow Z_u$) ($Z_u \rightarrow M_u$). If k = 3 or 7, a new 30-bit word is entered into memory ($D_L \rightarrow Z_L$, $D_u \rightarrow Z_u$) and ($Z_L \rightarrow M_L$, $Z_u \rightarrow M_u$).

If an f = 64 or 65 instruction is executed, (see table 10-2) and the jump condition is satisfied, the contents of the P-adder (P + p) must be stored in M_L. This action stores the address specified by (P + p) for future use after the completion of a subroutine. The storage address is transmitted to S (U-adder → S). During the Read cycle, the transmission of $M_L \rightarrow Z_L$ is unconditionally disabled (L^{70} set) and the transmission of $M_u \rightarrow Z_u$ is unconditionally enabled (L^{71} cleared). The contents of the P-adder are transmitted to Z_L. During the Write cycle, the information in Z is stored in memory.

WIRED MEMORY

The computer uses an auxiliary memory wired memory) to store a Bootstrap load routine. This short-load routine is used to enter a comprehensive (more elaborate) load routine into core memory. This procedure bypasses the need for a time consuming manual load operation.

Wired memory consists of the first 16_{10} core. memory addresses (00000_8 through 00017_8),

There are three Bootstrap load routines; A, for paper tape; B, for intercomputer operations, and C, for magnetic tape systems.

The Bootstrap load routine is wired into a set of adapter plugs (not shown). The adapter plugs

Table 12-2.—Interpretation of k-Designator.
(See specifically the interpretations for k = 0, 4, or 7 in Read, Store, or Replace instructions.)

READ instructions (01 through 13, 20 through 23, 26, 27, 30, 31, 40 through 43, 50 through 53, and 60 through 76 see table 10-2):

$\hat{k}$ or k = 0:	Y_u = 0's; Y_L = y. (Operand in U_L)
$\hat{k}$ or k = 1:	Y_u = 0's; $Y_L = (Y)_L$. (Operand in M_L)
$\hat{k}$ or k = 2:	Y_u = 0's; $Y_L = (Y)_u$. (Operand in M_u)
$\hat{k}$ or k = 3:	Y = (Y). (Operand in M)
k = 4:*	Y_u = same bits as Y_{14}; Y_L = y. (Operand in U_L)
k = 5:*	Y_u = same bits as Y_{14}; $Y_L = (Y)_L$. (Operand in M_L)
k = 6:*	Y_u = same bits as Y_{29}; $Y_L = (Y)_u$. (Operand in M_u)
k = 7:	Y = (A). (Operand in A register)

For instructions 23, 52, and 53, k = 7 is not used.

For instruction 13, only $\hat{k}$ = 3 is permitted.

For instructions 73 through 76, k = 2 is not used.

STORE instructions (14 through 16, 17, 32, 33, and 47 see table 10-2):

k = 0:	Store in Q.
k = 1:	Store in M_L leaving M_u undisturbed.
k = 2:	Store in M_u leaving M_L undisturbed.
k or $\hat{k}$ = 3:	Store in M.
k = 4:	Store in A.
k = 5:	Store complement in M_L, leaving M_u undisturbed.
k = 6:	Store complement in M_u, leaving M_L undisturbed.
k = 7:	Store complement in M. (Storing the complement of B_j uses the same procedures as for a 30-bit register.)

For instruction 17, only $\hat{k}$ = 3 is permitted.

REPLACE instructions (24, 25, 34 through 37, 44 through 46, and 54 through 57 see table 10-2):

k = 0:	Not used.
k = 1:	Read portion, Y_u = 0's; $Y_L = (Y)_L$. (Operand in M_L) Store portion, stores in M_L, leaving M_u undisturbed.
k = 2:	Read portion, Y_u = 0's; $Y_L = (Y)_u$. Store portion, stores in M_u, leaving M_L undisturbed.
k = 3:	Read portion, Y = (Y) Store portion, stores in M.
k = 4:	Not used.
k = 5:*	Read portion, Y_u = same bits as Y_{14}; $Y_L = M_L$. Store portion, stores complement in M_L, leaving M_u undisturbed.
k = 6:*	Read portion, Y_u = same bits as Y_{29}; $Y_L = M_u$. Store portion, stores complement in Mu, leaving M/L undisturbed.
k = 7:	Not used.

must be rewired according to wire tabulations (not shown) in order to change the Bootstrap routine. To insure that the adapters used are all for the same bootstrap routine, interlock protection is provided on each memory chassis. The operation of the Wired Memory mode is controlled by the AUTOMATIC RECOVERY switch (fig. 10-9) on the computer console.

SPECIAL CIRCUITS

In the memory section the operation of the drive current transformers, the generation of read/write pulses, the enabling of the sense circuits, and the generation of inhibit and disturb pulses are accomplished by special circuits (nonstandard logic circuits). Since the Memory section of the computer uses a magnetic core memory, the special circuits are mainly used to generate a current pulse or to provide a low impedance path for current.

READ/WRITE CIRCUITS

The Read/Write circuits operate in an identical manner except that Read current flows in the opposite direction to Write current. The purpose of both Read and Write circuits is to allow a current pulse to flow through the primaries of a selected X and Y transformer.

For an example refer to figure 12-11 and assume a selection for X primary 0. The "1" output from ${}^{47}S^{00}$ (fig. 12-10) is inverted to a "0" output by ${}^{50}Y^{32}$ (fig. 12-11). This output drives Q5 on ${}^{53}Y^{10}$ into conduction as discussed earlier. The collector potential of Q5 decreases. However, the base of Q1 and Q3 remains positive with respect to the emitter and the transistors remain cutoff.

During the Write cycle of a memory reference, L^{16} (fig. 12-17) in the Memory timing chain is set. The "0" output from ${}^{03}L^{16}$ causes ${}^{51}Y^{01}$ to produce a "0" output to ${}^{52}Y^{10}$ (fig. 12-11). This action cuts off Q6 in ${}^{52}Y^{10}$. When Q6 is cutoff, a current pulse is generated as the fields around the coils in the emitter and collector circuits collapse. The collapsing field tends to keep current flowing in the same direction through the 470-ohm resistors as when the transistor was conducting. When Q6 is cutoff, the coil effectively acts like a battery with the positive terminal connected to the emitter of Q3 on ${}^{53}Y^{10}$. The increase in positive potential on the Q3 emitter of ${}^{53}Y^{10}$ allows the transistor to conduct. When Q3 conducts, the 15-14 primary windings of the drive current transformer #0 has a low-impedance path to ground via the relatively low conducting resistance of transistor, Q3.

The current diverter circuit acts as a constant current generator for the period of the current pulse. The pulse is coupled across into the secondary winding of the transformer.

The Read circuit operates in the same manner except that the other primary (the 12-13 winding) of the transformer is used and current flows in the opposite direction in the secondary windings.

The generation of a Read/Write pulse in any selected X and Y drive line transformer is accomplished in an identical manner when the required enables are present.

SENSE CIRCUITS

The sense circuits (fig. 12-16) pass the outputs from the memory stacks to the Z-register as discussed. Each sense circuit connects the output from the eight sense windings in a bit-plane (two memory boards) to a stage in the Z-register. Only one sense winding on a memory plane is enabled for each memory reference.

A representative sense amplifier and sense amplifier output circuit is shown in figure 12-19A. The input to a sense amplifier is from one of the diagonally opposite sense windings on a memory plane (say from quadrant 0 or 3).

When no sense input signal is applied at the base of Q3 and Q4 of ${}^{62}Y^{00}$ (the static condition of the circuit) the output at the CR1-CR2 junction is effectively at ground potential (see simplified diagram, figure 12-19B).

If a core in a memory plane contains a "1", and a READ pulse is applied, the resulting flux reversal in that core induces a low amplitude voltage in the sense winding (see fig. 12-2A). The connections from the sense winding of the two quadrants (fig. 12-19A) are arranged so that the input to the base of Q3 is negative with respect to the base of Q4 when the signal is received from one quadrant and positive with respect to the Q4 base when the signal is received from the other quadrant. Resistors, R13 and R14, are connected in series across the sense winding. Assume that the sense signal drives the Q3 base negative with respect to the base of Q4. This action causes an increase in the Q3 conduction and a decrease in the conduction of Q4. Because of the unequal conduction of the two transistors, the voltage drops across R_9 and R_{10} (in the emitter circuits of Q3 and Q4 respectively) also become

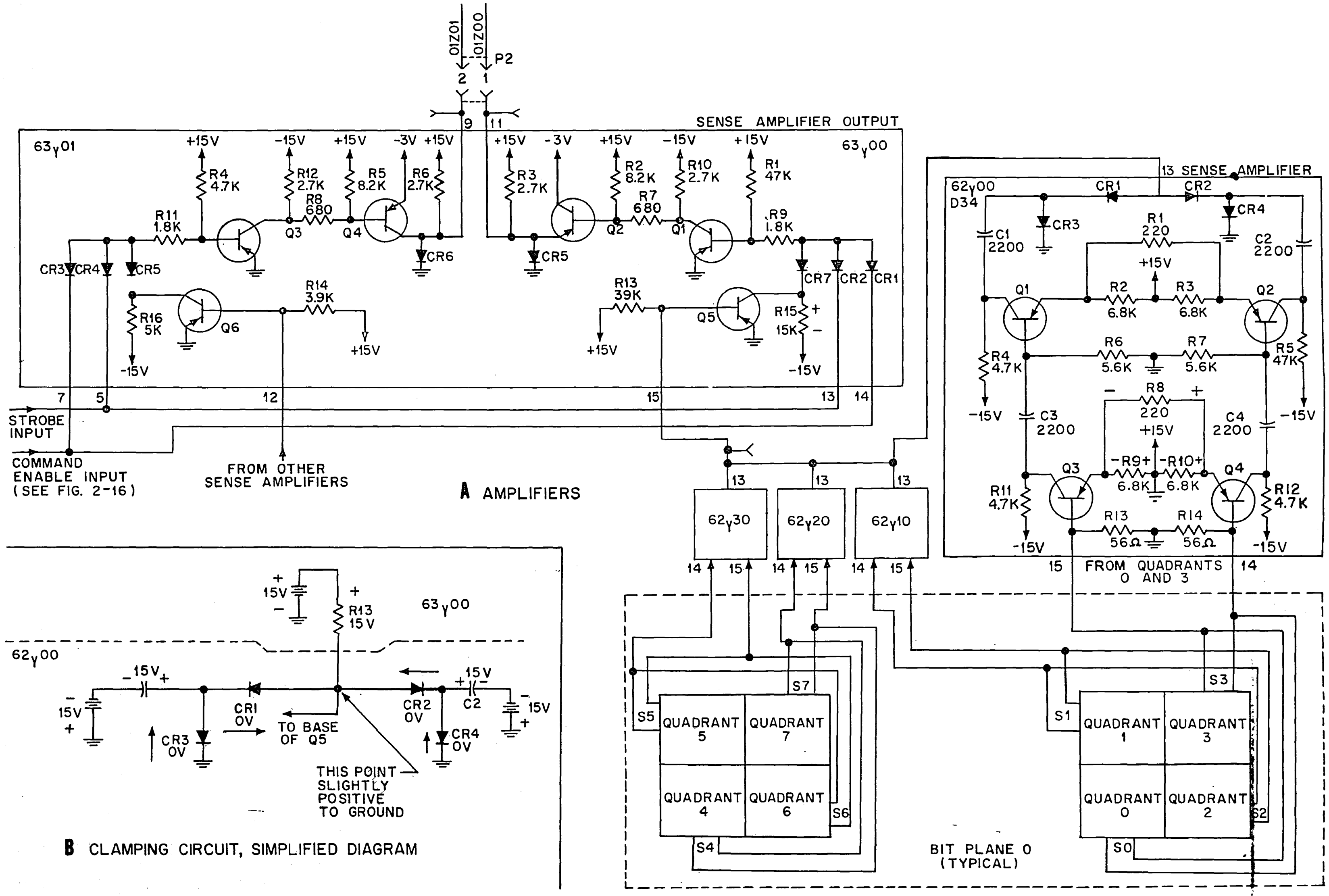


Figure 12-19.—Sense amplifiers.

124.169

unequal. Thus, a net voltage is established across R_8 which is negative toward the Q3 emitter. Because the Q3 emitter current is now conducted through parallel paths (R_9 in one path and R_8 and R_{10} in series in the other) to the +15 volt supply it follows that the Q3 emitter-to-ground resistance is lowered. This decreased emitter-to-ground resistance allows Q3 to conduct more heavily, and the increasing voltage across R_8 (positive toward the Q_4 emitter) biases Q4 further toward cutoff. The collector potential of Q3 rises in a positive direction while the collector potential of Q4 decreases in a negative direction. The signal input between base and emitter of each of the amplifiers is thus amplified and inverted at the collector of each of the transistors.

The collector voltage changes of Q3 and Q4, respectively, are coupled to the base of Q1 and Q2, respectively. The circuit operation of Q1 and Q2 is exactly the same as that described for Q3 and Q4. Thus, Q2 conducts heavily and Q1 is driven toward cutoff.

Any positive voltage at the collector of Q2 is clamped at ground potential by the action of CR4. The negative voltage change at the collector of Q1 is coupled to the CR1 -CR3 junction. Diode CR1 is forward-biased by this negative-going pulse and the resultant current is conducted from the emitter to collector of Q1, through C1, CR1, the Base-emitter resistance of Q5 (in the sense amplifier output circuit, $^{63}Y^{00}$) to ground, the +15 volt supply (which has the negative terminal at ground potential not shown) and through R2 paralleled by the series connection of R3 and R1, to return to the Q1 emitter.

The current change between the base and emitter of Q5 ($^{63}Y^{00}$) serves as the amplified sense signal input to the sense amplifier output circuit. This input signal drives Q5 into conduction. (Transistor Q5 is normally cutoff by a near-ground potential established at the CR1-CR2 junction of $^{62}Y00$.)

Before the sense amplifier output circuit can be enabled, two additional enable signals (the M $\rightarrow$ Z and strobe enable signals) must be applied. The following discussion assumes that these inputs are present at the cathodes of CR2 and CR1, respectively, of $^{63}Y00$.

Transistor, Q1, in the sense amplifier output circuit ($^{63}Y^{00}$) is normally conducting as a result of a negative base potential tapped from the voltage divider connected between the -15 and +15 volt supplies and comprising R15, CR7, R9 and R1. The conduction of Q5 ($^{63}Y^{00}$) applies a near-ground potential at the CR7-R15 junction, and the potential at the R9-R1 junction goes positive to ground. This action cuts off Q1, thus causing its collector potential to become more negative. The negative voltage swing is applied (from a voltage divided similar in operation to that just described) between base and emitter of Q2 ($^{63}Y^{00}$). The negative-going voltage swing taken from emitter-to-ground of Q2 serves as the sense amplifier output.

If a core held a "0" and a read pulse were applied, the output from the sense amplifier circuit would be the same as under static conditions. The sense amplifier output circuit would produce a "0" input.

As aforesaid, in order to fully enable the sense amplifer output circuit, $^{63}Y^{00}$, the M $\rightarrow$ Z enable and the strobe enable must be present. During a memory reference if the $M_U \rightarrow Z_U$ and/or $M_L \rightarrow Z_L$ transmissions are enabled, L_{20} and/or L_{19} (fig. 12-17) are cleared. The "1" output from $^{03}L^{19}$ and/or $^{03}L^{20}$ drives the command enable gates in figure 12-16. The gates are conventional inverters that are driven into saturation. Their "0" outputs are applied as enable inputs to the sense amplifier output circuits (fig. 12-19).

Whenever the three inputs to the sense amplifier output circuit are "0's" simultaneously a "1" is loaded in the corresponding stage of the Z-register. If any one or all of the input signals are "1", a "0" is loaded into the corresponding stage of Z.

INHIBIT CIRCUITS

An inhibit enable is generated prior to the write cycle of any memory reference, (when L^{15} in the memory timing chain of figure 12-17 is set). The output from the S-translator provides the signal to select the inhibit line in a specified quadrant to inhibit the inhibit pulse. Six inhibit lines are capable of being enabled on each of the five memory chassis during the writing of each word. Thus, 30 inhibit lines are selected in this manner, one for each bit position of the word.

Whenever an inhibit pulse for a specified bit is applied on the inhibit line, the corresponding bit position in memory stores a "0". When a "1" is to be written in memory, no inhibit pulse is required.

CHAPTER 13

NTDS COMPUTER, INPUT/OUTPUT SECTION (PART I)

As explained in earlier chapters, the CP-642A/USQ-20(V) computer operation is controlled by a stored progress capable of operational-modification. Each program instruction contains a function code (6 bits), instruction operand designator (15-bits), and three execution modifiers (3 bits each). Execution modifiers provide for address incrementation, operand interpretation, and branch-point designation. The operand may be increased by the amount contained in any one of seven index registers. The operand specified by the execution address may be interpreted as a 30-bit quantity, or as a 15 bit half-word with or without sign extension.

Communication between the computer and its associated external equipment is normally handled by a block transfer of data with overall timing under control of the external device due to its low speed. Operating asynchronously with the main computer program, such transfers of data have independent access to storage.

A communication path is established by a sequence of request and acknowledge signals between external equipment and computer. Such signals may originate in either the computer or the external device. The main computer program is interrupted by external request signals and a communications channel is established. Once the link has been created, the computer returns to the main program sequence. Block transfer of input or output data then proceeds without program reference until completed.

A total of 14 input and 14 output channels is provided in the computer; each data channel consists of 30 parallel lines. Two input and two output (special) channels are reserved only for communications with other computers. The maximum possible transfer rate of input or output data over a given channel is greater than 30,000 words per second. Output channels carry External Function Words as well as data words to external equipment. These specify data words to external equipment. These specify the function desired of the external device. An External Function Word to a tape control unit, for example, may specify ''Rewind Tape Unit 2.''

The Input/Output section includes those data paths and control circuits used by the computer for communicating with external equipment. The main parts of the Input/Output section are (1) two output registers (C_0 and C_1, see figure 8-1) and their associated line drivers, (2) 14 sets of gated input amplifiers, (only 3 shown in figure 8-1) and (3) priority and access control circuits.

The C_0 register is used for transmissions to all external devices except other computers. The C_0 register receives its input directly from the storage section via gates controlled by the priority and access circuits (control lines not shown in figure 8-1). Three sets of 30 line drivers branch from the output of C_0; each set drives four output channels. Gated registers located in the external devices sample (by priority control) on the channel which is active during any particular transmission.

The C_1 -register handles transmissions over the two special output channels,; i.e., those used to transmit data to other computers. Operation of C_1 is similar to that of C_0. Words enter C_1 from storage via gates controlled by the priority and access circuits and are transmitted over the active channel by a set of 30 line drivers.

The output channels are numbered from 0 to 13. If two or more output transmissions are simultaneously requested, the channel with the highest number is granted priority; others follow in order.

A set of 30 gated amplifiers is provided for each of the 14 input channels. Gates are controlled by the priority and access circuits (control lines not shown) with the channel having the highest number being given priority if two or more inputs are simultaneously requested. As aforestated, Channels 0 and 1 are used for intercomputer communication, and consequently receive the lowest priority. This method of treating input data eliminates the need for input buffer registers (on which data would be intermediately stored before its transmission to or from memory) and gives external equipment direct access to the computer's internal memory.

Some functions of the priority and access circuits (namely, gating input and output transmissions and assigning priorities to the channels) have been previously described. In addition to the actions already described, the priority and access control circuits accept and transmit the control and timing signals which must be exchanged between the computer and the equipment with which it communicates. The priority and access control circuits also include a means of testing various channels to determine whether they are busy. This feature prevents the computer from attempting to communicate over channels already in use.

Main Memory addresses referenced during a particular input or output transfer are determined by a special I/O control word. A signal generated by sensing the I/O control word is used by the priority and access circuits to deactivate the channel after the proper number of words has been transferred.

DATA AND CONTROL SIGNALS

The transfer of data to and from the computer is accomplished by the I/O section utilizing 14 separate input and output channels. (A brief description of the input/output section is treated in chapter 8.) The data is received from and transmitted to external equipment in a 30-bit parallel mode. Channels 2 through 13 (see table 13-1) communicate with any external equipment except other computers. Channels 0 and 1, classified as special channels, handle the data transfer to and from external computers.

Data input to the computer from any external equipment is through input amplifiers directly into the z-register. Along with the data, control signals are also transmitted from the computer to the external equipment and to the computer from the external equipment.

The computer is designed to use a d-c level type input and output. Signals are represented as d-c levels that may be changed upon interchange of control information. Signals may exist for microseconds or days, depending on the nature of the particular task. The control lines are carried in the same cables as the data lines and have the same voltage levels; hence, delay times, rise and fall times, and storage times are similar.

Each control and data line uses a twisted pair to carry the signals to and from the computer. The return side of the twisted pair is grounded in the computer and in the external equipment, thereby minimizing noise pickup on the signal lines.

Each input and output channel has its own separate cable associated with it (28 cables in all). Each cable has 30 data lines plus three of the four possible control lines. Figure 13-1 shows the computer receiving input data from equipment B and sending output data to equipment A. Table 13-1 gives the channel assignments for the external equipment.

Note the direction of information flow in figure 13-1. The Data Request signals are always sent from the external equipment to the computer. The Acknowledge signals are always sent from the computer to the external equipment. The third set of control signals, called Interrupt signals in the input cables and External Function signals in the output cables, are always sent in the same direction as the data flow.

COMMUNICATION

The sequence of events for three cases of communication between the computer and the external equipment follows.

Buffer Mode (Input)

The normal input sequence procedures for data transfer from an external equipment to the computer (Buffer modes) are:

a) Computer initiates input buffer for a given channel.

b) External equipment places data word on 30 data lines (in preparation for 30-bit parallel transfer).

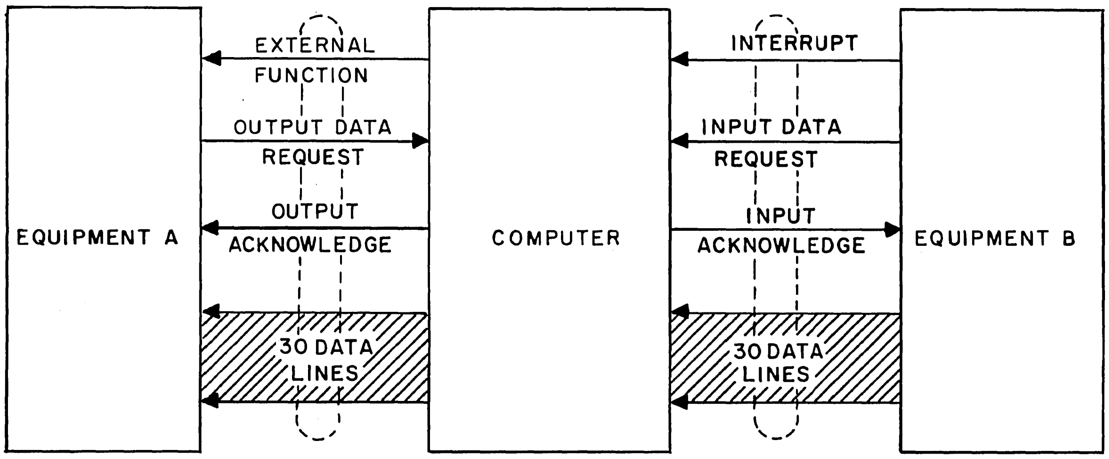
c) External equipment sets the Input Data Request line to indicate that it has data ready for transmission to the computer.

d) Computer detects the Input Data request.

e) Computer samples (reads-in) the information on the 30 data lines at its own convenience.

f) Computer sets the Input Acknowledge line indicating that it has sampled the data.

g) External equipment senses the Input Acknowledge line (this indicates to the external equipment that the Computer has received the word).



124.170

Figure 13-1.—Interconnections computer to external equipment for input and output.

Table 13-1. — Channel Assignments

CHANNELS		EQUIPMENT
0		COMPUTER
1		COMPUTER
2	AS ASSIGNED	KEYSET CENTRAL
3	AS ASSIGNED	DISPLAY UNITS
4	AS ASSIGNED	PAPER TAPE
5	AS ASSIGNED	SYSTEM MONITORING PANEL
6	AS ASSIGNED	MAGNETIC DRUM (if available)
7	AS ASSIGNED	VIDEO PROCESSOR
8	AS ASSIGNED	IDAC or FLEXOWRITER
9	AS ASSIGNED	MAGNETIC TAPE
10	AS ASSIGNED	TELETYPEWRITER
11	AS ASSIGNED	"A" LINK
12	AS ASSIGNED	"C" LINK (if available) or HIGH-SPEED PRINTER
13	AS ASSIGNED	USC-2

h) External equipment drops the data lines and the Input Data request line (voltage on these lines change from enabled to disabled).

Events b) through h) of this sequence are repeated for every data word until the number of words specified in the input buffer has been transferred.

Interrupt

Sequence for external equipment transmitting an Interrupt code to the computer:

a) External equipment places the Interrupt code on the 30 data lines.

b) External equipment sets the Interrupt line.

c) Computer detects the Interrupt signal.

d) Computer samples the 30 data lines.

e) Computer sets the Input Acknowledge line indicating that it has sampled the data.

f) External equipment senses the Input Acknowledge line.

g) External equipment drops the Interrupt code from the data lines and drops the Interrupt line.

Note that the Input Acknowledge signal is the computer response to either an Input Data request or to an Interrupt. In order to eliminate misinterpretation of the Input Acknowledge signal, external equipment must not interrupt until its last Input Data request has been acknowledged by the computer. Under emergency conditions when data loss is of secondary importance, the Input Data request may be dropped and the Interrupt raised a minimum of 100 microseconds later. When these conditions prevail, an Input Acknowledge signal that occurs after the Interrupt is raised, is in answer to the Interrupt.

Buffer Mode (Output)

Normal output sequence for data transfer from computer to external equipment (Buffer mode):

a) Computer initiates output buffer for given channel.

b) External equipment sets the Output Data request line indicating that it is in a condition to accept data.

c) Computer detects Output Data request.

d) Computer places information on the 30 data lines at its convenience.

e) Computer sets the Output Acknowledge line indicating that data lines are ready for sampling.

f) External equipment detects the Output Acknowledge signal.

g) External equipment may drop Output Data request any time after detecting Output Acknowledge signal.

h) External equipment samples the 30 data lines.

i) Computer drops Output Acknowledge signal.

Events (b) through (i) of this sequence are repeated for every data word until the number of words specified in the output buffer has been transferred.

FUNCTION CONTROL

External Function codes are carried over the same 30 lines that are used for output data, but the control signals used with External Function codes are carried in control lines which are in addition to those which carry the request or acknowledge signals.

Sequence for computer transmitting an External Function code to external equipment:

1. Computer places the External Function code on the 30 data lines.
2. Computer sets the External Function line.
3. External equipment detects the External Function line.
4. External equipment samples the 30 data lines.
5. Computer drops the External Function line.

INTERCOMPUTER DATA TRANSFER

Communication between two computers takes place using the two special channels reserved for this purpose. Figure 13-2 illustrates the connections for computer A to transmit data to computer B. Another cable using a special output channel of computer B and a special input channel of computer A would be necessary if computer B were to transmit data to computer A.

Sequence of events for normal transfer of data from computer A to computer B (Buffer mode):

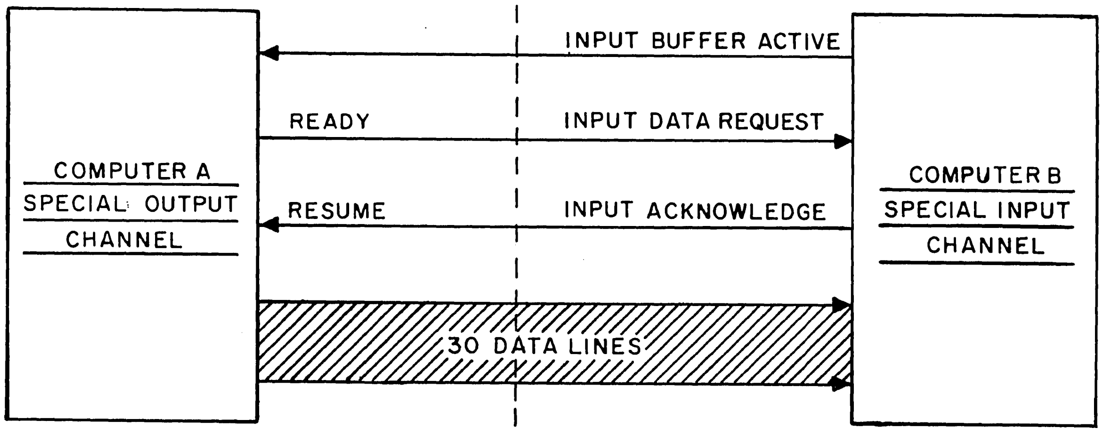
1. Computer B sets Input Buffer Active signal indicating computer B is available to receive data.
2. Computer A detects Input Buffer Active signal, and is now aware that computer B can receive a data word input.
3. Computer A places data on 30 data lines.
4. Computer A signals "Ready" which becomes Input Data request in computer B.
5. Computer B detects Input Data request.
6. Computer B samples 30 data lines.
7. Computer B sets Input Acknowledge line which is returned to computer A as Resume (this releases computer A from previous word).
8. Computer A senses Resume line.
9. Computer A drops Ready line.

Events 3 through 9 of this sequence are repeated for every data word. Input buffer active remains energized during transfer of an entire block of words. Since data transfer between computers presents a very complex timing problem, stringent control must be exercised in the establishment and termination of intercomputer buffers to ensure good intercomputer communications. Sequence of control must be followed closely to ensure that data transfer between computers will stay in synchronism.

Interlocks and check points for intercomputer communication are required in two areas: 1) hardware logic, and 2) program logic.

Hardware Logic

The two special input and two special output channels used for intercomputer communications are necessary so that the transmitting computer may exhibit the same control and data timing characteristics as does the receiving computer. The two special output channels use Ready and Resume logic for data-transfer control. The input channels use the normal input



124.171
Figure 13-2.—Intercomputer communications, computer A transmitting to computer B.

Request and input Acknowledge logic for data-transfer control. In addition to these control signals, an Input Buffer Status signal originates in the receiving computer and is determined by the state of the Input Buffer Active/Inactive designator. The transmitting computer can sense the Input Buffer Status signal, thus determining the status of the receiving computer's input buffer. These special channels do not have two of the normal control lines contained in the other input/output channels. The special output channels do not have an External Function Control line since only the computer on the selected channel is active. When used for intercomputer data transfer, the special input channels do not have an External Interrupt Control line.

The C_1 register (see figure 8-1) is incorporated in the computer to service the two output channels reserved for intercomputer communications. All output intercomputer communication is time-shared through this output register. If, after the data is placed on the lines, a Resume signal is not received within 32 to 64 seconds the computer is notified of this condition by an internally generated interrupt to ensure that a communication with the other computer is not held up indefinitely. The interrupt utilizes the special External Interrupt registers for notification of the faulty output channel.

Program Logic

The major problem encountered by the program is synchronization of buffer control in computer-to-computer communications. The control problem arises (1) when intercomputer data buffers are initiated and (2) when the buffers are terminated. Between initiation and termination, the buffers are maintained in synchronism by data control signals.

Initiation of data transfer is the main key to establishing synchronism between computers. Buffer termination characteristics are defined within this initiation process. Limits for the input buffer (first word address and last word address) must be defined before the Input Buffer mode in the receiving computer can be initiated. Establishment of this mode in the receiving computer is a prerequisite for intercomputer data transfer. The receiving computer cannot define the limit addresses of the input buffer because its length is not known by the computer. Therefore, it is necessary that the transmitting computer define these limits for the receiving computer.

The Input Buffer Control register of the receiving computer is programmed so that the first buffer word received from the transmitting computer is entered into the Buffer Control word address of the receiving input buffer control register. In other words, the initial address (the lower-order 15 bits of the Input Buffer Control word is the same as the address of the receiving Input Buffer Control register. The terminal address (the upper-order 15 bits of the Input Buffer Control register) is never equal to the initial address.

The initial address of the receiving computer is set equal to the address of the Buffer control register in the receiving computer so the first word received from the transmitting computer will be stored in the Buffer Control Word Register causing a new control word to be referenced by the receiving computer.

For example, an instruction 75 030 10000 (see the 75 instruction in table 10-2). The 75 initiates an input buffer on $C\hat{j}$ with monitor. In this case channel 0 is cited. At cell 10000 in the receiving computer's memory is a 30 bit word, the lower 15 bits of which define the initial address of the input buffer and the upper 15 bits of which define the terminal address of the input buffer. For this example, address 10000 will contain 00101 00100. Thus, the first word to be received will be stored at address 00100 (which is the buffer control word address for input buffers on channel 0).

The transmitting computer will now send a block transfer of data, the first word of which will be descriptive of the limits of the block of data to be transferred. The receiving computer will store this word at cell 00100 and from this point on it will use cell 00100 as the control cell for the remainder of the input buffer.

Before the transmitting computer can initiate the output buffer in order to read data into the receiving computer, the receiving computer must establish an input buffer mode. The transmitting computer determines this by testing the state of the input buffer status line with an f = 13 instruction (see table 10-2 chapter 10) and $\hat{j}$ value of either 0 or 1, depending on the channel to be sampled. When the receiving computer has met the above conditions and is ready to receive data, the transmitting computer establishes the input buffer control word for the receiving computer. This is the first word of the output buffer. The lower-order 15 bits of this word defines the initial address for data storage and the upper-order 15 bits define the terminating address for data storage. In the example, cited above, the word is stored at address 00100. Thus, the receiving computer buffer area is defined.

Buffer limits for the output buffer of the transmitting computer must also be defined. Initial and final addresses (first word to be transmitted) of the output buffer data are entered into the lower- and upper-order halves, respectively, of the 30-bit Output Buffer Control register. When output buffer limits are defined, the Output Buffer mode can be initiated.

After the buffers have been established, data transfer proceeds under control of the Ready and Resume signals for the transmitting computer and Request and Acknowledge signals for the receiving computer. The names Ready, Resume, Request, and Acknowledge are themselves self-explanatory. When used in connection with computer buffer operations the signals are interpreted as follows:

Acknowledge—Indication of the status of data on the input/output lines. Abbreviated, ACK.

Ready—The condition existing between two computers where the computer initiating the Ready signal has data on the output lines. This signal is further interpreted as an Input Data Request signal by the receiving computer.

Request—A signal originated in the receiving computer demanding a data word from the transmitting computer.

Resume—The signal generated when the receiving computer has sampled its input lines and has generated an Input Acknowledge which is interpreted as the Resume signal by the transmitting computer. As soon as the transfer is complete, each computer is interrupted (as a result of the buffer monitor). The Interrupt routine, associated with the output buffer, establishes an input buffer with the monitor. The Interrupt routine, associated with ^{1} the input buffer, transfers program control to the Executive routine (main program) and does not initiate an output buffer. Only the Executive routine can initiate an output buffer. This entire process is repeated for each transfer cycle.

SPECIAL CIRCUITS

In the I/O section, special circuits (nonstandard logic circuits) are used as line drivers and amplifiers. The line drivers furnish power (signal voltages) for the control and data lines to external equipment. The input amplifiers receive their inputs from the control and data line drivers of the external equipment. When the input amplifier is used on the data line, a gate signal is normally required. When the input amplifier is used on the control line, no gate signal is used. The output from either of the two line drivers is a nominal 0 v-d-c for a binary "1" and a nominal -13 v-d-c for a binary "0". The output from the input amplifiers is -3 v-d-c for a binary "1" and 0 v-d-c for a binary "0".

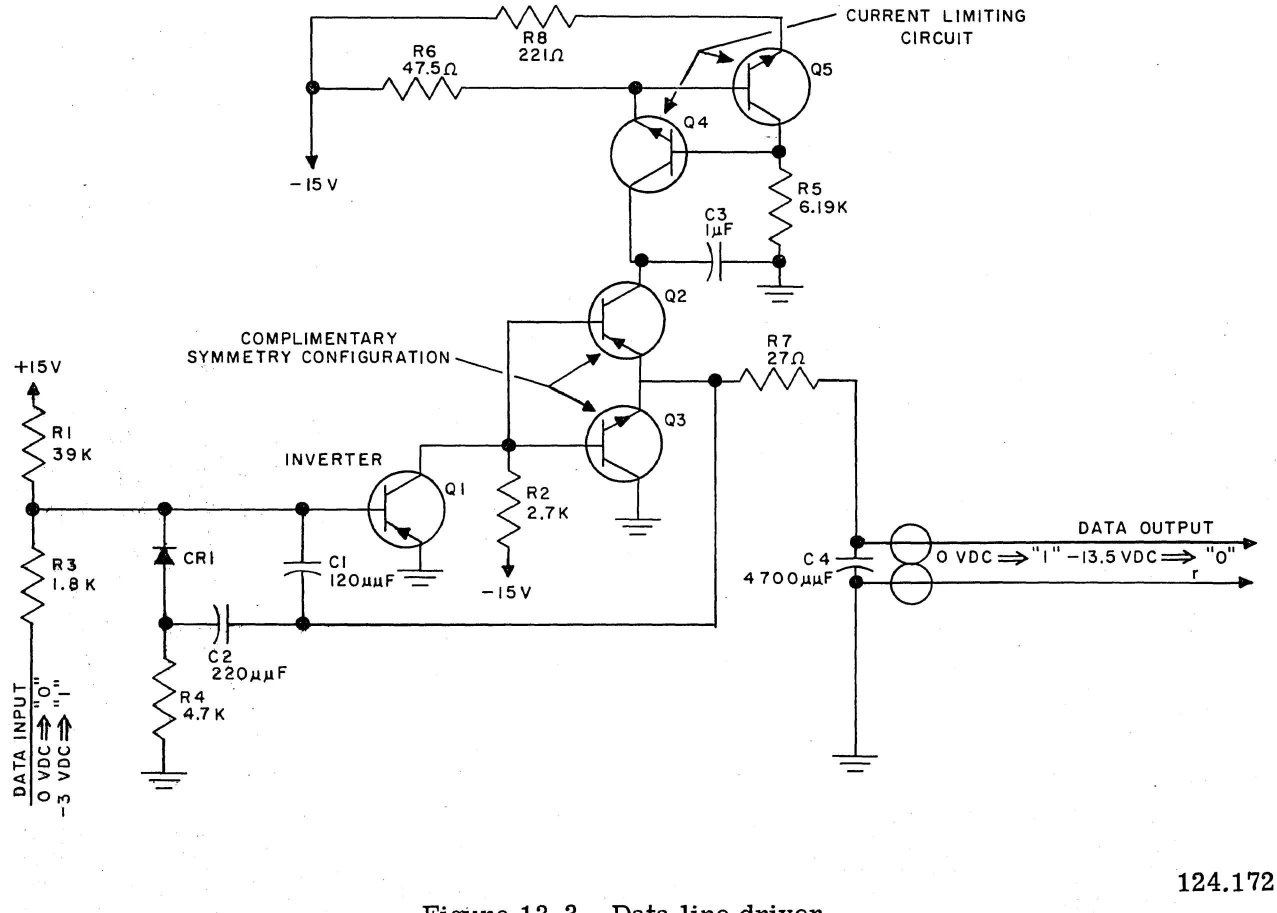
DATA LINE DRIVER

The data line driver (fig. 13-3) transmits data to input amplifiers in external equipment. The circuit consists of one inverter Q1, two transistors in a complementary symmetry configuration Q2 and Q3, and two transistors used as a current limiting circuit Q4 and Q5. The overall circuit provides the drive power required by the data input amplifiers in the external equipment.

The input to the data line driver is 0 v-d-c for the "0" state or -3 v-d-c for the "1" state.

The input is applied to inverter Q1 which drives the complementary symmetry (output) stage. With 0 v-d-c input on the data input line ("0" state) the base of Q1 is held positive with respect to the emitter and transistor Q1 is cut off. The Q1 output voltage is approximately -15 v-d-c which is fed to the base-emitter circuits of the complementary symmetry configuration transistors, Q2 and Q3. Since the base of both transistors becomes negative with respect to the emitter, Q2 (PNP) conducts and Q3 (NPN) is cut off. The Q2 collector is at the Q4 collector voltage of -13.5 volts due to the conduction of Q4 and Q5 and the 13.5 volt drop across R5. The output is approximately -13.5 v-d-c, representing the "0" state at the data output terminal.

When the input to Q1 switches from 0 v-d-c to -3 v-d-c ("1" state), the base of Q1 becomes negative with respect to the emitter. This transistor attempts to conduct saturation current causing the Q1 output to rise toward 0 v-d-c. With the change in base bias, the condition in the output stage (complementary symmetry circuit) switches from Q2 to Q3 (Q3 going into conduction and Q2 going to cutoff). When the output stage switches, there is degenerative



124.172

Figure 13-3.—Data line driver.

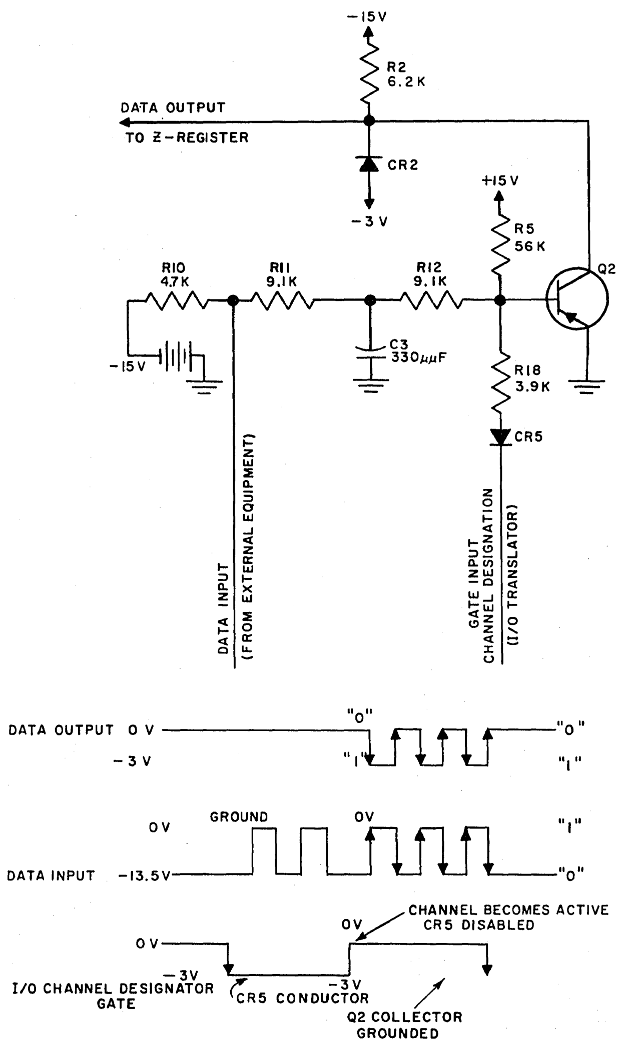
feedback to the input of the inverter from the output stage.

Because the Q2-Q3 output is 0 v-d-c at this time (and going negative) the voltage at the data input terminal (which tends to change abruptly from 0-to -3 v-d-c) is not in complete control to change the base-emitter voltage of Q1. The degenerative feedback permits the Q1 collector current to increase at a rate determined by the net base emitter negative-going voltage increase. This feedback effectively prevents the inverter from switching immediately from cutoff to saturation. Instead, a controlled rise time of the input step function to the output circuit is obtained such that the transition time for switching from the cutoff state to the saturation state is greater than 3.0 microseconds but less than 6.0 microseconds. Now, when the input to the inverter switches from -3 v-d-c to 0 v-d-c, the conduction of transistor Q1 decreases toward cutoff. With the change in base bias, the output stage switches with Q2 going into conduction and Q3 going towards cutoff. The feedback from output to input controls the fall time by preventing the inverter from switching immediately from saturation to cutoff. The transition of the fall time approximates that of the rise time. The output from the complementary symmetry circuit varies between 0 v-d-c on the data line when Q3 conducts and -13.5 v-d-c on the data line when Q2 conducts.

The remaining portion of the circuit containing Q4 and Q5 provides current limiting. Transistor Q4 is normally conducting. When the current through Q4 increases, Q5 conducts because its base potential rises above that of the emitter due to the voltage drop across R6. The current through Q5 produces a voltage drop across R5 which tends to decrease conduction through Q4 by increasing the base emitter bias of Q4. Therefore, if Q4 attempts to conduct heavily, Q5 samples the voltage change across R6 and produces a controlling voltage drop across R5 which limits the current through Q4. The initial surge current required to charge the data lines is obtained from C3. When there is no data on the lines, the signal to the input amplifier is a binary ''0'' (-13.5 v-d-c), on the line.

GATED INPUT AMPLIFIER

The gated input amplifier circuit (fig. 13-4) is used in the computer to receive the data input from external equipments. The circuit configuration is a grounded emitter amplifier Q2 using a PNP transistor. Assuming that the external equipment and I/O translator input lines are not providing signal inputs, (lines disconnected) the base of transistor Q2, is negative with respect to the emitter (due to the conduction from the -15 volt supply through R10, R11, R12 and the low forward bias resistance of Q2) and the transistor is conducting at saturation. The data output is approximately 0 v-d-c since the collector is effectively grounded through the conduction of Q2 and CR2 is back biased -3 volts. The transistor conducts as long as the I/O



124.173

Figure 13-4.—Gated input amplifier.

channel designator gate circuit is held at -3 volts because CR5 conducts holding the Q2 base negative with respect to the emitter. When the gate signal from the I/O translator, is -3 v-d-c, CR5 conducts and the base potential of Q2 remains negative with respect to the emitter. Thus the transistor remains at saturation. In this manner, regardless of what signal appears on the data input line, no information is entered into the Z-register. When a channel becomes active, the gate signal rises from -3 v-d-c to 0 v-d-c (see waveforms). Diode CR5 is disabled due to the negative voltage appearing at the base of the transistor from the voltage divider between +15 v-d-c and -15 v-d-c. The transistor remains in saturation. Now, the data input data signal exercises control. When the data input goes from -13.5 v-d-c to 0 v-d-c, the junction of the 4.7K resistor (R10) and the 9.1K resistor (R11) effectively becomes ground. The voltage divider action is such that the base of Q2 becomes positive with respect to the emitter. The transistor is turned off and the data output drops toward -15 v-d-c from 0 v-d-c. However, CR2 clamps the output at -3 v-d-c. When the data input goes from 0 v-d-c to -13.5 v-d-c, the Q2 base again becomes negative with respect to the emitter and the transistor goes into saturation. (The output rises from -3 v-d-c to 0 v-d-c.) Therefore, when a channel is active, a data input of a "0" or a "1", is repeated by the output as a "0" or a "1", respectively (as illustrated by the waveforms in figure 13-4). This is the time when information is entered into the Z-register by one of the 14 channels utilizing 30 input amplifiers. The input circuit, consisting of the two 9.1K resistors and the 330-micromicrofarad capacitor (C3), forms a filter network which prevents ringing of the input step function.

INPUT AMPLIFIER FOR CONTROL SIGNALS

The input amplifiers in the computer, used to receive control signals from external equipment, are not gated. The circuit is as shown in figure 13-4, with the gate signal line open. Under this condition, the transistor is normally in saturation. When a "1" or a "0" appears on the control line, the transistor switches as it did for the same inputs on the data input line. Essentially, the circuit operation is the same as for a gated data amplifier when a channel is active.

CONTROL LINE DRIVER

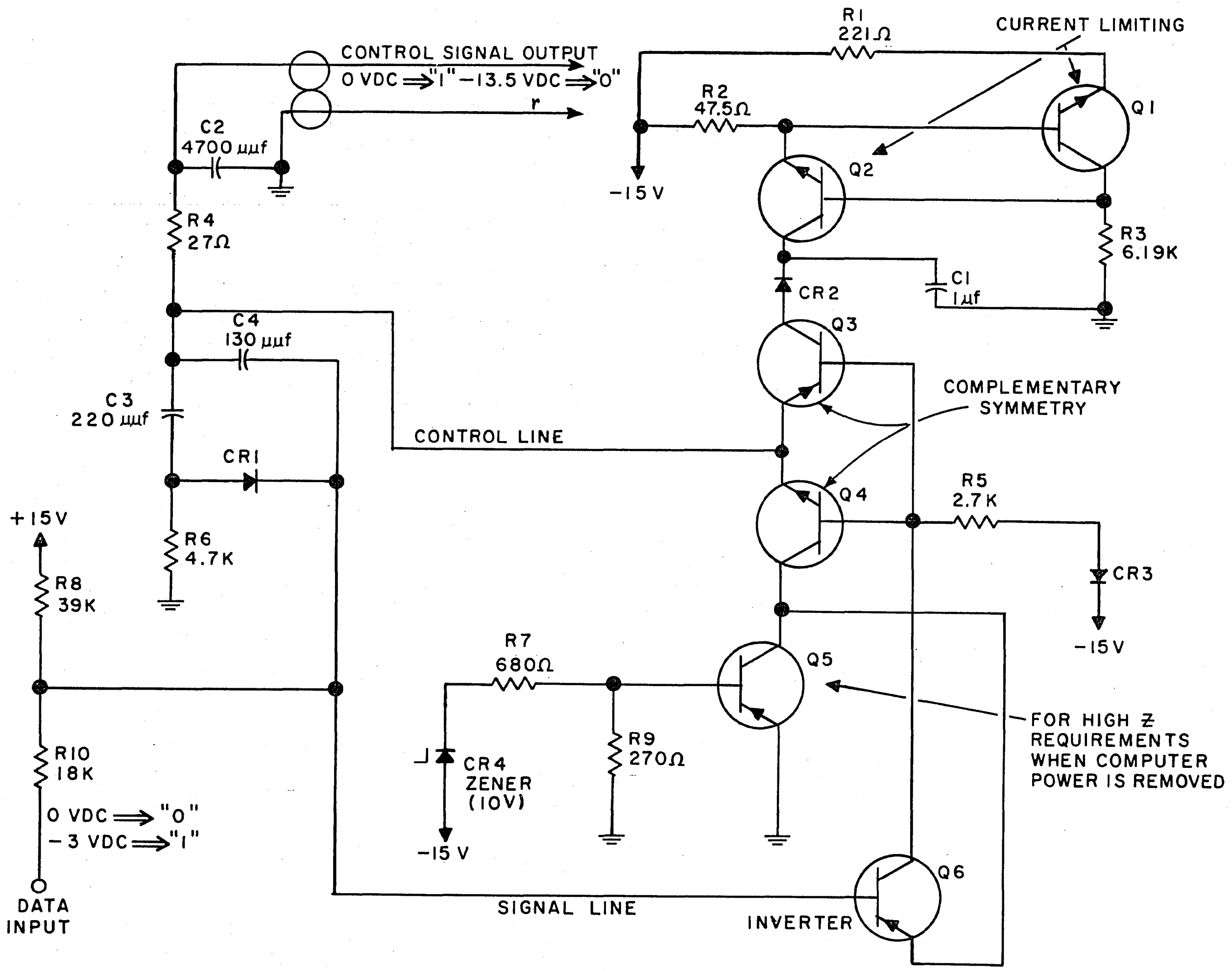
The control line driver (fig. 13-5), transmits control signals to an input amplifier in the external equipment. The circuit consists of six transistors; in a complementary symmetry circuit, Q3 and Q4, two transistors one inverter, Q6, two transistors for current limiting, Q1 and Q2, and one transistor Q5, used to meet high-impedance requirements when the computer has its power removed. The circuit is used in much the same manner as the data line driver with the difference being that the control line driver presents a high impedance to the line when power is off to avoid falsely putting a "1" on the line.

The functional operation of the control line driver is the same as described for the data line driver (fig. 13-3) with the addition of the impedance circuit, Q5.

When computer power is turned off, Q5 along with the isolation diodes CR2, CR3, and CR4 present a high impedance to the control line by back-erasing the diodes with -13.5 v-d-c from the external equipment when the signal is "0". The input amplifier in the external equipment sees this high impedance or effective binary "0" (-13.5V) on the line. Diode CR4, a 10V Zener diode, is incorporated in the base circuit of Q5 to set the turn-on and turn-off voltage levels of Q5. When the computer is first turned on, the -15V supply must rise to -10V before any voltage is applied to the base of Q5. When the computer is turned off, the voltage is removed from the base of Q5 as soon as the supply voltage decays below -10V. In this way, transients in the -15V supply do not cause conduction of Q5. This maintains the high impedance on the control line and prevents a false signal (a "1") from being applied to the line.

INPUT/OUTPUT CONTROL

The Input/Output (I/O) control function utilizes an 8.0-microsecond timing chain and an associated set of six flip-flops. The timing chain (figures 13-6 and 13-7) is called the e-designator timing chain. The six flip-flops (fig. 13-8) are called the t-designator flip-flops. The flip-flops are designated (1) Scan, (2) I/O_1, (3) Disable I/O_1, (4) I/O_2, (5) Disable I/O_2, and (6) Clock Overflow. (The scan, I/O_1 and I/O_2 sequences are defined in chapter 8.) Signals generated at different times in the timing chain, coupled with signals generated by any one of the



124.174

Figure 13-5. —Control line-driver.

six flip-flops, provide enable signals that permit the I/O section to function. The enable signals generated are the command signals used by (I/O) for processing requests.

e DESIGNATOR

The timing chain (figures 13-6 and 13-7) is a closed-loop timing device. This means that it recycles itself every 8.0 microseconds under normal circumstances.

Initially, to start the timing chain the MASTER CLEAR switch (fig. 10-9) is momentarily placed in the down position. In this position the output from $^{11}I^{84}$ (an inverter associated with the external interrupt circuit, not shown) is a "1" that clears V^{91} (fig. 13-6). Now the MASTER CLEAR switch is momentarily placed in the UP position. The HIGH SPEED switch is momentarily depressed. This action sets an inverter ($^{01}G^{73}$) in the console control circuit (not shown) and puts the computer in RUN condition.

At phase 2, $^{13}V^{60}$ is fully enabled by having all "0" inputs. The "1" output from this circuit sets T^{80} to start the timing chain and sets Scan.

Flip-flop V^{91} is set on the following phase 4 via $^{13}T^{81}$ to disable $^{13}V^{60}$, thus removing this "set" signal from T^{80}. As the timing chain runs, each flip-flop is set and cleared in a definite sequence as shown in figure 13-9.

Each phase (ϕ) as described in chapter 10 (figure 10-4) is a series of pulses each having a duration of 400 millimicroseconds (m μsec). Adjacent phases ($\emptyset_1$, $\emptyset_2$, etc.) are separated by 400 m μsec. Each flip-flop is first set and cleared later on the same clock phase. After 8.0 microseconds. T^{80} is again set if certain conditions are present that can enable the $^{12}T^{80}$ or the $^{03}T^{80}$ gate circuits.

In order to enable $^{12}T^{80}$, the MASTER CLEAR switch must be in the normal position so that the input from $^{11}I^{50}$ (not shown) is a "0". Then with Scan set and at 0_1 time, T^{90} is set, and $^{12}T^{80}$ is fully enabled at phase 2.

In order to enable $^{13}T^{80}$, a different set of conditions must be present. First, the output from $^{05}V^{60}$ must be a "0" to indicate that either V^{61} (I/O_1) or V^{62} (I/O_2) is set. Second, the MASTER CLEAR switch must be in the normal position so that the input from $^{11}I^{92}$ (not shown) is a "0". Third, memory must be available for use during I/O_1 or I/O_2. Therefore the input from $^{01}L^{17}$ (not shown) must be a "0". With these conditions satisfied at phase 2 time, $^{13}T^{80}$ can be fully enabled and T^{80} is set.

When the output from $^{05}V^{60}$ is a "0" and the output from $^{13}T^{80}$ is a "1", I/O is going to make a memory reference. Note that memory must be available in order to produce a "1" output from $^{13}T^{80}$. The "1" output from $^{13}T^{80}$ along with the "1" output from T^{80} forces the output from $^{03}T^{80}$ to a "0" that is an input to $^{19}V^{61}$. The output from $^{13}T^{80}$ is held at a "0" after phase 2 by the "1" input from $^{00}T^{80}$. Now, $^{19}V^{61}$ with "0" inputs from $^{13}T^{80}$ and $^{05}V^{60}$ has a "1" output that is transmitted to $^{14}L^{11}$ (fig. 12-17). The "0" output from $^{14}L^{11}$ is one of the enables required to start the Memory timing chain and to set the memory lockout. I/O is now using memory and main computer control cannot make use of it at this time.

When the computer is not in high-speed operation, manual control can be exercised over the timing chain. Each flip-flop in the timing chain can be set by depressing the associated indicator-pushbutton on the computer console. When a flip-flop is set, its associated indicator is lighted.

t-DESIGNATOR

The flip-flops associated with the t designator (fig. 13-8) indicate the status of any request being processed by the I/O section. It is during the scan sequence (which interrogates all input/output channels) that requests are entered into I/O. The initial memory reference, if necessary in the execution of the I/O request, is made during the I/O_1 sequence (see discussion in chapter 8). The second memory reference, if necessary, is made during I/O_2. In other words, during I/O_1 the contents of a special memory address are read up into the Z register. The upper 15 bits and the lower 15 bits define an area in memory where data is to be stored or read out. With the Z register remaining unchanged, its contents are written back into memory with the lower half incremented by one. In the case of Real-Time Clock, this incrementing updates the clock count at address 00036. Also, during I/O_1 the contents of Z_u are compared with the contents of Z_L. This action compares the upper and lower halves of the contents of a special address which, for the Buffer mode, is a comparison of the initial and the terminal addresses. If they are equal, the Buffer mode is terminated.

During I/O_2 the contents of Z_L are transmitted to the S-register (mode address$\rightarrow$S). From the address in memory (as determined by information transmitted to the S-register) data is read out for transmission to external equipment or data is written in from external equipment. In the case of Real-Time Clock, it is during I/O_2 that the upper half of address 0036 is incremented by one.

More than one flip-flop of the t-designator (fig. 13-8) can be set at one time. Data transfer can be taking place while the next request is being entered into I/O. For example, note in figure 13-9B that the Scan sequence overlaps the time that the Disable I/O_2 sequence is enabled.

The clearing of the t-designator is accomplished (1) by operating the MASTER CLEAR switch and (2) by enables (application of "0's") from the e-designator. When the MASTER CLEAR switch is depressed, the output from $^{10}O^{42}$ (not shown) to $^{11}V^{16}$ (fig. 13-8) is a "0" which is inverted to a "1" by $^{11}V^{16}$. This "1" signal clears V^{60} (Scan), V^{61} (I/O_1), V^{62} (I/O_2), and V^{65} (Clock Overflow). As shown, Disable I/O_1 V^{64} and Disable I/O_2 V^{63} are not cleared directly by master clear. I/O_1 (V^{61}) can only be set when $^{13}V^{61}$ is fully enabled at phase 4. The conditional enable inputs to $^{13}V^{61}$ are (1) $^{01}V^{16}$ — memory access flip-flop set indicating that I/O is going to use memory, (2) $^{01}L^{17}$ — L^{17} set indicating that memory is available, (3) $^{00}G^{15}$ — I/O Lockout flip-flop cleared indicating that

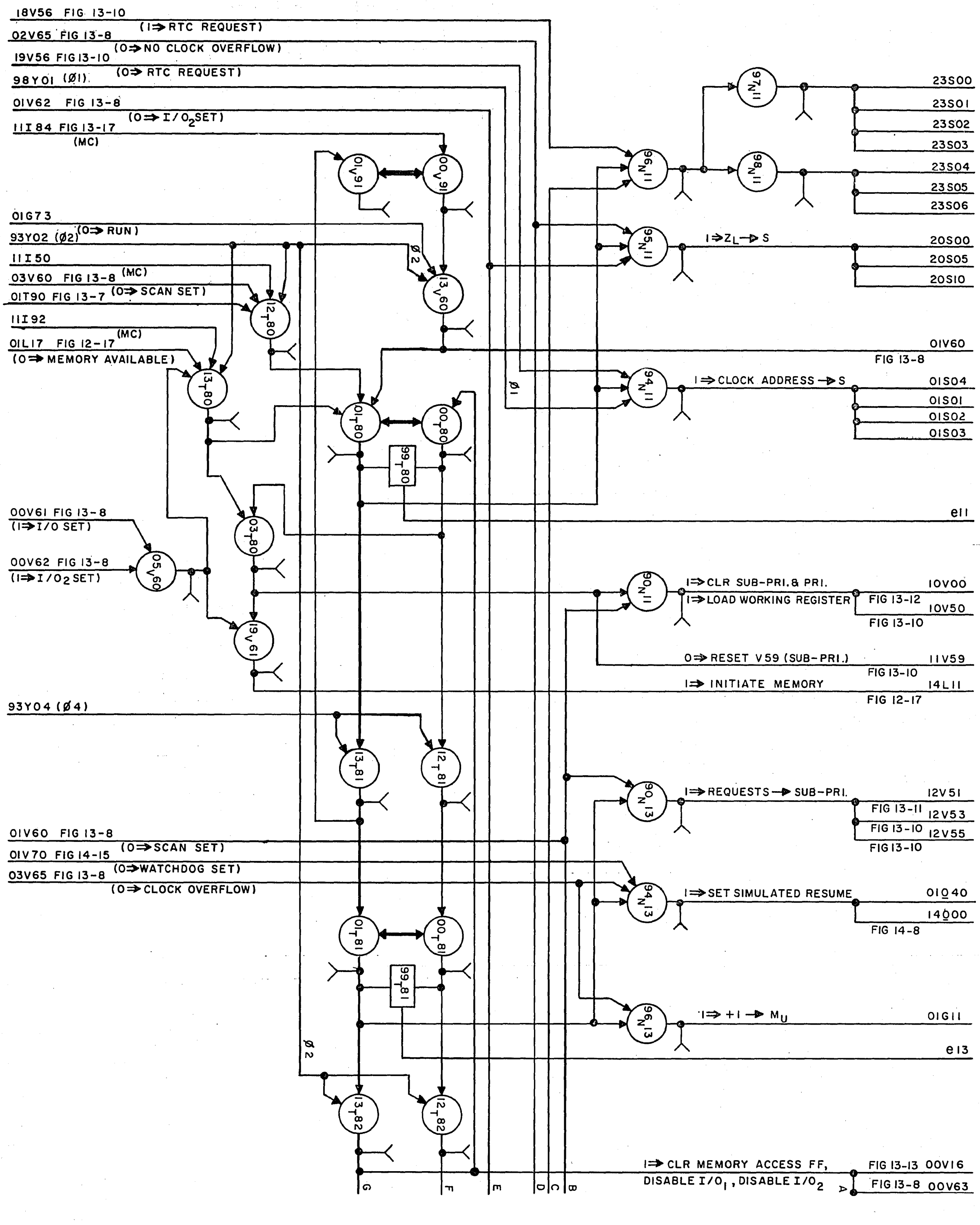


Figure 13-6.—e designator, (Part I).

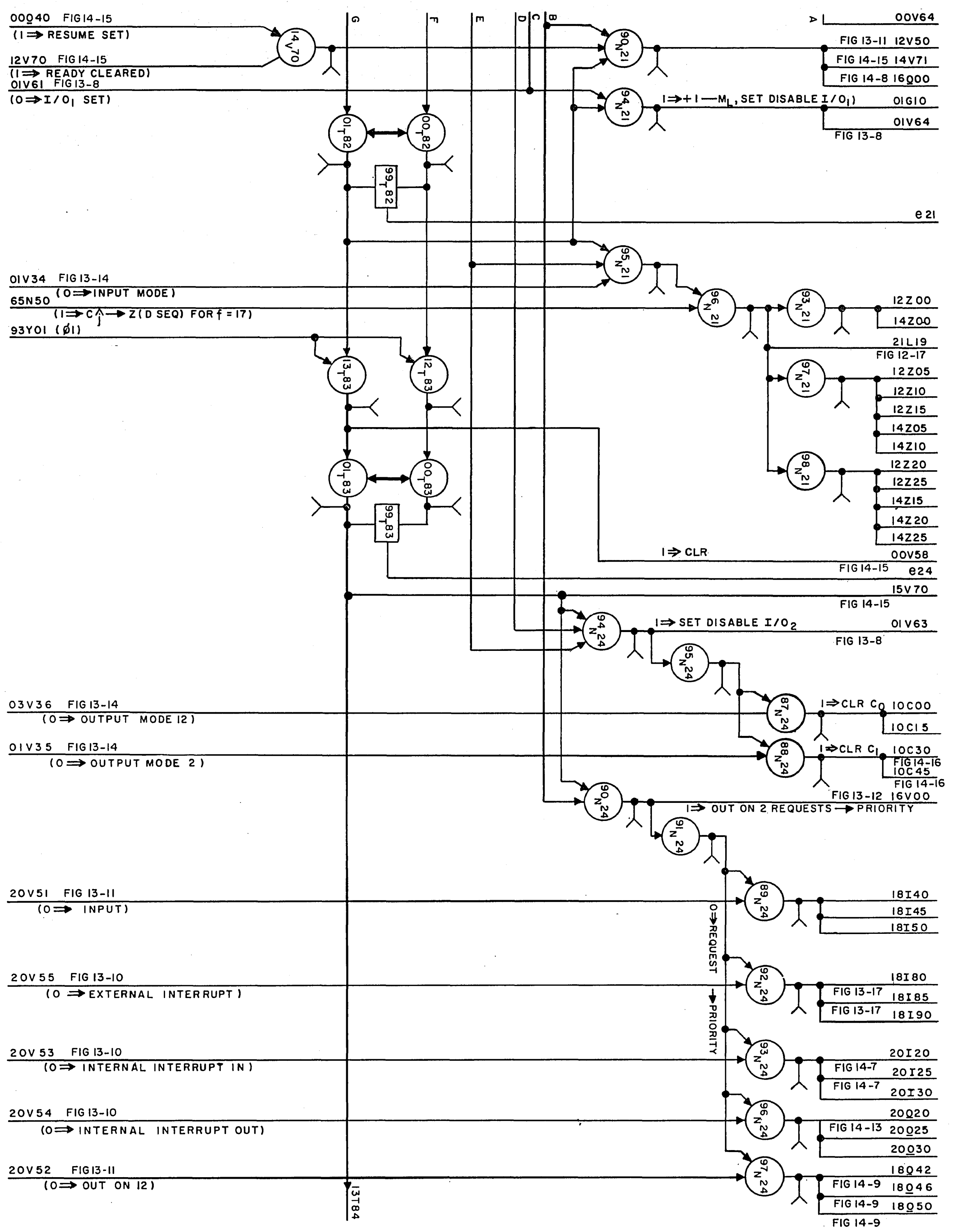


Figure 13-6.—e designator, (Part I)—Continued.

124.175

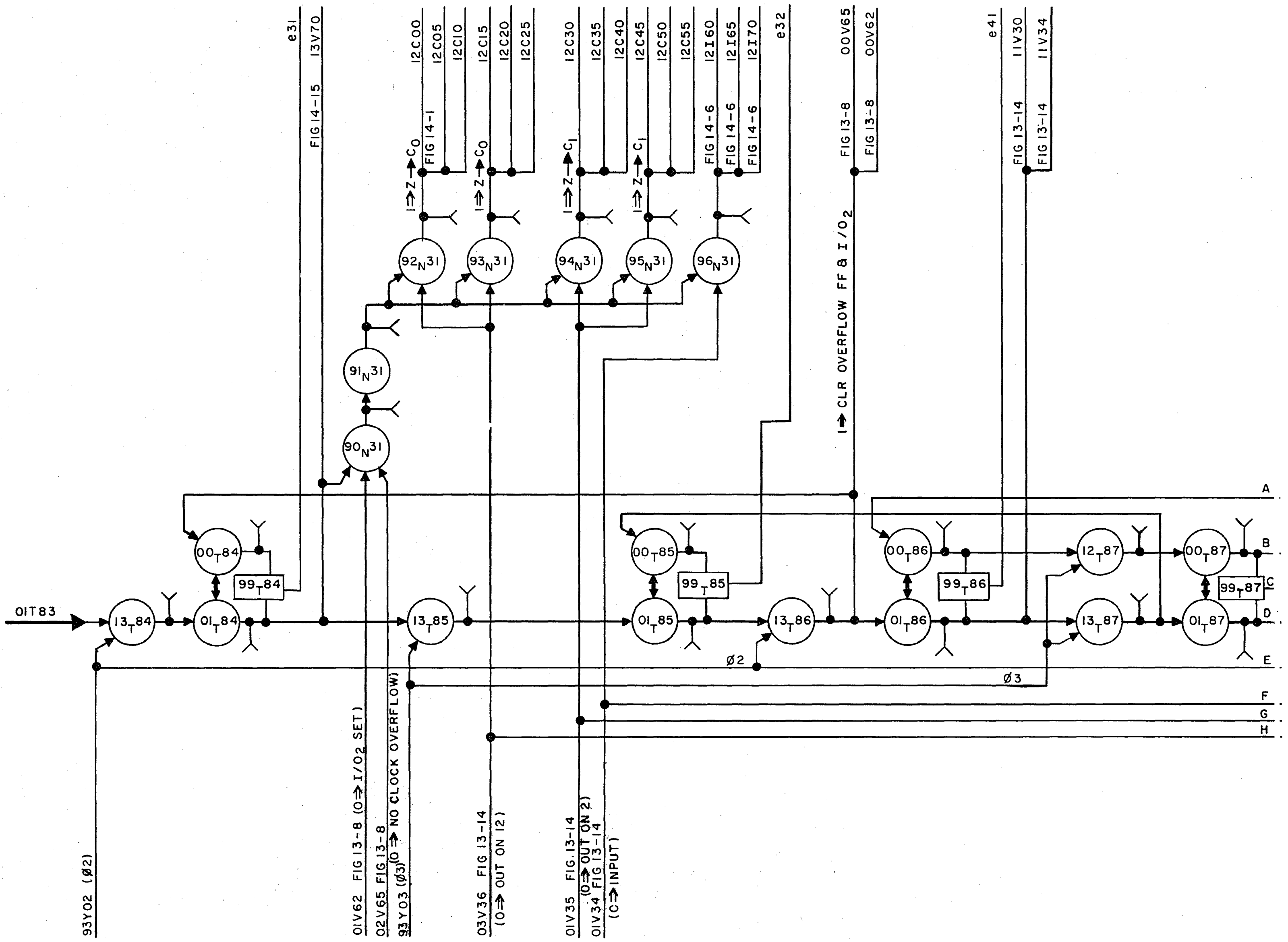
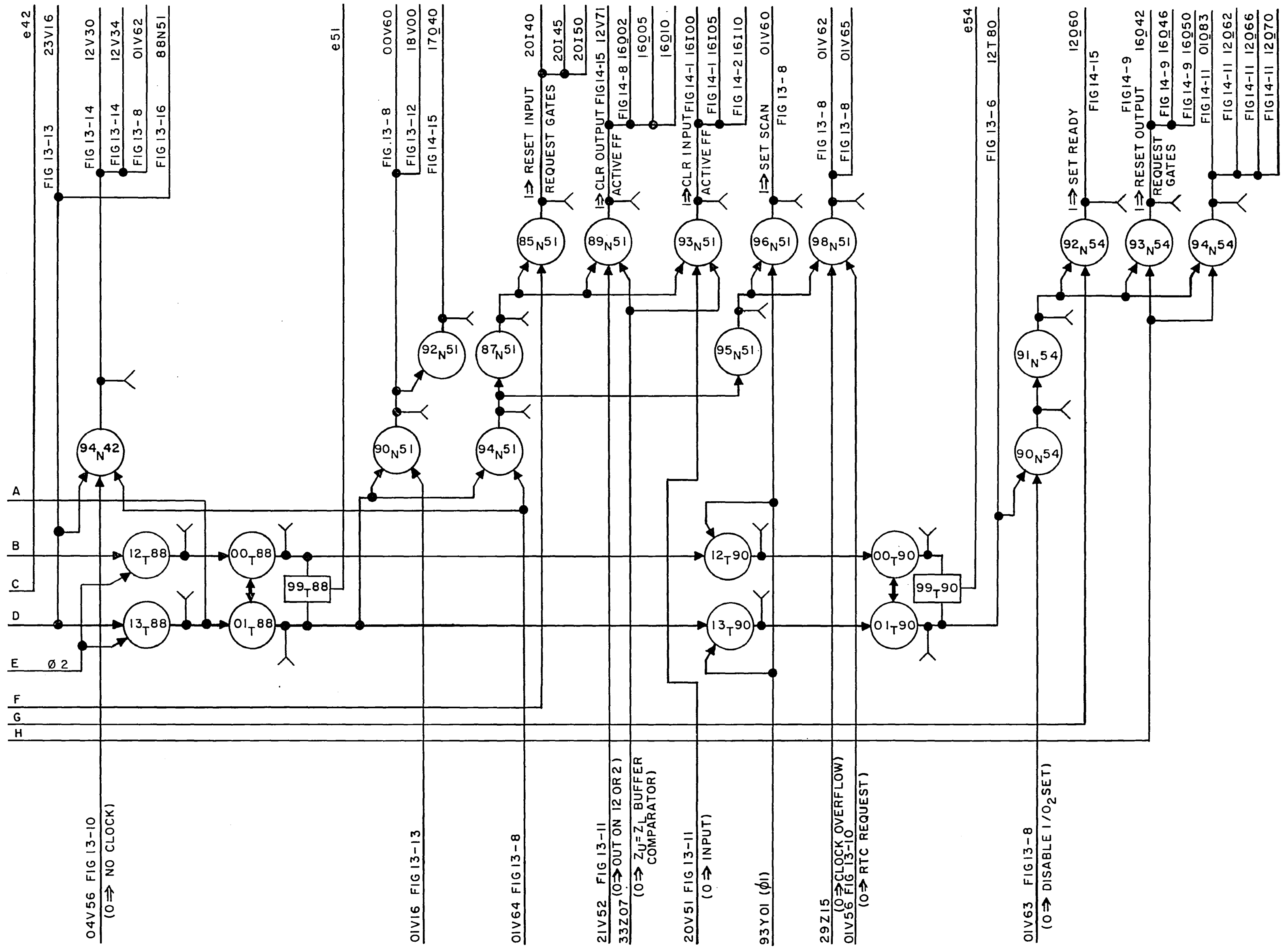
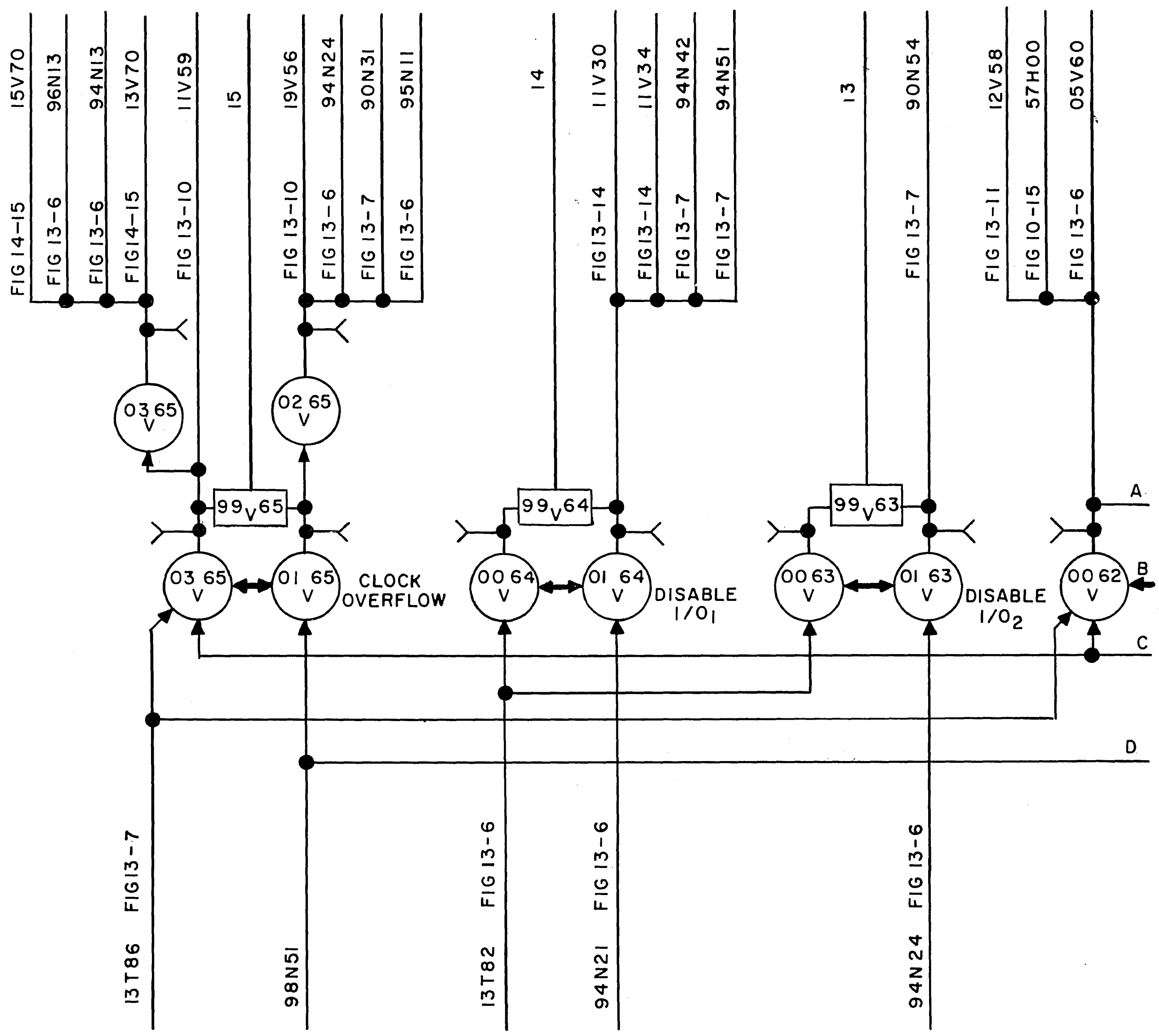


Figure 13-7.—e designator, I/O timing chain, (Part II).



124.176

Figure 13-7.—e designator, I/O timing chain, (Part II)—Continued.

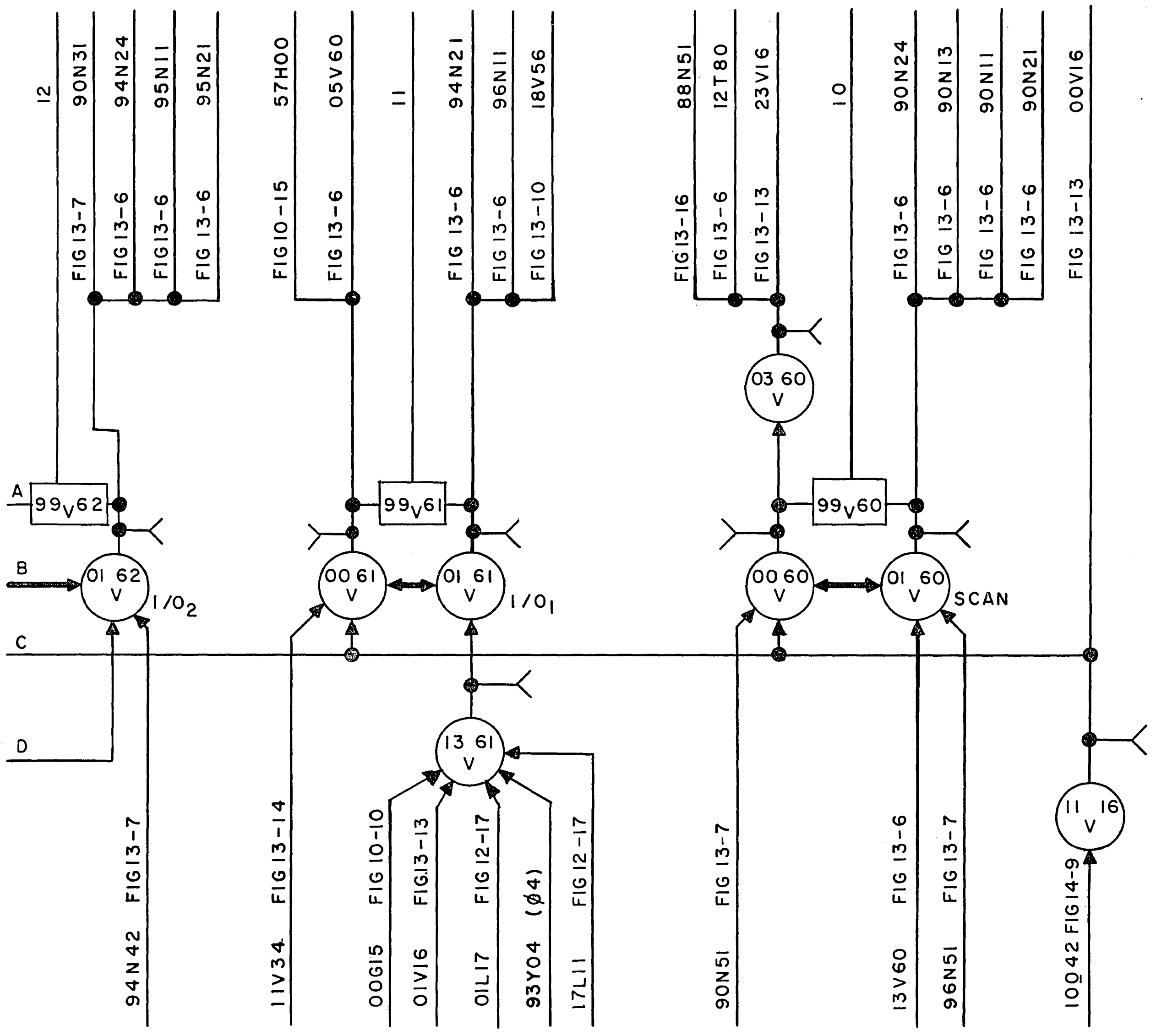


an I/O instruction (f = 13, 73, 74, etc.) is not being processed by main computer control, and (4) ${}^{17}L^{11}$ — indicating that main computer control is not using memory. When the above conditions are satisfied, ${}^{13}V^{61}$ is enabled to set V^{61} and permit an I/O reference as necessary for executing an I/O request.

Manual control can be exercised over the t-designator when the computer is not in high-speed operation. Each flip-flop can be set by depressing the appropriate indicator-pushbutton.

CONTROL TIMING

In the following topics, reference is made to times during the I/O timing chain, e.g., T80, T81. Since the times used are in reference to a particular request, they may occur during the first cycle of the timing chain or the second or the third cycle. With regard to a particular request (input), the first cycle of the timing chain is designated T80→T90, the second cycle $T80_1$→$T90_1$, the third cycle $T80_2$→$T90_2$. When



124.177

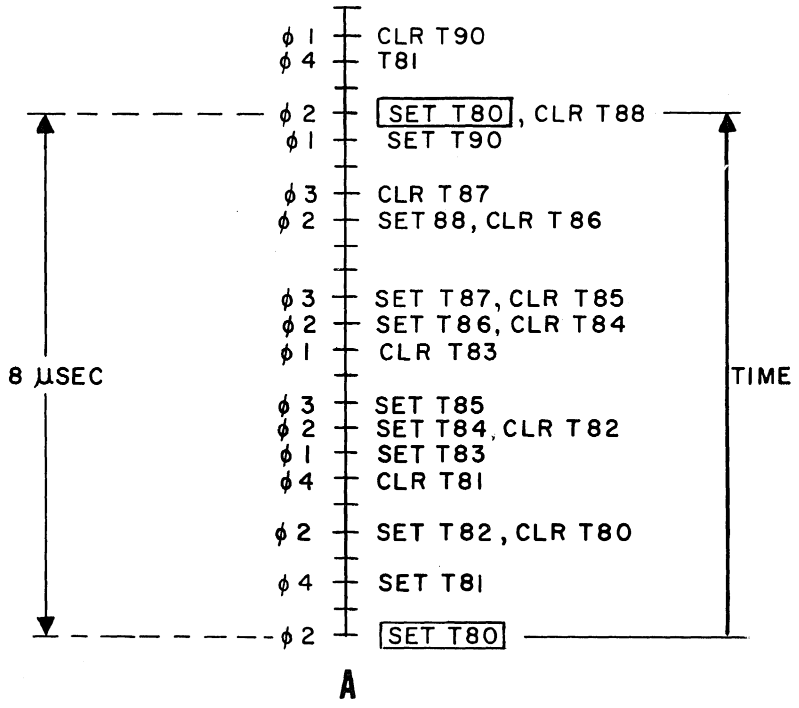
Figure 13-8.—t designator timing chain.

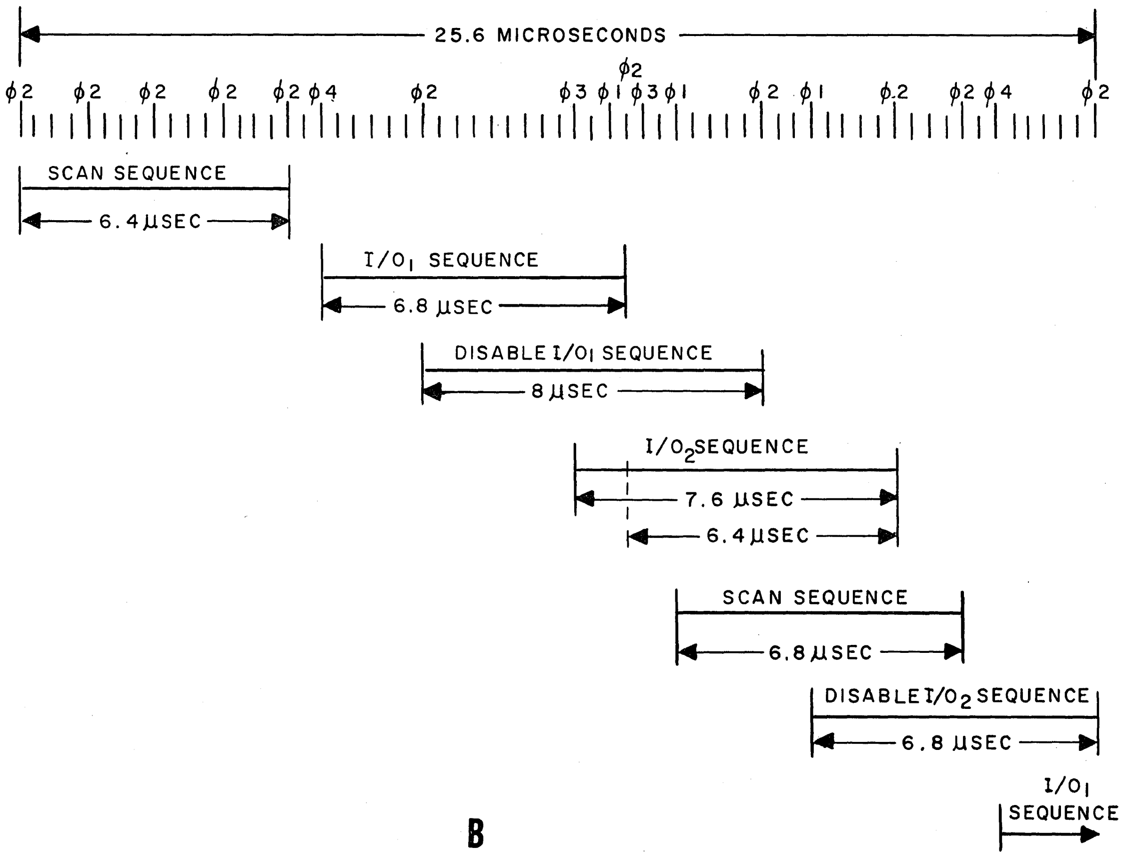
time is an indeterminate factor, as during an Out-on-2 request, it is designated $T80_x \rightarrow T90_x$.

PRIORITY DETERMINATION

The I/O section has the capability of using different modes of operation. It determines the priority of these modes through its own control functions. Since the I/O section is essentially independent of main computer control, the processing of the I/O modes can be accomplished regardless of the current program in the computer. The modes of operation in descending priority are (1) Read-Time Clock, (2) External Interrupt, (3) Internal Interrupt Out, (4) Internal Interrupt In, (5) Output-on-12, (6) Input, and (7) Output-on-2. These modes are defined as follows:

1. Real-time clock: The means by which the actual elapsed time is measured in seconds or fraction thereof.





124.178

Figure 13-9.—e designator and t designator timing.

2. External interrupt: A signal from an external equipment (Not a computer) which requires computer attention.
3. Internal interrupt out: A program interrupter which indicates that an output buffer has been completed.
4. Internal interrupt in: A program interrupter which indicates that an input buffer has been completed.
5. Output-on-12: Outputting data on one of the 12 normal channels. (To external equipments, not to external computers.)
6. Input: Inputting data from any external source which is connected to the computer.
7. Output-on-2: Outputting data on either channel 0 or channel 1 which are the two channels reserved for intercomputer communications, so that the I/O section must determine channel priority. Channel 13 has the highest priority descending to channel 0 with the lowest priority.

SUBPRIORITY NETWORK (m-DESIGNATOR)

The subpriority network m-designator (figures 13-10 and 13-11) is used to process requests to determine which mode of operation is to be processed by I/O at a particular time. Subpriority is composed of seven circuits, each one with its associated flip-flop and gates. When a request is to be processed by subpriority, the designated flip-flop for the requested mode of operation is set by enables from the request gates and I/O control during the scan sequence. Real-Time Clock operation is an exception in that the clock oscillator (not shown) provides the enable signal in conjunction with I/O control. The clock is not associated with any particular channel. In fact, the clock inputs a request to subpriority once every 977 microseconds (the period of the oscillator) so that the lower 15 bits of the clock address (00036) can be incremented.

When a mode of operation is selected by subpriority, it disables all other modes with lower priority at that instant. If a Real-Time Clock request and an External Interrupt request entered subpriority at the same time, an output from the Real-Time Clock circuits would disable the external interrupt until after the real-time clock requests have been acted upon. During the next Scan time, the External Interrupt can be processed. The outputs from subpriority provide the required enables to priority to designate the special address for the particular mode of operation.

Every circuit in subpriority is cleared by the same command enables. There is an associated MANUAL CLEAR indicator-pushbutton, m_1 (see figure 10-8) which, when depressed, clamps the output from $^{10}V^{50}$ (fig. 13-10) at "0" (ground) via $^{99}V^{98}$. This "0" output is fed to $^{11}V^{54}$ and $^{11}V^{50}$. At phase 1 these two circuits are fully enabled and provide "1" outputs that clear subpriority.

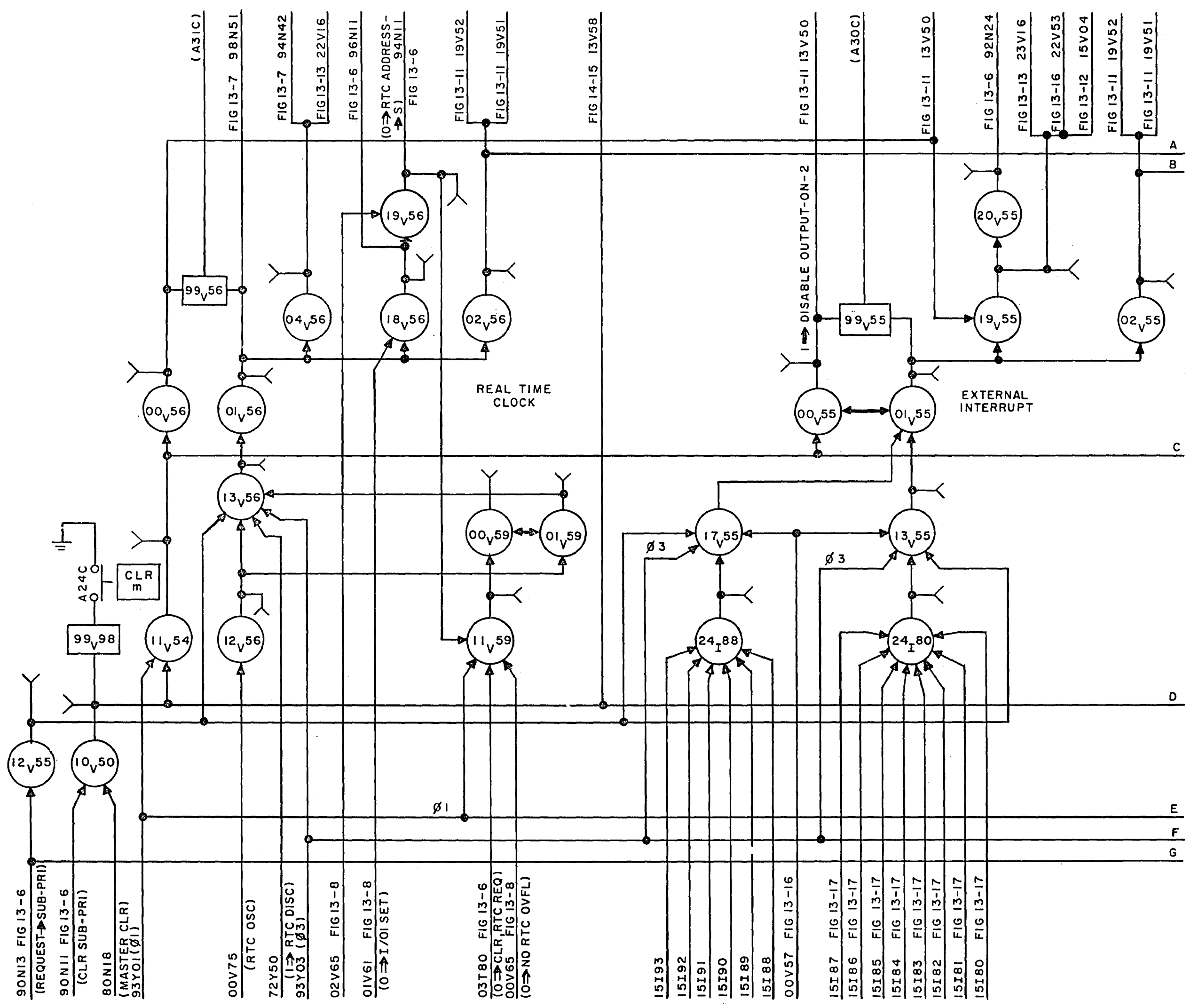
When the MASTER CLEAR switch is momentarily depressed, the input to $^{10}V^{50}$ from $^{80}N^{18}$ (not shown) is a "1". The output from $^{10}V^{50}$ is a "0" that is fed to $^{11}V^{54}$ and $^{11}V^{50}$. The clearing action is now the same as for manual clear.

During Scan at the start of the I/O timing sequence, time T^{80}, (see fig. 13-9), a clear command is also generated. At this time, the input to $^{10}V^{50}$ from $^{90}N^{11}$ is a "1". The clearing action now is as for master clear. Following the clearing of subpriority during Scan, a command is generated at time T^{81} by the I/O timing sequence to enable subpriority to process a waiting request. This command, "Requests to Subpriority", is generated by $^{90}N^{13}$ (fig. 13-6). The input from this circuit is a "1" that is transmitted to $^{12}V^{55}$, $^{12}V^{53}$, (fig. 13-10), and $^{12}V^{51}$ (fig. 13-11). The outputs from these circuits are "0's" that are utilized in conjunction with phase 3 to fully enable the subpriority request gates. Which mode of operation is enabled depends on the priority of the waiting requests. When a mode of operation is enabled, its associated indicator is illuminated on the console. (The Out-on-2 mode, which is explained later, is a special case that is not enabled by $^{90}N^{13}$.)

MEMORY ADDRESSES

The computer addressable storage locations include those in the magnetic core storage, supplemented by those in the A-, Q-, and B-registers. The addresses in magnetic core storage and their uses are listed below.

00000		:	Fault entrance
00000	—	00017:	Memory (Bootstrap)
00020	—	00035:	External interrupt
00036		:	Real-time clock
00037		:	Memory
00040	—	00055:	Internal interrupt (In)
00056	—	00057:	Memory
00060	—	00075:	Internal interrupt (out)
00076	—	00077:	Memory

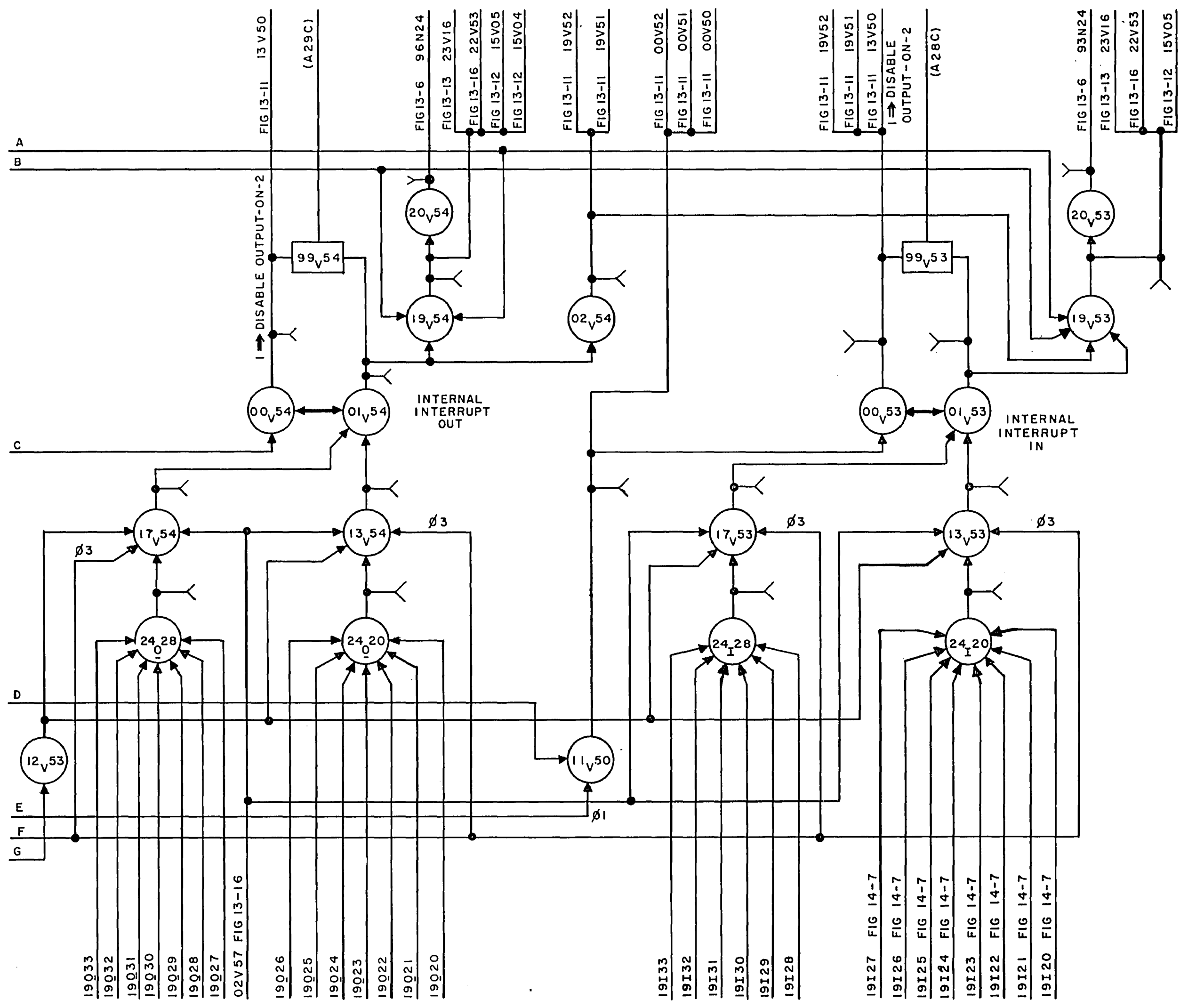

REAL TIME CLOCK
EXTERNAL INTERRUPT
1 → DISABLE OUTPUT-ON-2
CLR m
A24C
90N13 FIG 13-6 (REQUEST→SUB-PRI)
90N11 FIG 13-6 (CLR SUB-PRI)
80N18 (MASTER CLR) 93Y01(Ø1)
00V75 (RTC OSC)
72Y50 (1→ RTC DISC) 93Y03 (Ø3)
02V65 FIG 13-8
01V61 FIG 13-8 (0→I/01 SET)
03T80 FIG 13-6 (0→CLR RTC REQ) 00V65 FIG 13-8 (0→NO RTC OVFL)
00V57 FIG 13-16
FIG 13-7 98N51
(A31C)
(A30C)
FIG 14-15 13V58
FIG 13-11 13V50
FIG 13-6 92N24

00100 — 00115:	Input Buffer Control register
00116 — 00117:	Memory
00120 — 00135:	Output Buffer Control register
00136 — 07777:	Registers for 4, 002 words
10000 — 17777:	Registers for 4, 096 words
20000 — 27777:	Registers for 4, 096 words
30000 — 37777:	Registers for 4, 096 words
40000 — 47777:	Registers for 4, 096 words
50000 — 57777:	Registers for 4, 096 words
60000 — 67777:	Registers for 4, 096 words
70000 — 77777:	Registers for 4, 096 words

REAL-TIME CLOCK

When a Real-Time Clock request is to be processed by subpriority, the Real-Time Clock Control flip-flop, V^{59}, (fig. 13-10) and the subpriority circuits are in the cleared state. Also, the DISCONNECT RTC (fig. 10-9) must be inactive. In other words the clock is being utilized and the output from $^{72}Y^{50}$ on the console control circuit (not shown) is a "0".

As the input to $^{12}V^{56}$ from $^{00}V^{75}$ in the real-time clock oscillator (not shown) becomes a "0", the output from $^{12}V^{56}$ becomes a "1"



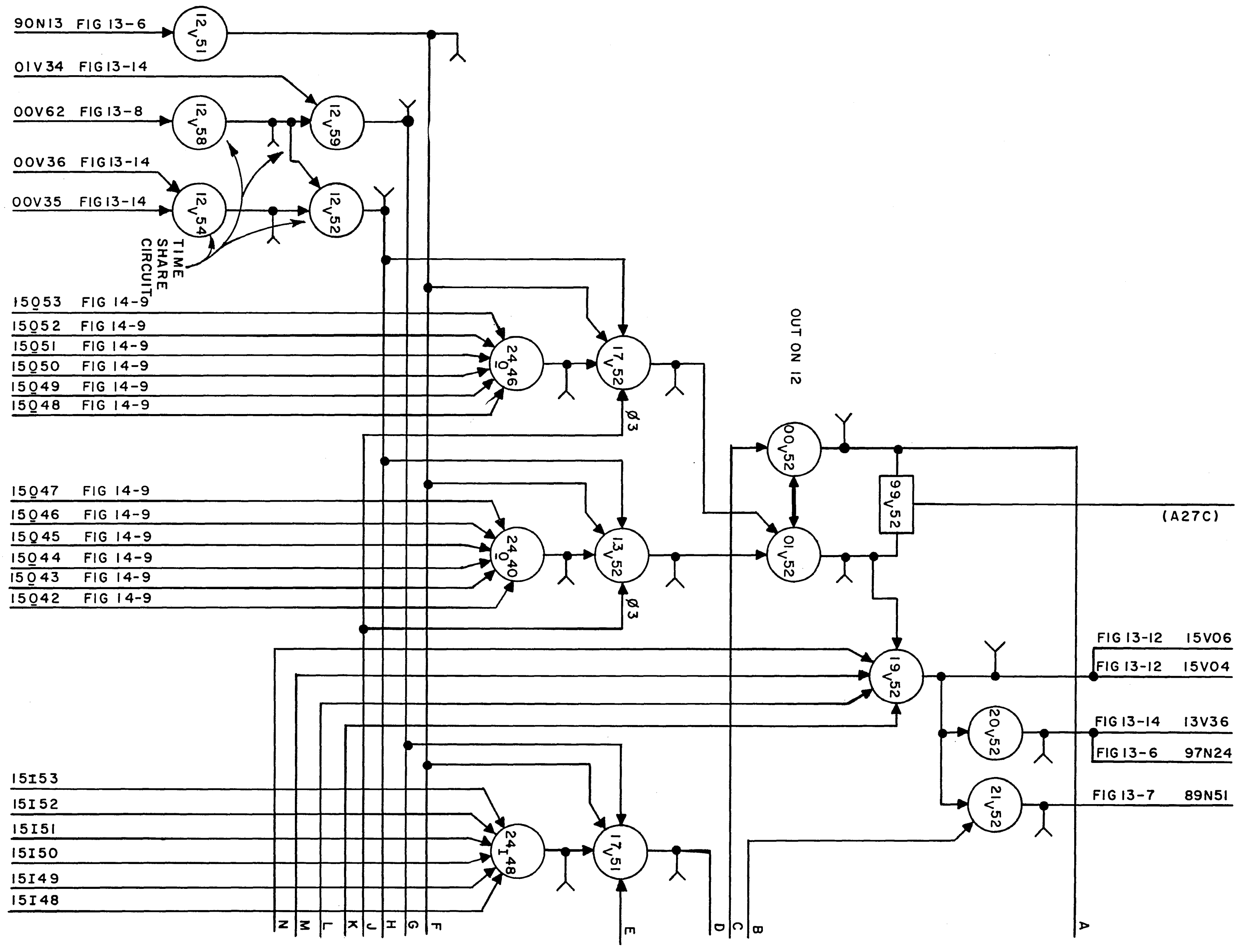
124.179

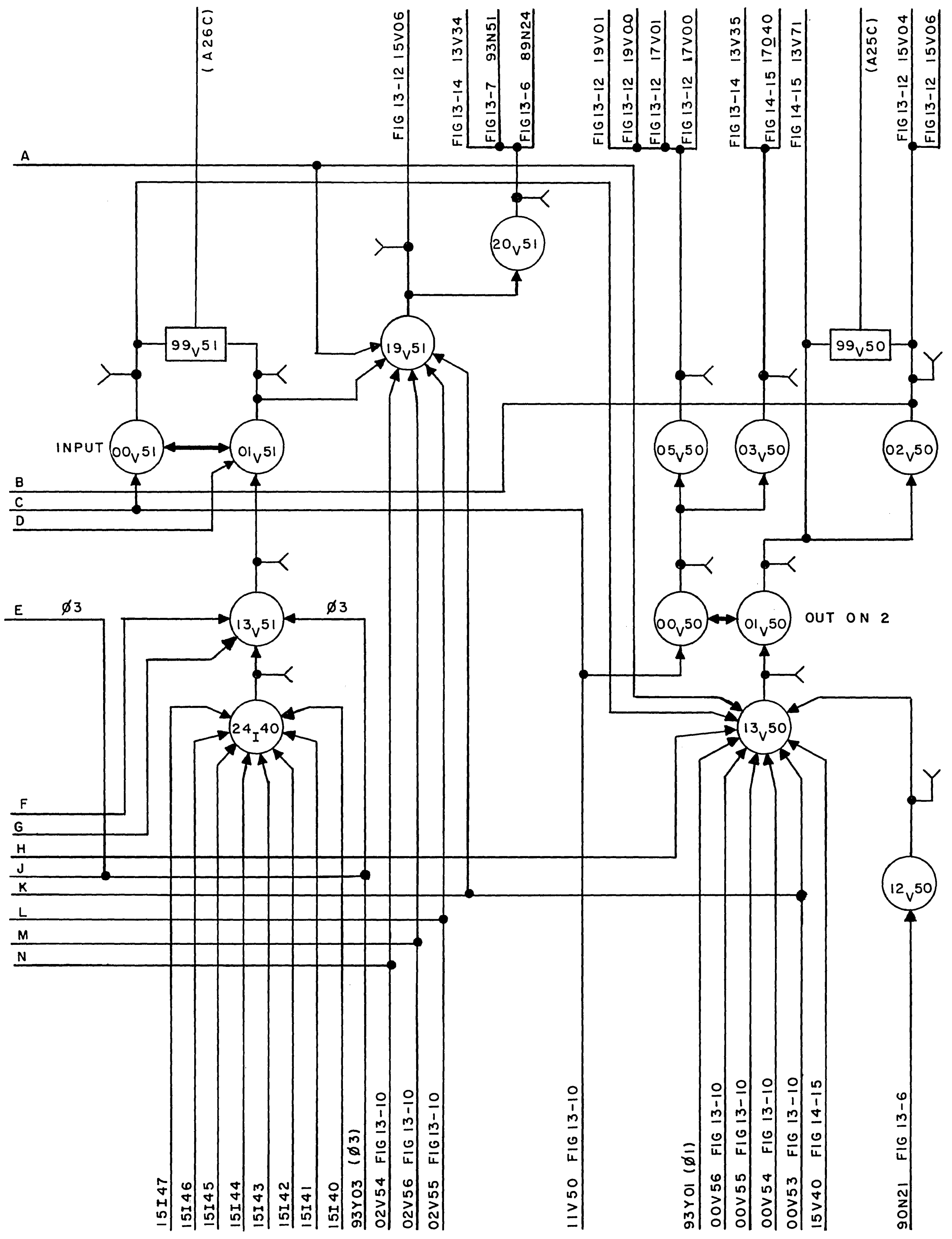
Figure 13-10.—Subpriority, Part 1.

that sets V^{59}. When the input to $12V^{56}$ changes to a "1" as the Real-Time Clock oscillator changes state, the $13V^{56}$ gate has "0" inputs from $V59$, $72Y50$, and $12V56$. This gate is fully enabled at phase 3 time if there is a Req→Sub-Pri command from $90N13$ a "0" output from $12V55$ to $13V^{56}$. The "1" output from $13V^{56}$ sets V^{56} indicating a Real-Time Clock request is in subpriority.

The "1" output from the zero side of $00V^{56}$ disables external interrupt by disabling $19V^{55}$ and disables Out-on-2 by disabling $13V^{50}$ (fig. 13-11). The "1" output from $02V^{56}$ (fig. 13-10) disables the remainder of subpriority by disabling each of the remaining $19V^{5-}$ circuits.

With V^{56} (fig. 13-10) set, the "1" output from $04V^{56}$ provides one of the enable signals required to set the Memory Access flip-flop, V^{16} (not shown). When V^{16} is set it is then possible to set I/O_1 (fig. 13-8). With I/O_1 set, the inputs to $18V^{56}$ (fig. 13-10) are a "0" from $01V61$ and a "0" from $01V^{56}$. The "1" output from $18V^{56}$ prevents the transmission at time $T80_1$ of the mode address via enables from $96N^{11}$ to the S-register. Instead, this "1" output forces the output from $19V^{56}$ to a "0". This


90N13 FIG 13-6
01V34 FIG 13-14
00V62 FIG 13-8
00V36 FIG 13-14
00V35 FIG 13-14
TIME SHARE CIRCUIT
15O53 FIG 14-9
15O52 FIG 14-9
15O51 FIG 14-9
15O50 FIG 14-9
15O49 FIG 14-9
15O48 FIG 14-9
15O47 FIG 14-9
15O46 FIG 14-9
15O45 FIG 14-9
15O44 FIG 14-9
15O43 FIG 14-9
15O42 FIG 14-9
15I53
15I52
15I51
15I50
15I49
15I48
12V51
12V58
12V59
12V54
12V52
24O46
17V52
24O40
13V52
24I48
17V51
Ø3
Ø3
OUT ON 12
00V52
01V52
99V52
(A27C)
19V52
20V52
21V52
FIG 13-12 15V06
FIG 13-12 15V04
FIG 13-14 13V36
FIG 13-6 97N24
FIG 13-7 89N51
N
M
L
K
J
H
G
F
E
D
C
B
A



124.180

Figure 13-11.—Subpriority, Part 2.

enables the transmission of the clock address to the S-register at time T80$_1$. Now, the ^{11}V^{59} gate (which controls the clearing of V^{59}) is fully enabled by the "0" inputs from ^{19}V^{56}, ^{03}T^{80}, and ^{00}V^{65} at phase 1. The "1" output from ^{11}V^{59} clears V^{59} in preparation for another clock cycle.

It is during the time of I/O$_1$ that the contents of the special clock address (00036) are read out of memory into the Z register. At time T82$_1$ (the second timing chain cycle after the request has been received) Disable I/O$_1$ is set and the contents of the lower half of address 00036 are incremented by one as they are written back into memory.

The normal sequence, i.e., the procedure which usually occurs, clears I/O$_1$ during time T86$_1$ and sets I/O$_2$ during time T87$_1$. However, for a Real-Time Clock request, the "1" output from ^{04}V^{56} disables ^{94}N^{42} (fig. 13-7) the gate used to set I/O$_2$ (fig. 13-8). Therefore, a normal Real-Time Clock request is processed and the clock address content updated using only the time from Scan through Disable I/O$_1$.

Whenever the clock count in the lower half of 00036 becomes all "1's", such that the addition of another count causes these 15 bits to become all "0's", a special condition arises called "clock overflow." There is now an "overflow" bit (a "1") that has to be added to the upper half of address 00036. Under these conditions gate circuit ^{98}N^{51} (fig. 13-7) is enabled at time T88$_1$ by "0" inputs from ^{01}V^{56} and an overflow detection circuit (not shown). The "1" output from ^{98}N^{51} sets I/O$_2$ (V^{62}) and Overflow flip-flop V^{65} (fig. 13-8). The "1" output from ^{02}V^{65} holds the output from ^{19}V^{56} (fig. 13-10) at a "0". At time T80$_2$, the clock address is again transmitted to the S-register and its contents are again read up into the Z-register. The upper half of address 00036 is incremented by one as it is written back into memory. The processing of a complete Real-Time Clock request is now finished.

During the processing of the Real-Time Clock request, there may be requests waiting to be processed by subpriority. Since requests are entered into subpriority during Scan, a request must wait until Scan is reset during Disable I/O$_1$ of a previous request.

EXTERNAL INTERRUPT

At the start of the I/O timing sequence subpriority is cleared, and an enable is generated to send requests to subpriority. In subpriority (fig. 13-10) the inputs to ^{24}I^{88} and ^{24}I^{80} are from the ^{15}I-- circuits (not shown) of the external interrupt circuits, one input from each of the 14 channels. Since subpriority is solely concerned with a mode of operation and not the channel on which it occurs, a "0" output from ^{24}I^{88} and/or ^{24}I^{80} indicates the presence of an external interrupt on some channel. When an external interrupt is indicated, another condition must be satisfied before the request can be honored. The input from an Interrupt flip-flop (^{00}V^{57}, not shown) must be a "0" indicating that the computer is not in fault condition, not in wired memory mode, and not processing a previous interrupt. The gate circuit ^{17}V^{55} (fig. 13-10) can have "0" inputs from ^{24}I^{88}, ^{00}V^{57}, and ^{12}V^{55}. Gate circuit, ^{13}V^{55}, can have "0" inputs from ^{24}I^{80}, ^{00}V^{57}, and ^{12}V^{55}. The gate circuit or circuits are fully enabled at phase 3 to set V^{55}.

The "1" output from ^{00}V^{55} disables Out-on-2 by disabling ^{13}V^{50} (fig. 13-11). The "1" output from ^{02}V^{55} disables the remainder of subpriority below the external interrupt by disabling the ^{19}V^{5-} circuits (fig. 13-11). The inputs to ^{19}V^{55} (fig. 13-10) are a "0" from ^{01}V^{55} and a "0" from ^{00}V^{56} indicating no Real-Time Clock request (a higher priority condition). The "1" output from ^{19}V^{55} forces the output from ^{20}V^{55} to a "0". This output is one of the enables required to enable ^{92}N^{24} (fig. 10-6). The output from ^{92}N^{24} enables the transmission of the request to priority to determine the channel to be honored.

The "1" output from ^{19}V^{55} (fig. 13-10) is also applied to the priority circuits (figures 13-12 and 13-13) and is used to disable ^{23}V^{16} (fig. 13-13). This prevents the setting of the Memory Access flip-flop which is not required by I/O during an interrupt (no memory reference is to be made). Another output from ^{19}V^{55} (to ^{22}V^{53} not shown) is one of the enables required to set the Interrupt flip-flop (not shown) and to load the storage register with the special address from priority. The other output from ^{19}V^{55} to ^{15}V^{04} in priority (fig. 13-12) is used to set the initial address of the External Interrupt Entrance register to 00020. This is the special address for channel 0 and it is incremented by the channel number of the particular channel that is being referenced. For an external interrupt, the special addresses are for 00020 through 00035.

Since the Memory Access flip-flop (fig. 13-13) remained cleared, I/O$_1$ cannot be set (the

I/O_1 sequence is not required since no Memory reference is made.) This means that I/O control remained in Scan so that when the timing sequence returns to time T80, subpriority and priority are cleared in preparation for another request. The significance of this is that I/O sequence relinquishes control of the interrupt to main computer control.

INTERNAL INTERRUPT OUT

The enabling sequence for the gate circuits, $^{17}V^{54}$ and $^{13}V^{54}$ (fig. 13-10) is the same as for an external interrupt except that the inputs to $^{24}O^{28}$ and $^{24}O^{20}$ are from the ^{19}O-- circuits of the output monitors (not shown). An internal interrupt occurring on any of the output monitor channels causes the output from $^{24}O^{28}$ and/or $^{24}O^{20}$ to be a "0". When the $^{17}V^{54}$ and $^{13}V^{54}$ gate circuits are fully enabled at phase 3, (the internal interrupt out flip-flop) V^{54} is set. With V^{54} set, the "1" output from $^{00}V^{54}$ disables Out-on-2. The "1" output from $^{02}V^{54}$ disables the other lower modes in subpriority by disabling the $^{19}V^{5}$- circuits (fig. 13-11).

The inputs to $^{19}V^{54}$ are a "0" output from $^{01}V^{54}$, the "0" output from $^{02}V^{55}$ indicating no External Interrupt request, and the "0" output from $^{02}V^{56}$ indicating no Real-Time Clock request. A "1" output from $^{19}V^{54}$ forces the output from $^{20}V^{54}$ to a "0". This output is one of the enables required to enable $^{96}N^{24}$ (fig. 13-6) for the transmission of the request to priority. Again the "1" output from $^{19}V^{54}$ (fig. 13-10) disables the Memory Access flip-flop (fig. 13-13) and provides an enable to load the storage register. The two outputs from $^{19}V^{54}$ (fig. 13-10) to $^{15}V^{05}$ and $^{15}V^{04}$ in priority (fig. 13-12) set the initial address of the Internal Interrupt Out Entrance register to 00060. This is the special address for channel 0 and is incremented by the channel number of the particular channel being referenced. For this internal interrupt the special addresses are from 00060 through 00075. Again I/O relinquishes control of the interrupt to main computer control.

INTERNAL INTERRUPT IN

The same sequence of event follows for this interrupt that enabled the previous two, except that the input monitors (not shown) generate the Interrupt signal. The ^{19}I-- circuits in the input monitors provide the inputs to $^{24}I^{28}$ and $^{24}I^{20}$ (fig. 13-10) from a particular channel. When the gate circuits $^{17}V^{53}$, and/or $^{13}V^{53}$ are fully enabled at phase 3, V^{53} is set. With V^{53} set, the "1" output from $^{00}V^{53}$ disables Out-on-2 by disabling $^{13}V^{50}$ and disables the other lower modes by disabling the $^{19}V^{5-}$circuits (fig. 13-11). The inputs to $^{19}V^{53}$ (fig. 13-10) are a "0" output from $^{01}V^{53}$ and the "0" outputs from the higher priority modes (already discussed), if they are not being processed. Again, a "1" output from $^{19}V^{53}$ disables the Memory Access flip-flop (memory is not used during interrupt modes). Another "1" output forces the output from $^{20}V^{53}$ to a "0". This is used to enable $^{93}N^{24}$ (fig. 13-6) for the transmission of the request to priority.

The "1" output from $^{19}V^{53}$ to $^{22}V^{53}$ provides an enable to load the Interrupt Storage register (circuit not shown). The "1" output to $^{15}V^{05}$ (fig. 13-12) in priority sets the initial address of the Internal Interrupt In Entrance register to 00040. This is the special address for channel 0 and it is incremented by the channel number of the particular channel being referenced. For this internal interrupt, the special addresses are from 00040 through 00055. Again, control of the interrupt is passed from I/O to main computer control.

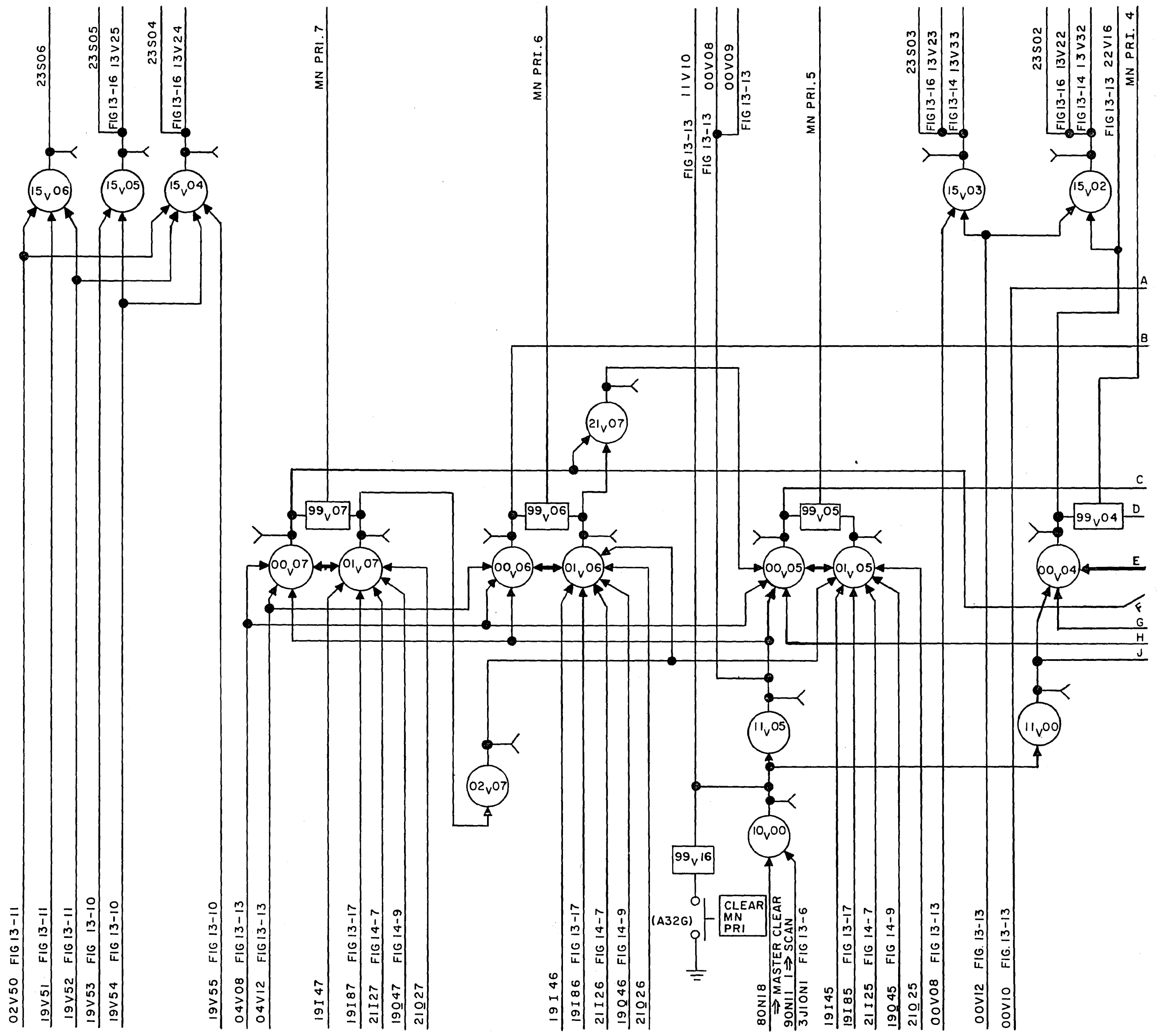
TIME-SHARE CIRCUIT

In subpriority where Output-on-12 has precedence over input, and input has precedence over Output-on-2, it is conceivable that the computer could be locked-out (prevented from honoring) an input mode or an output mode. In order to prevent this, a Time-Share circuit (fig. 13-11) is incorporated into subpriority.

The principle of operation of the time-share circuit is as follows. Assume that an input request is being honored. While this request is being executed, certain circuits are enabled to prevent another input request from being honored in sequence if an output request is waiting. If an Output-on-2 request is waiting for example, it would be honored before another input request even though Output-on-2 has lower priority. Likewise, the time share circuit prevents output requests from following one another. If only one type of request is waiting on different channels, the time-share circuit merely causes a delay of 8 μsec between sequential requests.

OUT-ON-12

This circuit in subpriority processes output requests for the normal channels (channels 2



through 13). The inputs to $^{24}0^{46}$ and $^{24}0^{40}$ (fig. 13-11) are from the $^{15}0^{--}$ circuits of the output request gates (not shown). When a request is up, the output from a $^{15}0$-- circuit is a "1" that forces the output from $^{24}0^{46}$ and/or $^{24}0^{40}$ to a "0". When Scan is reset by I/O timing, the command signal input from $^{90}N^{13}$ (fig. 13-6) to $^{12}V^{51}$ (fig. 13-11) forces the $^{12}V^{51}$ output to a "0". The output from $^{12}V^{52}$ is a "0" if the last request processed was not an output request.

The gate circuits, $^{17}V^{52}$ and/or $^{13}V^{52}$, are fully enabled at phase 3 to set V^{52}. With V^{52} set, the "1" output from $^{00}V^{52}$ disables an input by disabling $^{19}V^{51}$, and disables Out-on-2 by disabling $^{13}V^{50}$. The inputs to $^{19}V^{52}$ from the higher priority circuits are "0's" as long

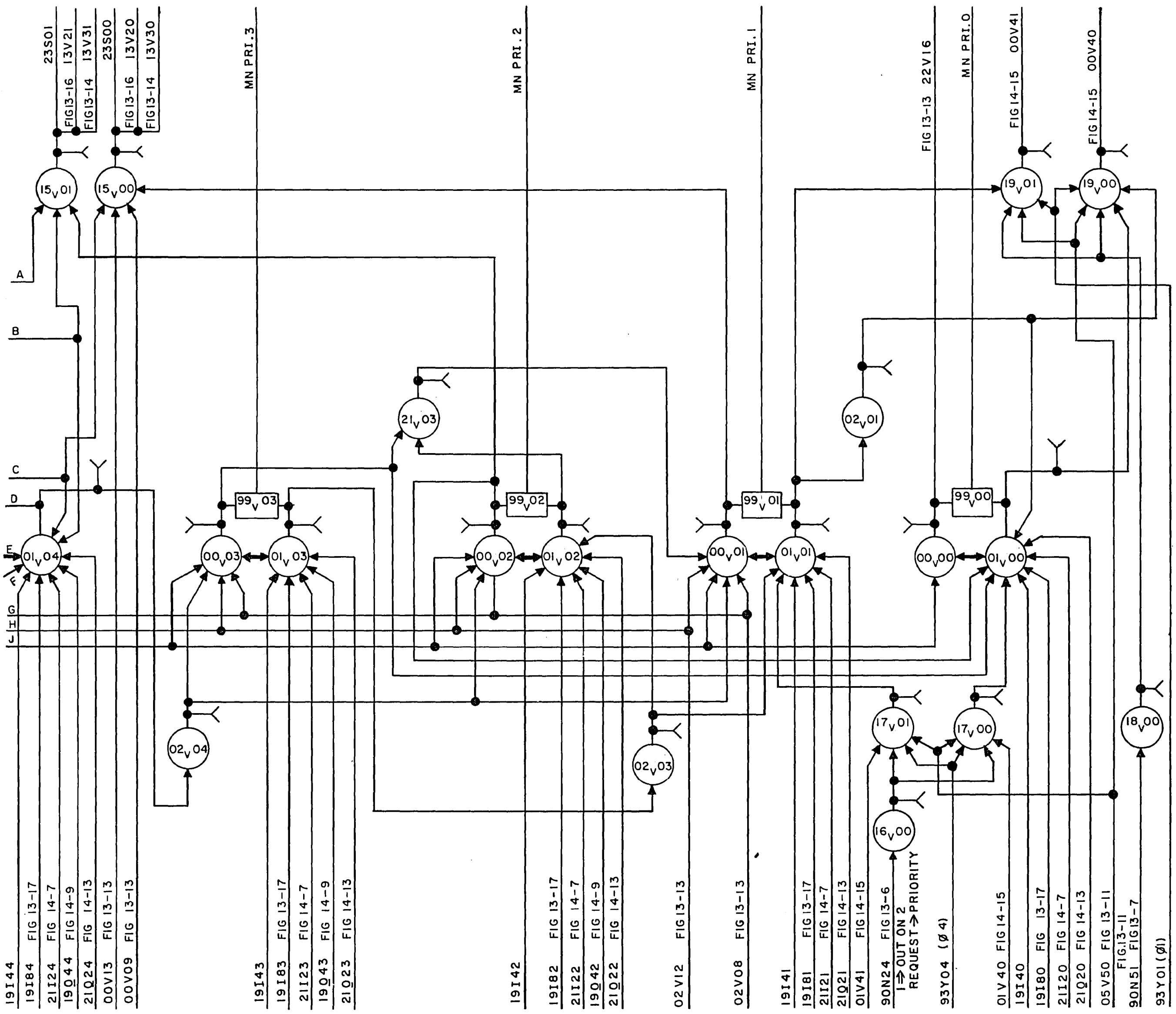
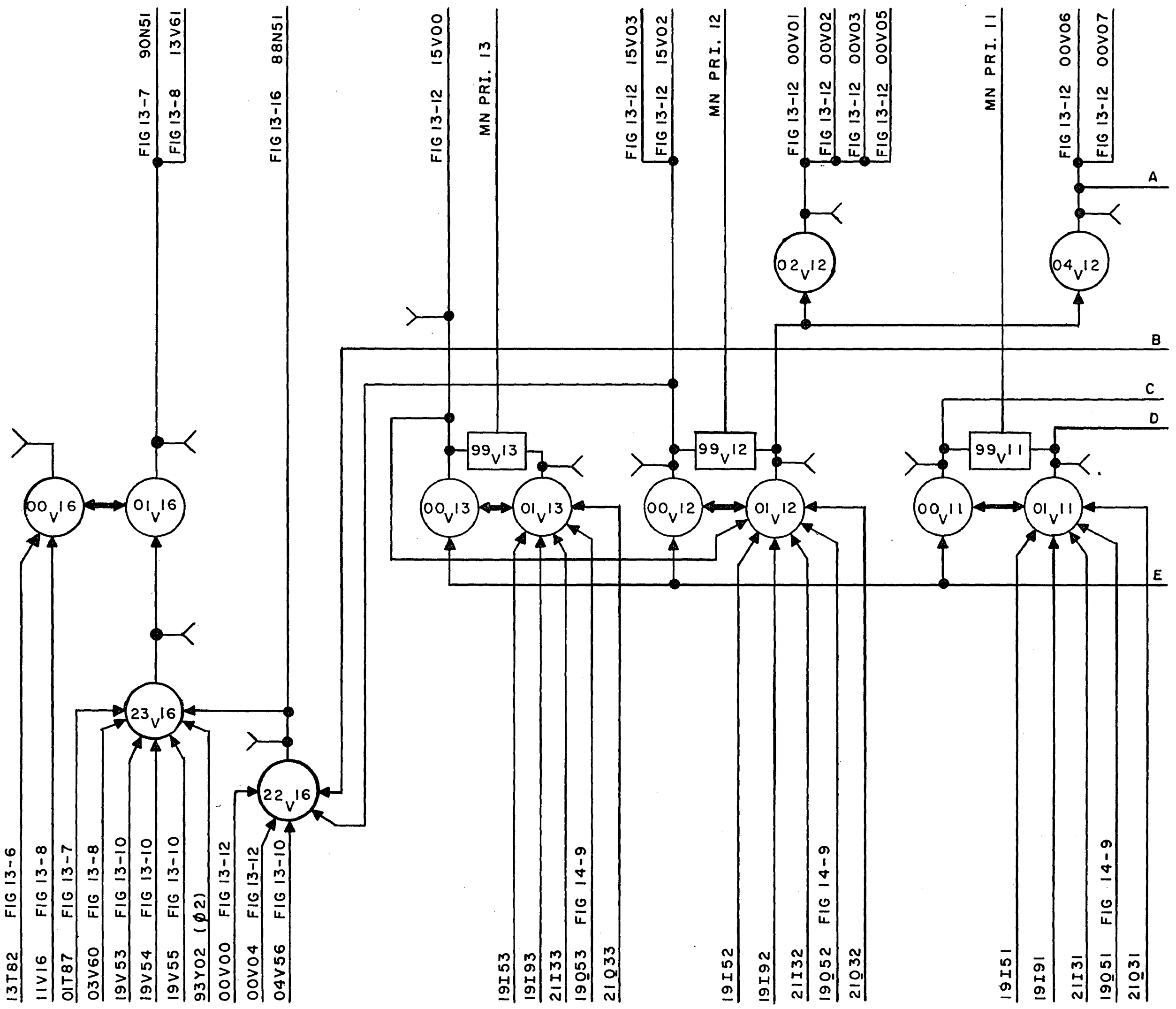

124.181

Figure 13-12.—Priority, Part 1.

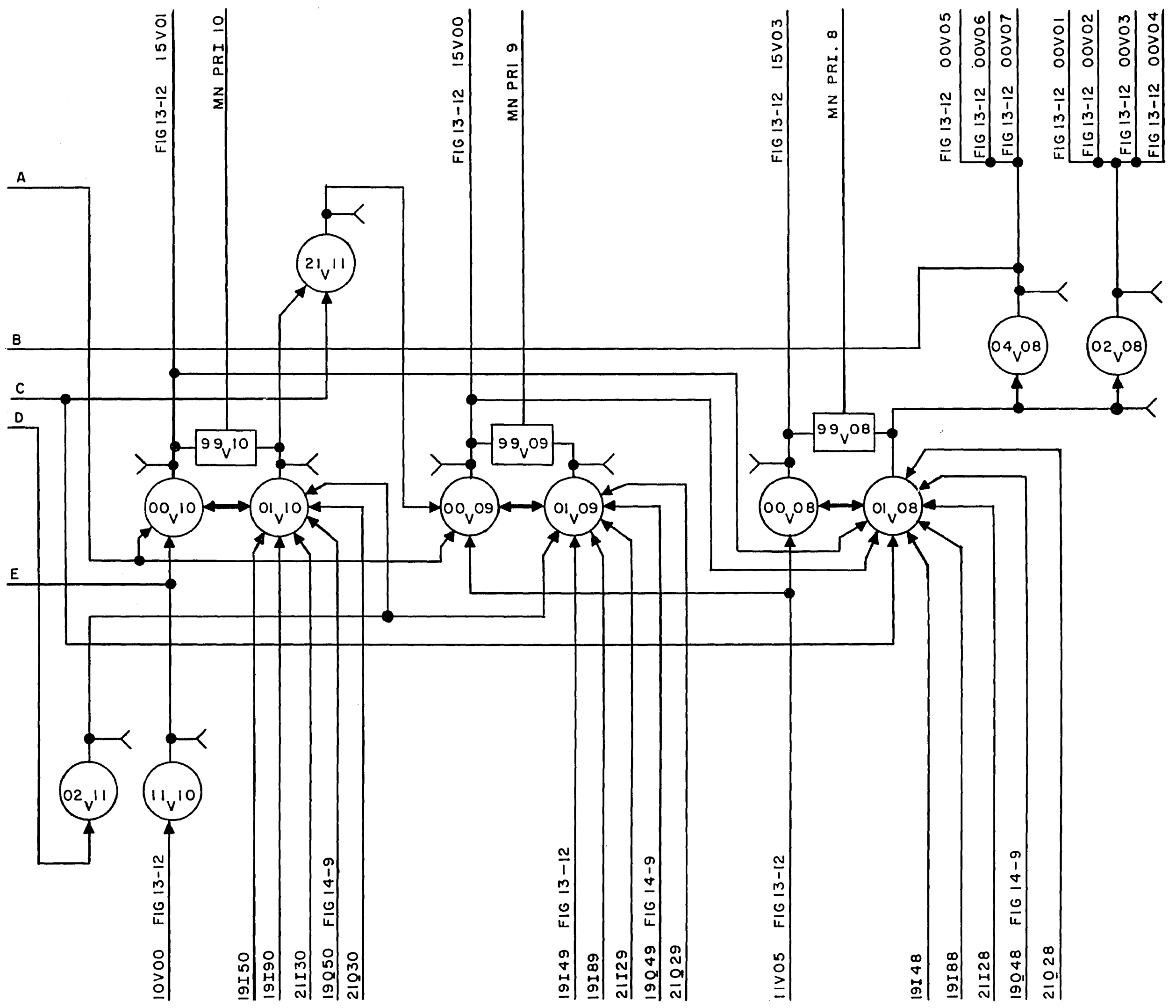
as there is no request being processed by these circuits. With the "0" input from $^{01}V^{52}$, the output from $^{19}V^{52}$ is a "1" that forces the output from $^{20}V^{52}$ to a "0". The output from $^{20}V^{52}$ is one of the enables required by $^{97}N^{24}$ (fig. 13-6) for the transmission of the request to priority. The "1" output from $^{19}V^{52}$ (fig. 13-11) to $^{15}V^{06}$ and $^{15}V^{04}$ in priority (fig. 13-12) sets up the initial address of the Output Buffer Control register to 00120, which is the address for channel 0. In this case for Out-on-12, address 00120 must be incremented by a channel number, since the lowest channel used is channel 2. The addresses for the Output Buffer Control registers however, are from 00120 through 00135.



INPUT

This circuit in subpriority processes input requests on all channels. The inputs to $^{24}I^{48}$ and $^{24}I^{40}$ (fig. 13-11) are from the ^{15}I-- circuits of the input request gates (not shown). When a request is up, the output from a ^{15}I-- circuit is a "1" that forces the output from $^{24}I^{48}$ and/or $^{24}I^{40}$ to a "0". The output from $^{12}V^{59}$ is a "0" if the last request processed was not an input. After the Scan sequence is reset, the command signal is generated. The gate circuits, $^{17}V^{51}$ and/or $^{13}V^{51}$, are fully enabled at phase 3 to set V^{51}. With V^{51} set, the "1" output from $^{00}V^{51}$ disables Out-on-2 by disabling $^{13}V^{50}$. If the inputs to $^{19}V^{51}$ from the higher-priority circuits are "0's", the output from $^{19}V^{51}$ is a "1". This forces the output from $^{20}V^{51}$ to "0". The output from $^{20}V^{51}$ is one of the enables required by $^{89}N^{24}$ (fig. 13-6) for the transmission of the request to priority.

The "1" output from $^{19}V^{51}$ (fig. 13-11) to $^{15}V^{06}$ (fig. 13-13) in priority sets up the initial address of the input buffer control register to



124.182

Figure 13-13.—Priority, Part 2.

00100, which is the address for channel 0. This address is incremented by the channel number to reference the specified address. The Input Buffer Control registers utilize addresses 00100 through 00115.

At time $T88_1$ the "0" output from $20V51$ (fig. 13-11) to $93N51$ (fig. 13-7) is used in conjunction with an output from the buffer comparator. If the contents of the lower half of address 00100 + (channel number) equal the contents of the upper half of the same address ($Z_U = Z_L$) the buffer has been completed. At this time $93N51$ is enabled to clear the Input Active flip-flop (not shown) on the designated channel. This completes the action of subpriority for an Input request.

The Out-on-2 mode is concerned with intercomputer communications and is explained later.

MAIN PRIORITY

The main priority circuit (figures 13-12 and 13-13) is used to determine the channel priority and to set up the special address for a particular mode of operation. The priority network utilizes 14 flip-flops ($^{00}V^{00}$ thru $^{00}V^{13}$) connected so that they are able to designate a specific channel number from 0 through 13. Since the higher numbered channels have priority over the lower numbered channels, the network specifies a particular channel by the flip-flop (or flip-flops) that is set. The priority circuit utilizes translation circuits $^{15}V^{03}$ through $^{15}V^{00}$ (fig. 13-12) to provide select enables for the designated channel.

All flip-flops in priority are cleared by the same command enables. An associated main priority manual clear indicator-pushbutton (MN PRI in figure 10-3), when depressed clamps the output from $^{10}V^{00}$ (fig. 13-12) at a "0" via $^{99}V^{16}$. This "0" output from $^{10}V^{00}$ is inverted to a "1" output by $^{11}V^{00}$, and $^{11}V^{05}$, (fig. 13-12) and $^{11}V^{10}$ (fig. 13-13) to clear priority.

When the MASTER CLEAR switch is depressed, the input from $^{80}N^{18}$ is a "1" which forces the output from $^{10}V^{00}$ to a "0". The clearing of priority is then the same as for manual clear.

During Scan (at the start of the I/O timing sequence), a clear command is generated. At this time, the input to $^{10}V^{00}$ is a "1" from $^{90}N^{11}$. Priority is then cleared in the same manner as for master clear.

Following the determination by subpriority of a mode of operation, requests are loaded into priority to determine which channel shall use his selected mode of operation. The flip-flop corresponding to the designated channel is set and other flip-flops may be set according to the requirements of the special address. When a flip-flop is set its corresponding indicator on the console is lighted.

Each of the $^{01}V^{--}$ sides of the priority flip-flops has five inputs that are capable of setting the flip-flop under certain conditions. The inputs are from gate circuits (not shown) associated with a particular mode of operation, as follows:

1. External Interrupt
2. Output Monitor
3. Input Monitor
4. Output Request Gate
5. Input Request Gate

The output from any of the associated gate circuits is a "1" when it is active. This output is one of the signals required to load the request into priority. The command to load priority is generated during Scan when $^{90}N^{24}$ (fig. 13-6) is enabled. The explanation for an output request on channels 0 and 1 is explained later.

MODE ADDRESS

A specific mode address is assigned to each channel for each of the seven modes of operation. The designated channel number is a function of the outputs from $^{15}V^{00}$ through $^{15}V^{03}$ (fig. 13-12) with the full mode address being a function of the outputs from $^{15}V^{00}$ through $^{15}V^{06}$. The outputs from $^{15}V^{06}$, $^{15}V^{05}$, $^{15}V^{04}$ depend solely upon inputs from subpriority which indicate the selection of the mode of operation. The outputs from $^{15}V^{03}$ through $^{15}V^{00}$ depend upon inputs from priority determined by the selected channel.

Since the outputs from the $^{15}V^{--}$ circuits are transmitted to gate circuits in the S-register (the Interrupt storage register, and the I/O translator), the enable output is a "0". In this manner, when a "1" appears in the address, it appears as "0" output from $^{15}V^{--}$ circuit; conversely, a "0" in the address appears as a "1" output from a $^{15}V^{--}$ circuit. The full mode address is loaded into the S-register (not shown) so the address can be referenced in memory.

The command to send the address to S is generated during I/O_1 when $^{96}N^{11}$ (fig. 13-6) is enabled. The Interrupt address is loaded into the Interrupt storage registers in the V-designator circuits (shown later) to be held for use during the next sequential A sequence. The command to send the Interrupt address to the storage register is generated during Scan. The designated channel number is loaded into the I/O translator (shown later) to generate enable signals required by I/O during the processing of a request. The command to send the channel number to the translator is generated during Disable I/O_1 when $^{94}N^{42}$ (fig. 13-7) is enabled.

Along with the address enable signals, priority also provides enables to set the Memory Access flip-flop (fig. 13-13). The outputs from $^{00}V^{00}$ and $^{00}V^{04}$ (fig. 13-12) and $^{00}V^{08}$, and $^{00}V^{12}$ (fig. 13-13) are fed to $^{22}V^{16}$. For a designated channel, the output from one of the above circuits is always a "1". (One of the flip-flops is set whenever there is a request on any channel.) Table 13-2 shows the priority setup for any selected channel.

MEMORY ACCESS FLIP-FLOP

This flip-flop is set whenever I/O has a request that requires a memory reference through I/O control. The flip-flop, (V^{16}) is unconditionally cleared by the "1" output from $^{13}T^{82}$ (fig. 13-6). Also, when the MASTER CLEAR switch is depressed, the "1" output from $^{11}V^{16}$ produces a "1" output which clears V^{16}.

The setting of the memory access flip-flop is conditioned by a number of factors. First, the output from $^{22}V^{16}$ must be a "0", i.e., any of its inputs must be a "1". A Real-Time Clock request to $^{22}V^{16}$ is signified by a "1" output from $^{04}V^{56}$ (fig. 13-10), while a request on any channel is signified by a "1" output from any one of the other four inputs from priority.

Second, the $^{23}V^{16}$ gate circuit must be fully enabled. For this to occur, the inputs from $^{19}V^{53}$, $^{19}V^{54}$, and $^{19}V^{55}$ must be "0's", signifying that there is no type of interrupt being processed by I/O (as discussed earlier). Now, during Scan ($^{03}V^{60}$ in figure 13-8 producing a "0" output) and at time T^{87} (where flip-flop T^{87} is set), the gate is fully enabled at phase 2. With the gate circuit enabled, all conditions have been satisfied and V^{16} is set. The "0" output from $^{01}V^{16}$ provides one of the enable inputs to $^{90}N^{51}$ (fig. 13-7). A "1" output from $^{90}N^{51}$ is used to clear the Scan flip-flop at time $T88_1$. The other output from $^{01}V^{16}$ is one of the required enable inputs to $^{13}V^{61}$ (fig. 13-8). The "1" output from $^{13}V^{61}$ is used to set V^{61} (I/O_1). With this action taking place, a memory reference is being made by I/O.

There is an output from $^{22}V^{16}$ (fig. 13-13) to $^{88}N^{51}$ in the Interrupt storage register (V-designator, shown later) which is used as a conditional enable for setting the Interrupt flip-flop. With no request in priority, the output from $^{22}V^{16}$ is a "1", which enables $^{88}N^{51}$. This output is used so that if an interrupt is being processed by subpriority and the Interrupt Control signal drops before processing by priority, a false interrupt on channel 0 does not take place.

I/O TRANSLATOR

The translator used in the I/O section (fig. 13-14), provides a unique translation for a certain specific channel that is being utilized or specified. The translation effectively takes place in a four-bit register (V^{33} thru V^{30}). The outputs from the four translator flip-flops are the inputs to the translator output circuits (a translator output circuit is provided for each channel). These circuits provide the multiple enable signals required by I/O to initially set a channel active; to designate a channel for a specific mode of operation; or to designate a channel on which an interrupt has occurred. Associated with the translator are three flip-flops (V^{36} thru V^{34}) which, when set, designate an input mode or an output mode of operation.

CHANNEL TRANSLATION

As aforestated, the four flip-flops that perform the channel translation are V^{33} through V^{30}. The flip-flops are cleared on phase 1 and loaded on phase 2 for three separate cases as follows:

1. During the normal sequence (see figure 10-10) at time T^{31} (when T^{31} is set on phase 2), the output from $^{11}T^{31}$ is a "0" which is inverted to a "1" output by $^{14}T^{31}$. This output forces the output from $^{11}V^{31}$ to a "0" that is applied as an enable input to $^{11}V^{32}$ (fig. 13-14) in the translator. On the following phase 1, the output from $^{11}V^{32}$ is a "1" that cleared the V^{33} thru $V30$ flip-flops. At time T^{33} (phase 4) in the A sequence (fig. 10-10) the output from $^{11}T^{33}$ is a "0" that is inverted to a "1" output by $^{14}T^{33}$. This output forces the output from $^{14}V^{30}$ (fig. 13-14) to a "0". This signal is one of the enables required by the $^{15}V^{3-}$ circuits at phase 2 to load the translator.

2. During the interrupt A sequence at time T^{11} (phase 2), the output from $^{10}T^{11}$ (fig. 10-10) is a "1" that forces the output from $^{11}V^{31}$ to a "0". This "0" is applied as an enable input to $^{11}V^{32}$ in the translator. On the following phase 1, the "1" output from $^{11}V^{32}$ clears the four translator flip-flops. At time T^{13} (phase 4) in the Interrupt A sequence (fig. 10-10) the "1" output from $^{10}T^{13}$ forces the output from $^{15}T^{13}$ to a "0". This "0" is one of the enables required by the $^{17}V^{3-}$ circuits (fig. 13-14) at phase 2 to load the translator during an interrupt.

3. During the processing of a request by I/O at time $T86_1$ (fig. 13-7) and during Disable I/O_1, the associated flip-flops are set and the outputs from $^{01}T^{86}$ (fig. 13-7) and $^{01}V^{64}$ (fig. 13-8) are "0's". These are applied as inputs to $^{11}V^{30}$ in the translator (fig. 13-14). At phase 1 the output from $^{11}V^{30}$ is a "1" that clears the four translator flip-flops. At time $T87_1$ (T87 in figure 13-7 set) and during Disable I/O_1 (V^{64} in figure 13-8 set), the output from $^{94}N^{42}$ (fig. 13-7) is a "1" that forces the output from $^{12}V^{30}$ in the translator (fig. 13-14)

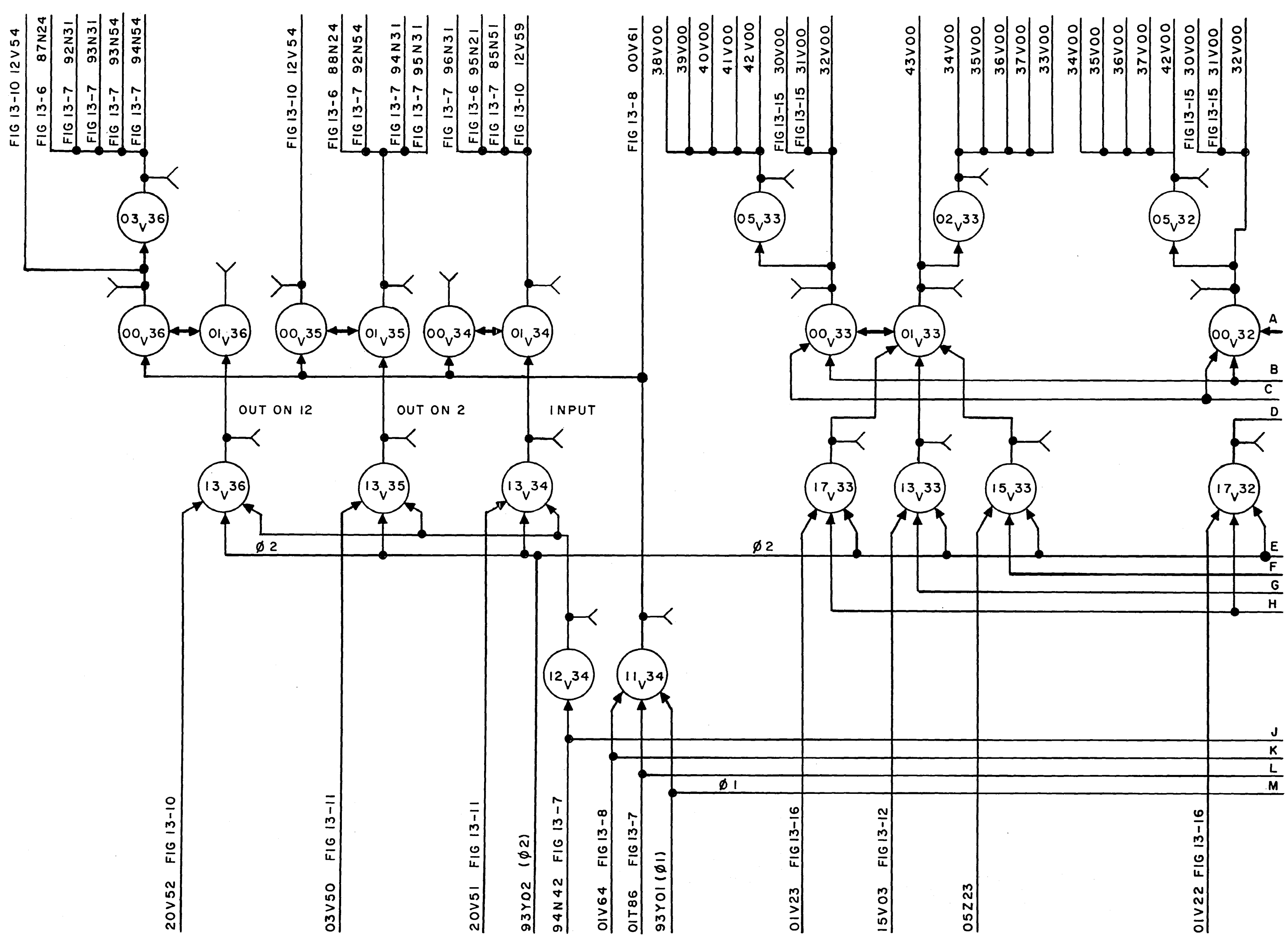


Figure 13-14.—Translator flip-flops.

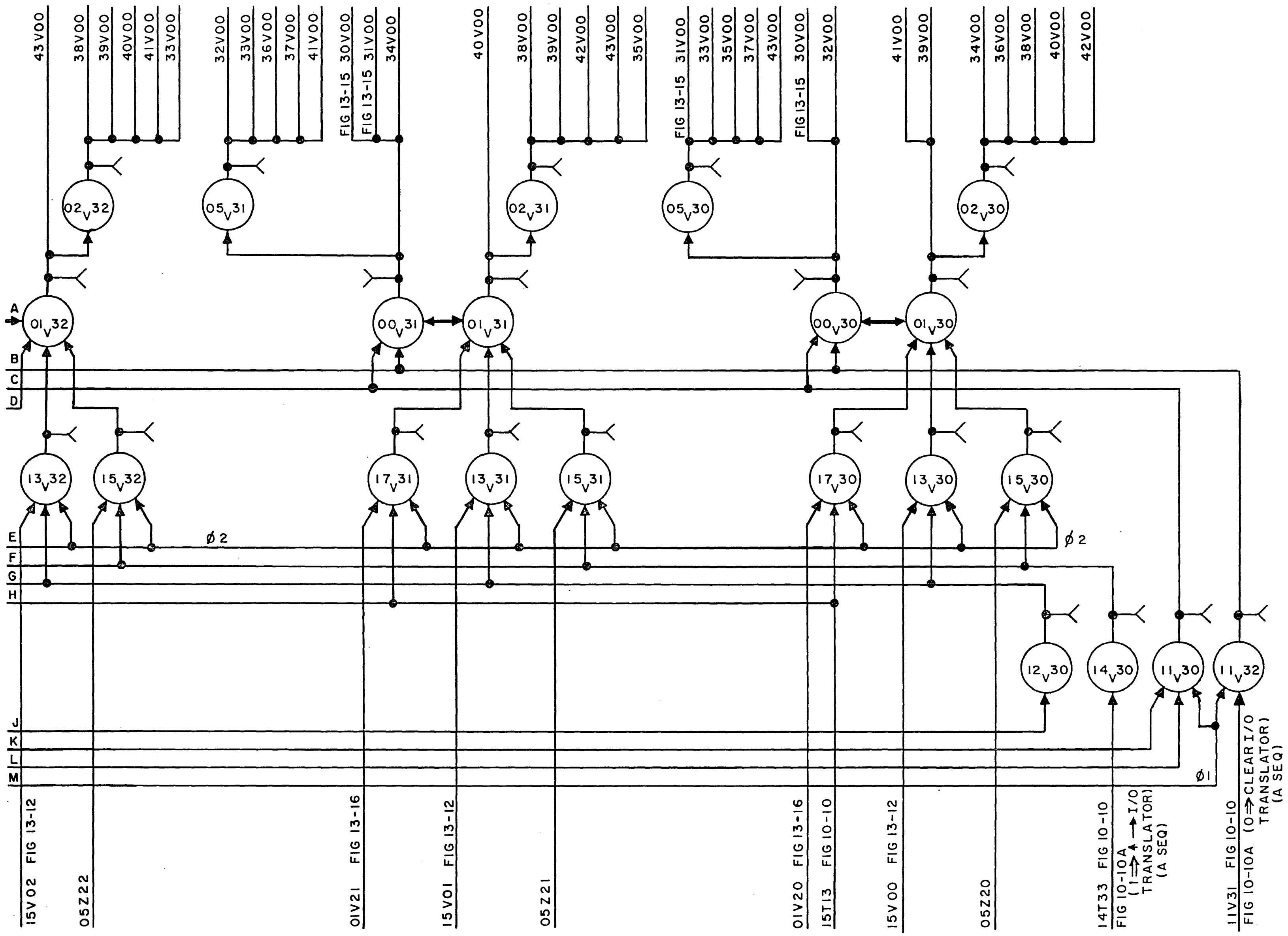


Figure 13-14.—Translator flip-flops.—Continued.

124.183

to a "0". This "0" is one of the enables required by the $^{13}V^{3}$- circuits at phase 2 to load the translator during the processing of a request by I/O or during disable I/O_1.

$\hat{j}$ DESIGNATOR

The data inputs to the $^{15}V^{3}$- circuits (fig. 13-14) are from the z-register, (Z^{33} through Z^{30}) and are used for a special interpretation of the j designator. The normal j-designator (as discussed) is three bits in length; but when it is being utilized by I/O, it becomes four bits in length. The upper bit of the k-designator becomes the lower bit of the new j-designator that is referred to as $\hat{j}$. This usage restricts the maximum value of k to three and designates it as $\hat{k}$.

The four bits of $\hat{j}$ represent a specific channel number in I/O, any channel from 0 through 13. At phase 2, the $^{15}V^{3}$- circuits which are fully enabled depend upon the designated channel. If channel 0 is specified, the outputs from selected state of the Z-register associated with flip-flops Z^{23} thru Z^{20} (see fig. 12-15) are "1's" that disable the $^{15}V^{3}$- circuits (fig. 13-14). The four translator flip-flops remain cleared and their outputs signify a designation for channel 0. If channel 13 is specified, the outputs from inverters associated with flip-flops Z^{23}, Z^{22}, and Z^{20} are "0's" and the output from Z^{21} is a "1". Thus, V^{33}, V^{32}, and V^{30} are set and V^{31} remains cleared. In binary notation this is 1101 which is 13_{10}. The outputs from the flip-flops signify a designation for channel 13.

Note that the translator is loaded each time an instruction is carried through the normal A sequence. However, the only time that the information is utilized is when the instruction is concerned with I/O (f = 13, 17, 62, 63, 66, 67, and 73-76 as shown in table 10-2).

CHANNEL DESIGNATION

The data input to the $^{13}V^{3}$- (fig. 13-14) circuits are from the $^{15}V^{0}$- circuits in priority (fig. 13-12). As shown in table 13-2, the output from the $^{15}V^{0}$- circuits indicates a specific channel on which an input or an output operation is taking place. Therefore, the outputs from the $^{15}V^{0}$- circuits enable or disable the $^{13}V^{3}$- circuits according to the channel specified. The inputs to the $^{17}V^{3}$- circuits (fig. 13-14) are from $^{01}V^{23}$ through $^{01}V^{20}$ in the Interrupt storage register (V designator, discussed later). Here, the designation is for a channel on which an interrupt has occurred as discussed earlier.

Table 13-2.—Priority Setup.

SELECTED CHANNEL	CHANNEL FF SET	AUXILIARY FF'S SET	ACTIVE 15V--CIRCUIT(S) ("0" OUTPUT)	MEMORY ACCESS FF ENABLE
0	V00	----------	----------	00V00
1	V01	V00	15V00	00V00
2	V02	V00	15V01	00V00
3	V03	V00, V01, V02	14V01, 15V00	00V00
4	V04	------------	15V02	00V04
5	V05	V04	15V02, 15V00	00V04
6	V06	V04	15V02, 15V01	00V04
7	V07	V04, V05, V06	15V02, 15V01, 15V00	00V04
8	V08	------------	15V03	00V08
9	V09	V08	15V03, 15V00	00V08
10	V10	V08	15V03, 15V01	00V08
11	V11	V08, V09, V10	15V03, 15V01, 15V00	00V08
12	V12	------------	15V03, 15V02	00V12
13	V13	V12	15V03, 15V02, 15V00	00V12

TRANSLATOR OUTPUT

The outputs from V^{33} through V^{30} (fig. 13-14) are fed to 14 output circuits, $^{30}V^{00}$ through $^{43}V^{00}$. All translator output circuits function the same so that the $^{30}V^{00}$ and $^{31}V^{00}$ circuits of figure 13-15 are representative.

When a channel is designated by the four translator flip-flops, the four inputs to only one of the output circuits are "0's", so that the output from only one of the circuits, ($^{30}V^{00}$ or $^{31}V^{00}$ in this case) is a "1". (In the actual circuit the four translator input are applied to all translator output circuits.) This means that the output circuits also provide a unique translation for a specified channel and only that channel.

Included in each output circuit along with the translator is a number of inverters that provide enable signals to the I/O section. These enable signals from a given channel are used in the I/O section whenever the associated channel is selected.

MODE FLIP-FLOPS

The three mode flip-flops (V^{36}, V^{35}, and V^{34} in figure 13-14) indicate, when set, an input or an output request being processed by I/O. The three flip-flops are cleared by the output from $^{11}V^{34}$ during the processing of request by I/O simultaneously with the other circuits in the translator, as explained previously. When the command is generated by the I/O timing chain (figs. 13-6 and fig. 13-7) to load the translator, it is also utilized as an enable to set one of the mode flip-flops. (The "1" output from $^{94}N^{42}$ in figure 13-7 forces the output from $^{12}V^{34}$ to a "0".) The other enable to the $^{13}V^{3-}$ circuits is from subpriority (fig. 13-11). Now, which flip-flop is set at phase 2 depends upon the type of request currently being processed by subpriority. If it is an Out-on-12, the output from $^{20}V^{52}$ is a "0" causing V^{36} to be set. If it is an Out-on-2, the output from $^{03}V^{50}$ is an "0" causing V^{35} to be set. If it is an Input, the "0" output from $^{20}V^{51}$ causes V^{34} to be set. Therefore, the translator generates signals indicating that a particular Output mode or an Input mode is being carried out on a designated channel. The outputs from the mode flip-flops are used by the I/O timing chain and the Time-Share circuit in subpriority.

v-DESIGNATOR

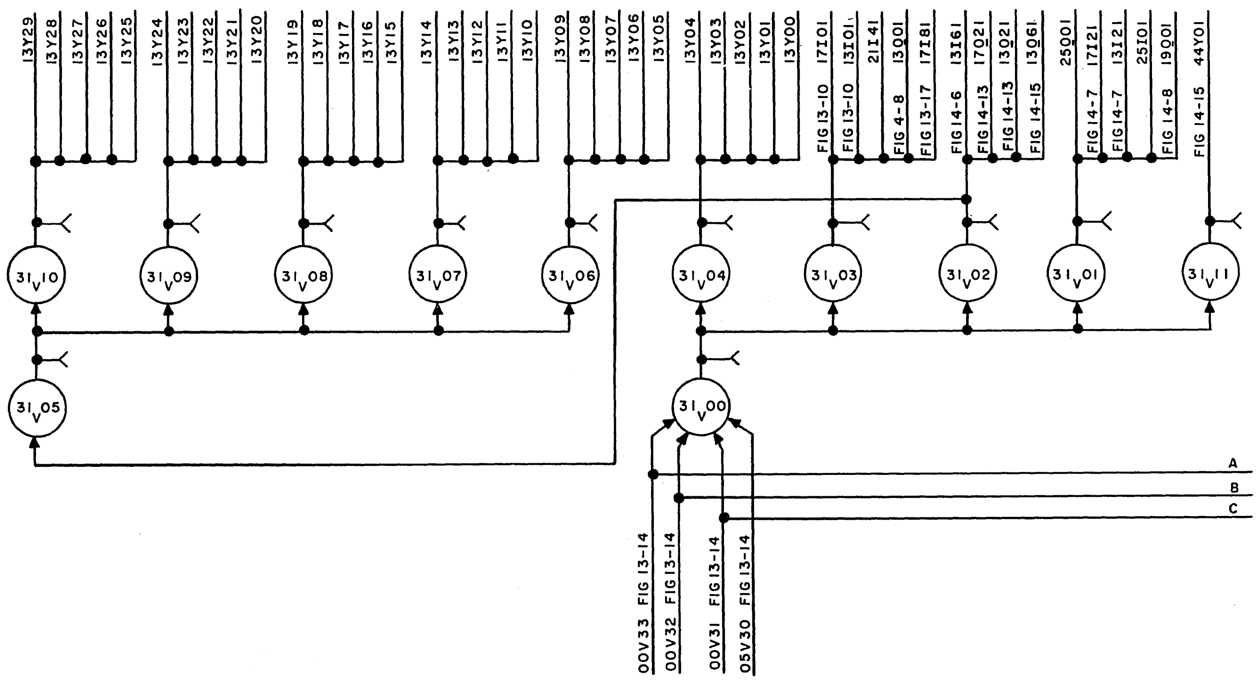
The v designator (fig. 13-16) consists of a six-bit Interrupt Address storage register (flip-flops V^{25} thru V^{20}), the Interrupt flip-flop (V^{57}), and the Fault Entrance flip-flop (V^{66}). The Interrupt flip-flop (V^{57}) is set, (1) when an Interrupt occurs in I/O, (2) when there is a program fault (f = 00 or 77), or (3) when the wired memory mode flip-flop (G^{80}) is set. The Fault Entrance flip-flop is set for conditions (2) and (3) above, but not for condition (1). The storage register is loaded only during an interrupt with the address of the particular interrupt entrance register to be referenced.

The v-designator can be cleared by master clear or manual clear. When the MASTER CLEAR switch (fig. 10-9) is activated, the "1" output from an inverter in the command enable circuit (not shown) forces the output from $^{10}V^{20}$ (fig. 13-16) to a "0". This output enables $^{11}V^{20}$ and $^{11}V^{23}$, at phase 1, to clear the v designator. When the clear v designator indicator-pushbutton (Vin) figure 10-8, is depressed, the output from $^{10}V^{20}$ and the input to $^{11}V^{23}$ and $^{11}V^{20}$ are clamped at a "0". Again at phase 1, the two circuits are enabled to clear the designator.

Also concerned with the clearing of the v designator is a circuit, $^{88}N^{51}$ (fig. 13-16) which is used to load the register and set V^{57} during an interrupt. When an interrupt is processed by subpriority (whether it is an external interrupt an internal interrupt in or an internal interrupt out), one of the inputs via $^{19}V^{53}$, $^{19}V^{54}$, $^{19}V^{55}$ to $^{22}V^{53}$ is a "1" that forces its output to a "0". As this request is processed through priority to designate a specific channel, a "1" output from priority forces the output from $^{22}V^{16}$ to a "0" (fig. 13-13) as discussed earlier. Now at time T^{87} during Scan, $^{01}T^{87}$ (fig. 13-7) supplies a 0 input to $^{88}N^{51}$ (fig. 13-16) causing this circuit to be fully enabled. The "1" output from $^{88}N^{51}$ forces the output from $^{10}V^{20}$ to a "0". This "0" level output is used to enable the clearing of the V-designator at phase 1. The output from $^{88}N^{51}$ is also the signal used to load the interrupt address storage register and to set V^{57} (the interrupt flip-flop) on the following phase 2.

INTERRUPT FLIP-FLOP

When V^{57} is set, the presence of an Interrupt mode, a Program Fault mode, or a Bootstrap



(wired memory) mode is indicated. For an Interrupt mode, the "1" output from $^{88}N^{51}$ forces the output from $^{12}V^{20}$ to a "0". This output enables $^{13}V^{57}$ at phase 2 to set V^{57}.

For a Bootstrap mode or a program fault, the output from $^{63}N^{07}$ (B sequence fig. 10-15) is a "0". This output enables $^{15}V^{57}$ at phase 2 to set V^{57}. As long as V^{57} is set, the "1" outputs from $^{00}V^{57}$ and $^{02}V^{57}$ prevent the processing of any interrupt request by subpriority. This ensures that the original condition that caused the flip-flop to be set will be satisfied. The "0" output from $^{01}V^{57}$ is used as a conditional enable to $^{61}N^{11}$ in the A sequence.

In order to clear V^{57} after an interrupt or bootstrap mode, a 60 instruction with a j = 0 or j = 1 is programmed (see table 10-2). During the following C sequence, V^{57} is cleared.

FAULT ENTRANCE FLIP-FLOP

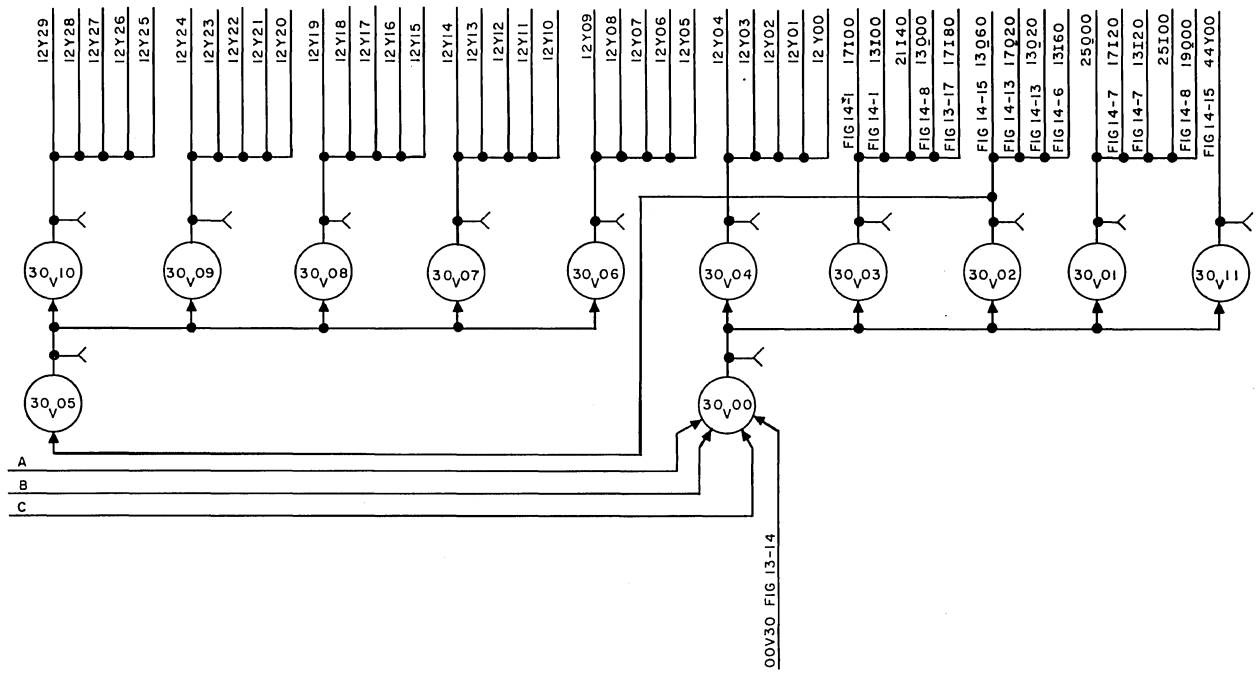
The primary function of V^{66} (fig. 13-16) is to initiate the Program Fault mode. For a Program Fault mode, the output from $^{63}N^{07}$ (B sequence fig. 10-15) is a "0". This output enables $^{15}V^{57}$ (fig. 13-16) at phase 2 to set V^{66} along with V^{57}. A "0" output from $^{01}V^{66}$ is used as a conditional enable in the A sequence to initiate a fault routine. The flip-flop is cleared by a signal from the A sequence. At time T31 in the A sequence (fig. 10-10) the "1" output from $^{14}T^{31}$ clears V^{66} (fig. 13-16).

INTERRUPT ADDRESS STORAGE REGISTER

This register is only loaded during an interrupt. The "1" output from $^{88}N^{51}$ forces the outputs from $^{12}V^{20}$ and $^{12}V^{23}$ to "0's", which are applied as inputs to the $^{13}V^{2-}$ circuits. A second input to these circuits is from priority and is the address of the particular interrupt. At phase 2 the enabling of the $^{13}V^{2-}$ circuits depends upon the outputs from the $^{15}V^{0-}$ circuits in priority. In other words, the setting of a register flip-flop is conditioned by the interrupt address.

Because I/O does not exercise control of the interrupt after the register is loaded, the information is held in the register until acted upon by the A sequence.

The upper two bits of the interrupt address, the outputs from V^{24} and V^{25}, are translated by three circuits to generate reset signals. These signals clear the circuits that originally generated the interrupt. If an external interrupt has



124.184

Figure 13-15.—Translator outputs (0-1).

been generated, the address must be between 20_8 and 35_8 (the special addresses for an external interval). In binary notation the address is between 010000 and 011101 and the usable portion of the upper octal bit is 01-. This means that V^{25} remains cleared and V^{24} is set (a condition which will always exist when an external interrupt is generated). The only circuit that could be enabled at this time is $22V^{24}$ with "0" inputs from $00V^{25}$ and $01V^{24}$. The output from this circuit is used to reset the external interrupt gates.

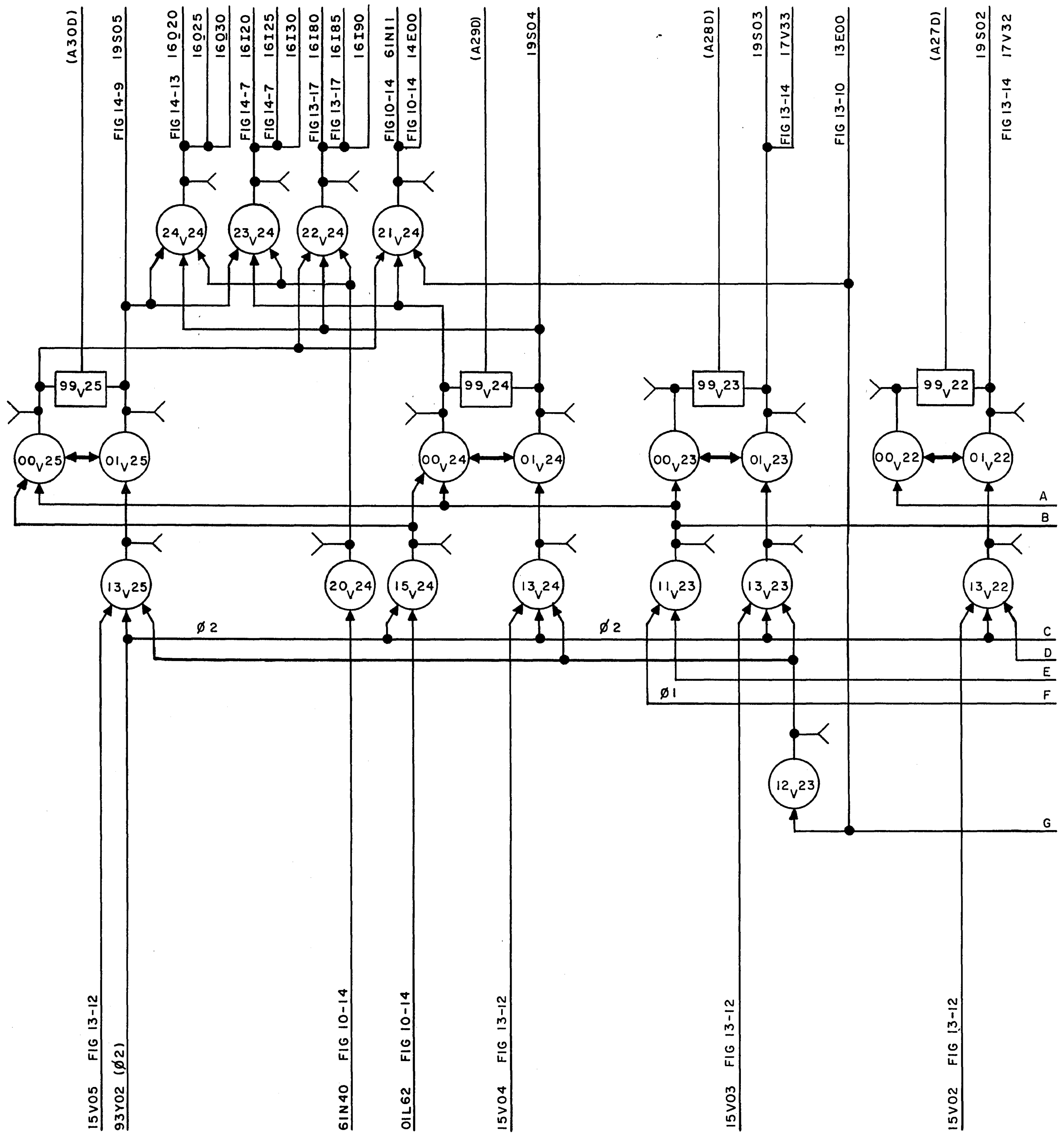
If an Internal Interrupt In has been generated, the address must be between 40_8 and 55_8. This means that the usable upper bits are always 10- that set V^{25} and keep V^{24} cleared. The circuit that could be enabled at this time is $23V^{24}$ whose output is used to reset the input monitors. The actions to fully enable $23V^{24}$ is discussed later.

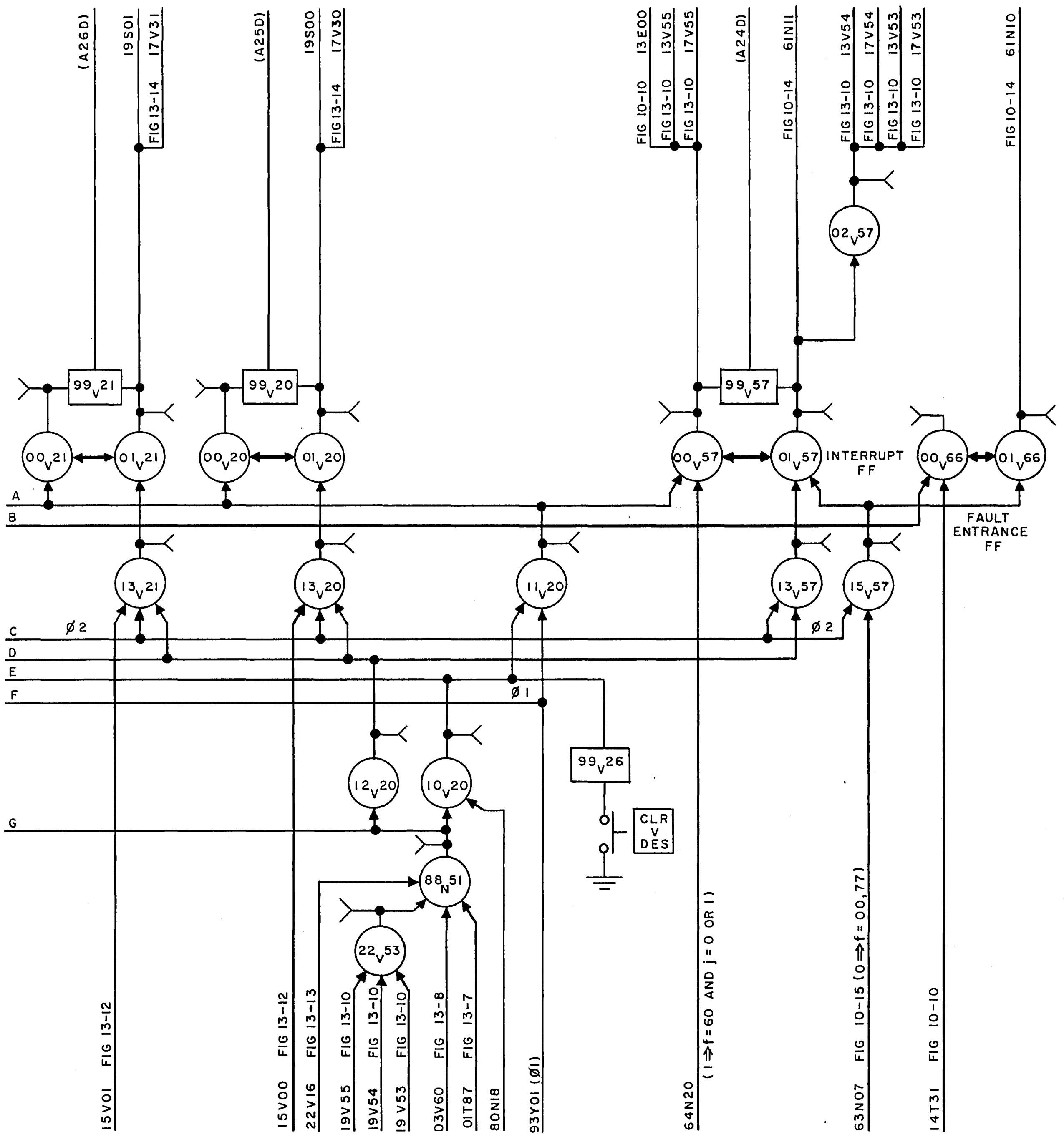
The last type of interrupt, Internal Interrupt Out, utilized addresses from 60_8 to 75_8. This means that the upper bits are 11- that set both V^{25} and V^{24}. The circuit that could be enabled at this time is $24V^{24}$, whose output is used to reset the output monitors. The command signal which fully enabled the required reset circuit is generated during the A sequence.

A SEQUENCE CONTROL

The transfer of data and control signals from the Interrupt storage register is performed under the control of the A sequence (fig. 10-10). An interrupt, when it is processed by the computer, causes the execution of the instruction stored at a special address. You will recall that the A sequence is not fully enabled until $13T^{00}$ is enabled to produce a "1" output. Thus, before an interrupt can be processed by the A sequence, a set of initial conditions must be satisfied in order to enable $13T^{00}$ (fig. 10-10). The conditions are as follows:

CIRCUIT	OUTPUT	CONDITION
$11T^{00}$	"0"	A sequence enabled
$16E^{00}$	"0"	No Repeat mode
$03G^{73}$	"0"	High-Speed mode
$57H01$	"0"	Memory available and I/O_1 and I/O_2 cleared
$14E^{00}$	"0"	j Hold flip-flops cleared (no skip evaluation or skip evaluation completed)


(A30D)
FIG 14-9 19S05
FIG 14-13 16Q20
16Q25
16Q30
FIG 14-7 16I20
FIG 14-7 16I25
16I30
FIG 13-17 16I80
FIG 13-17 16I85
16I90
FIG 10-14 6IN11
FIG 10-14 14E00
(A29D)
19S04
(A28D)
19S03
FIG 13-14 17V33
FIG 13-10 13E00
(A27D)
19S02
FIG 13-14 17V32
24V24
23V24
22V24
21V24
99V25
99V24
99V23
99V22
00V25
01V25
00V24
01V24
00V23
01V23
00V22
01V22
13V25
20V24
15V24
13V24
11V23
13V23
13V22
12V23
Ø 2
Ø 2
Ø 1
A
B
C
D
E
F
G
15V05 FIG 13-12
93Y02 (Ø2)
6IN40 FIG 10-14
01L62 FIG 10-14
15V04 FIG 13-12
15V03 FIG 13-12
15V02 FIG 13-12



124.185

Figure 13-16.—v designator.

At phase 2 $^{13}T^{00}$ is enabled to set T11 which allows the A sequence to run.

Running in parallel with the normal A sequence is another timing chain (fig. 10-14) associated with the interrupt. It is also started by satisfying a set of initial conditions that enable $^{61}N^{11}$ as follows.

CIRCUIT	OUTPUT	CONTITION
$^{15}T^{11}$	"0"	Normal A sequence running
$^{00}G^{29}$	"0"	No Abort mode
$^{42}F^{07}$	"0"	$f \neq 00, 77$ (no program fault)
$^{90}F^{70}$	"0"	$f \neq 70$ (no repeat)
$^{01}V^{57}$	"0"	Interrupt flip-flop set
$^{21}V^{24}$	"0"	Interrupt present in I/O

When $^{61}N^{11}$ is fully enabled, its "1" output disables the normal P adder $\longrightarrow$ S transmission by disabling $^{61}N^{12}$ (fig. 10-10). Instead, it provides a signal to the S-register to load the interrupt address and, also, a signal to start the Interrupt timing chain via $^{12}L^{60}$ (fig. 10-14). The interrupt address is loaded from the storage register into S at phase 1 so this special address in memory can be referenced. At time T^{11}, the "1" output from $^{10}T^{11}$ (fig. 10-10) forces the output from $^{11}V^{31}$ to "0". This output is used to enable $^{11}V^{32}$ (fig. 13-14) at phase 1 to clear the I/O translator. At time T^{13}, the "1" output from $^{10}T^{13}$ (fig. 10-14) forces the output form $^{15}T^{13}$ to "0". This output is used to enable the transmission of the designated interrupt channel from the storage register (fig. 13-14) to the I/O translator. At time L^{61} when L^{61} (fig. 10-14) is set, the "0" output from $^{01}L^{61}$, coincident with the "0" output from $^{42}F^{07}$ ($f \neq 00, 77$), enables $^{61}N^{40}$. The "1" output from $^{61}N^{40}$ forces the output from $^{20}V^{24}$ (fig. 13-16) to "0" that enables the required reset circuit in the storage register. This is the final input required to clear the reset circuits after an interrupt. The output from the reset circuit (fig. 13-16) coupled with the designated channel signal from the I/O translator (fig. 13-14), resets the Interrupt circuit at phase 1.

At time L^{62} (fig. 10-14) the "1" output from $^{00}L^{62}$ disables $^{61}N^{20}$ (fig. 10-10) to prevent the clearing of the P-register, to prevent the S $\longrightarrow$ P transmission, and to prevent the setting of the p-designator to +1. Effectively, this provides a means of returning to the main program after the completion of the Interrupt subroutine by the programming of a Return Jump instruction from the interrupt address to the interrupt subroutine.

Aslo at time L^{62} (fig. 10-14), the "0" output from $^{01}L^{62}$ is an enable signal to $^{15}V^{24}$ (fig. 13-16) in the storage register. At phase 2, the "1" output from $^{15}V^{24}$ clears V^{24} and V^{25}. With these two flip-flops cleared and the output from $^{88}N^{51}$ returned to "0", the output from $^{21}V^{24}$ becomes "1". This "1" output prevents the Interrupt timing chain from recycling by disabling $^{61}N^{11}$ (fig. 10-14). Instead, it provides an enable to run the normal A sequence by forcing the output from $^{14}E^{00}$ (fig. 10-10) to "0". At time T31, the subsequent part of the A sequence and the other sequences run in a normal manner.

EXTERNAL INTERRUPT GATES

There is an external interrupt gate associated with each channel. The signal from the external interrupt gates informs the computer that there is information of a special nature on the data lines. The external interrupt gate circuit for channels 0 thru 7 is shown in figure 13-17. A special consideration is made for channels 0 and 1 which is covered in the next chapter.

When the computer is master cleared the "1" input from $^{80}N^{16}$ (not shown) forces the output from $^{10}I^{80}$ (fig. 13-17) to a "0". This "0" output is inverted to a "1" output by $^{11}I^{80}$ and $^{11}I^{84}$ (fig. 13-17) and by $^{11}I^{88}$, and $^{11}I^{92}$ (not shown) to set the external interrupt flip-flops, I^{80} and I^{87}. Since the "0" output from the $^{01}I^{--}$ sides of these flip-flops is utilized as an enable signal, the setting of the flip-flops, in this case, is in anticipation of a waiting external interrupt. Under normal conditions, following an external interrupt, the flip-flop is reset or cleared. This makes it necessary for the external equipment to drop the interrupt line and bring it back up again before interrupting the next time.

The Interrupt Control signal is entered into the computer from the external equipment by an input amplifier, a $^{40}Y^{--}$ circuit. With no signal on the line, the input to the amplifier is line logic (-13 v-d-c) a "0". The output from the amplifier is also a computer logic (0 v-d-c) "0". This is inverted to a "1" output by a $^{13}I^{--}$ circuit to set an interrupt flip-flop. When the control signal is placed on the line, the input to the amplifier is a "1". The output from the amplifier is also a "1" forcing the output from the ^{13}I-- circuit to a "0". Now, one of the ^{15}I-- circuits has a "0" input from an associated flip-flop and from one of the ^{13}I-- circuit resulting in a "1" output. This "1" output from ^{15}I-- forces the output from a ^{24}I-- circuit (figures 13-10 and 13-11) in subpriority to a "0". A "0" output from a ^{24}I-- circuit informs subpriority of the presence of an external interrupt. When subpriority has processed the request, a command is generated by the I/O timing chain during Scan to load priority. At this time the output from ^{92}N^{24} (fig. 13-6) is a "1" that forces the output of the ^{18}I-- circuits (fig. 13-17) to a "0". This output is one of the enables required by the ^{19}I-- gates. Which of the ^{19}I-- circuits is fully enabled at phase 4 depends on the channel or channels on which the interrupt occurred. The "1" output from the ^{19}I-- circuit sets the associated channel flip-flop in priority (figures 13-12 and 13-13) that determines the channel request to be honored. If, for some reason, the interrupt line drops (output from ^{13}I-- returns to a "1") before the ^{19}I-- gate is enabled, no interrupt is generated.

When the interrupt is being processed by the A sequence, a reset command signal is generated at time L^{61} (fig. 10-14). The output from ^{22}V^{24} (fig. 13-16) becomes a "1" which forces the output from the ^{16}I-- circuits (fig. 13-17) to a "0". This output is one of the enable inputs to the ^{17}I-- circuits. Also during the A sequence the interrupt channel number is sent to the I/O translator which provides another enable input to a ^{17}I-- circuit. The ^{17}I-- circuit associated with the particular channel on which the interrupt occurred is fully enabled at phase 1. The "1" output from the circuit resets or clears the associated flip-flop to nullify the Interrupt request currently being honored.

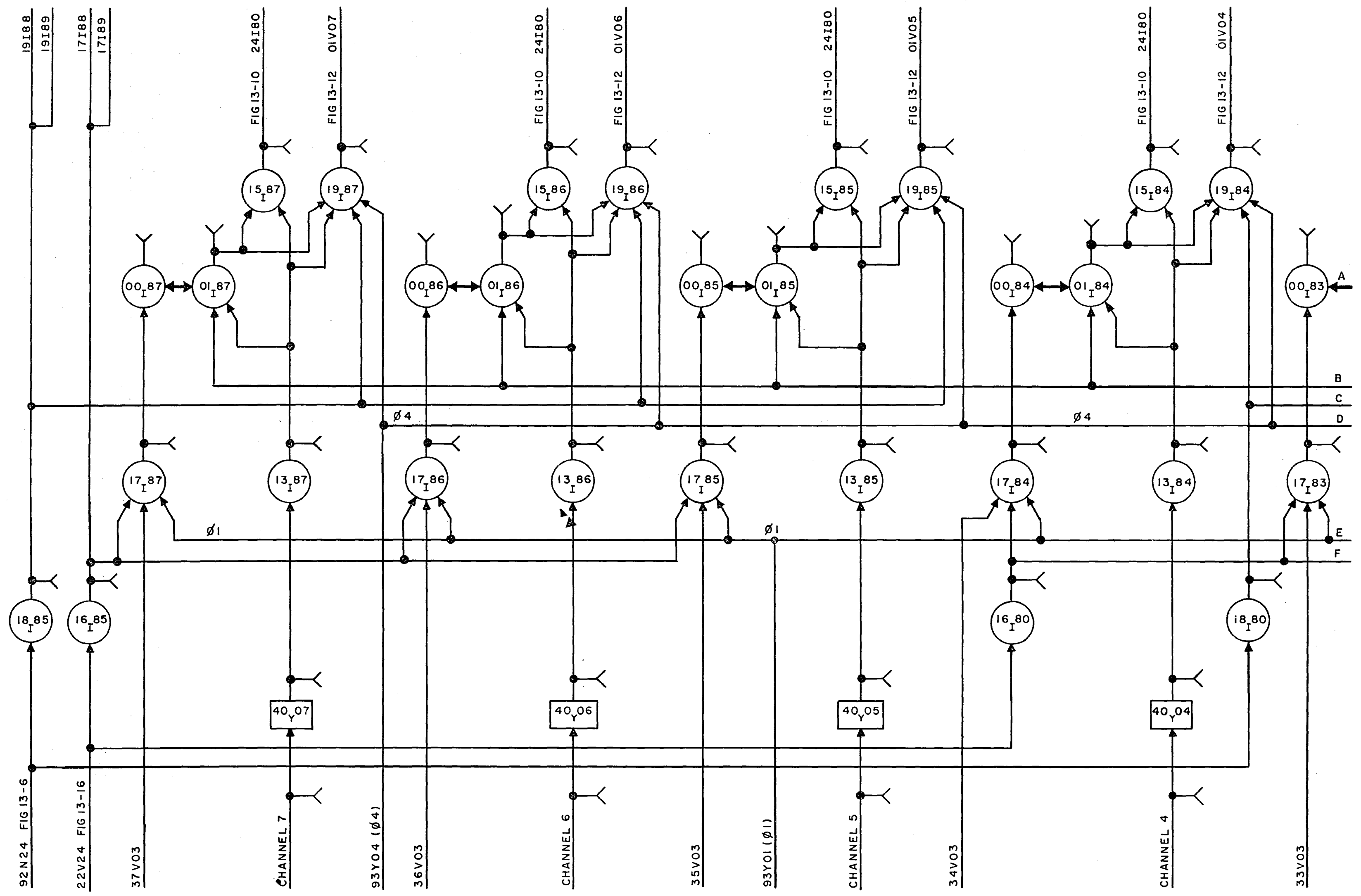
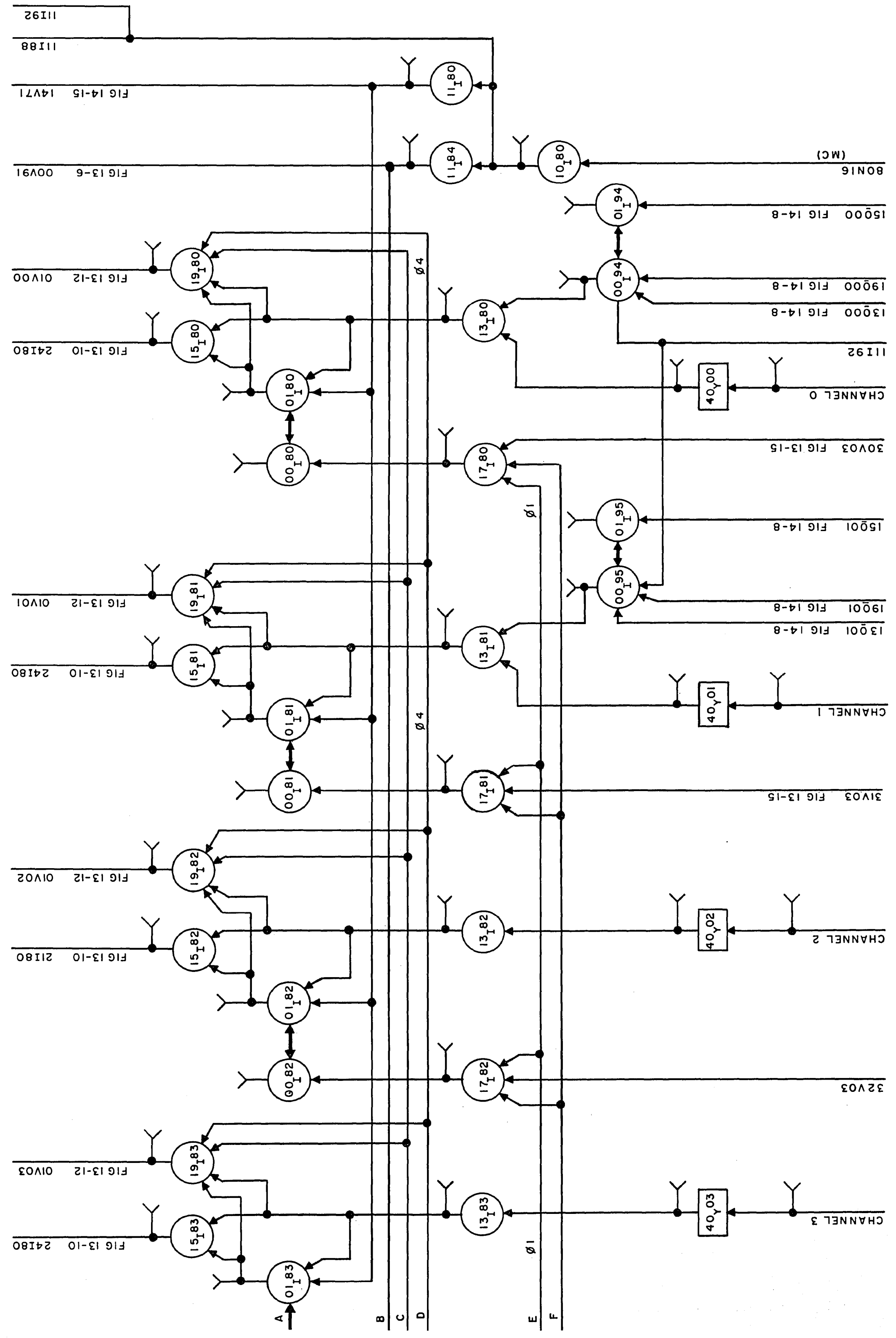


Figure 13-17.—External interrupt gates 0-7.



124.186

Figure 13-17.—External interrupt gates 0-7—Continued.

CHAPTER 14

NTDS COMPUTER INPUT/OUTPUT SECTION (PART II)

The circuits utilized by I/O to perform an input function are made up of data handling circuits and control circuits. Before data can be entered into the computer from an external equipment, the active flip-flop which activates the specific channel by which the data will enter must be set active. This activation is a function of main control and programming.

INPUT CIRCUITS

When the external equipment has placed the data on its output lines (the computer input lines), the computer is informed by a request signal being sent from the associated external equipment. After the computer has sampled the data lines, the external equipment is informed by a control signal sent from the computer. Under certain conditions, as determined by main computer control and programming, an internal control signal can be generated that informs the computer when an input buffer action is completed.

INPUT ACTIVE FLIP-FLOPS

There are 14 input active flip-flops (figures 14-1 and 14-2). Each of the 14 input channels has an Input Active flip-flop associated with it I^{00} thru I^{13}. Whenever a given flip-flop is set, the associated channel is active or capable of handling input data. When that flip-flop is cleared the channel is inactive or not capable of handling input data.

All of the flip-flops can be cleared by the MASTER CLEAR switch (fig. 10-9). When the switch is depressed, the "1" output from $^{80}N^{16}$ (which indicates MASTER CLEAR) forces the output from $^{10}I^{00}$ (fig. 14-1) to a "0". This signal is inverted to a "1" output by $^{11}I^{00}$ and $^{11}I^{05}$ (fig. 14-1) and by $^{11}I^{10}$ (fig. 14-2) to clear all of the input active flip-flops. Also, depressing the associated manual clear indicator-pushbutton (IN ACT in fig. 10-8) clamps the output from $^{10}I^{00}$ at a "0" via $^{99}I^{16}$ (fig. 14-1). This signal is again inverted by the same circuits used by master clear (as just described) to clear the flip-flops.

The control of initially setting a channel active belongs to the main computer program. The general form of two instructions that may be used to set a channel active as via the program are as follows:

$$73 \quad \hat{j} \quad \hat{k} \quad b \quad y$$

$$75 \quad \hat{j} \quad \hat{k} \quad b \quad y$$

With f = 73, (see table 10-2) a channel is set active without a monitor. With f = 75, a channel is set active with a monitor.

During the A sequence, $\hat{j}$ is transmitted to the I/O translator (as discussed in chapter 13) to designate which channel is to be set active. At the proper time in the D sequence an inverter $^{65}N^{32}$ (not shown) is enabled. The "1" input from $^{65}N^{32}$ forces the outputs from $^{12}I^{00}$, and $^{12}I^{05}$ (fig. 14-1), and $^{12}I^{10}$ (fig. 14-2) to "0's". The "0" output from these circuits is one of the enables required by the $^{13}I^{--}$ circuits. Another enable (the channel designator) is from the I/O translator (fig. 13-15). Which $^{13}I^{--}$ circuit is fully enabled at phase 1 is now a function of the original value of $\hat{j}$ in the instruction. When a flip-flop is set to indicate an active channel, enable signals are provided to certain circuits in the I/O section (as discussed later).

When the external equipment sets its request line to indicate that it has data to read into the computer, the request is entered into the computer by input request gates (similar to that shown in figure 14-3). However, in order for the request to be processed completely by I/O the request must be entered into subpriority and priority. The "0" output (figures 14-1 and 14-2) from the $^{01}I^{--}$ sides of the input active flip-flops (I^{00} thru I^{13}) is used to enable the $^{15}I^{--}$ circuits of the input request gates which, in turn, enter the request into subpriority. This output is also used to enable the $^{19}I^{--}$ circuits which cause the request to be entered into priority.

Another output from $^{01}I^{--}$ is used in a special application (jump to $\underline{Y}$ if $C\hat{j}$ input buffer is active) which is a function of main computer control. The instruction is programmed as follows:

$$62 \quad \hat{j} \quad \hat{k} \quad b \quad y \qquad \text{(See table 10-2.)}$$

During the A sequence, $\hat{j}$ (which has been read into the instruction register, not shown) is transmitted to the I/O translator (fig. 13-14) to designate the referenced channel. The "0" output from the translator together with the "0" output from $^{01}I^{--}$ is utilized by a check circuit (Test I/O Active not shown). In this application an active channel is being referenced to satisfy a jump condition, f = 62 (see table 10-2). The output from only one of the inverters in the Test I/O Active circuit is a "1" according to the "0" inputs from the translator and the Input Active flip-flop. If the designated channel is active, the output from the Test I/O Active circuit is a "0" to a jump evaluation circuit (not shown) which satisfies the jump. If the designated channel is not active, the output from the Test I/O Active circuit to the jump evaluation circuit is a "1" which does not satisfy the jump.

The buffer can be terminated by the I/O timing chain when the initial and terminal addresses of the buffer are equal. This means that the transfer of the number of words initially specified by the buffer has been completed.

During Disable I/O_1 at time $T88_1$ (the second cycle of the I/O timing chain fig. 13-7), $^{93}N^{51}$ is enabled. This inverter receives a "0" input from $^{33}Z^{07}$ (not shown) when $Z_u = Z_L$. It also receives a "0" input from $^{20}V^{51}$ (not shown) to indicate an input operation is in progress. The "1" output from $^{93}N^{51}$ forces the output from $^{16}I^{00}$ and $^{16}I^{05}$ (fig. 14-1) and $^{16}I^{10}$ (fig. 14-2) to a "0" indicating completion of the buffer. The translator at this time holds the channel number of the designated channel. At phase 1, a $^{17}I^{--}$ circuit is fully enabled by the "0" inputs from the translator and a $^{16}I^{--}$ circuit. The "1" output from $^{17}I^{--}$ clears the Input Active flip-flop.

Another way to terminate the buffer, whether it has been completed or not, is a function of main computer control. The instruction is programmed as follows:

$$66 \quad \hat{j} \quad \hat{k} \quad b \quad y \qquad \text{(See table 10-2.)}$$

The I/O translator (fig. 13-14) is loaded with the designated channel number, $\hat{j}$ during the A sequence (by a command via $^{14}T^{33}$ in fig. 10-10). During the C sequence, a "1" output from an inverter, $^{64}N^{38}$, (not shown) forces the output from $^{16}I^{--}$ (fig. 14-1) to a "0". At phase 1, the referenced $^{17}I^{--}$ circuit is enabled to clear the Input Active flip-flop.

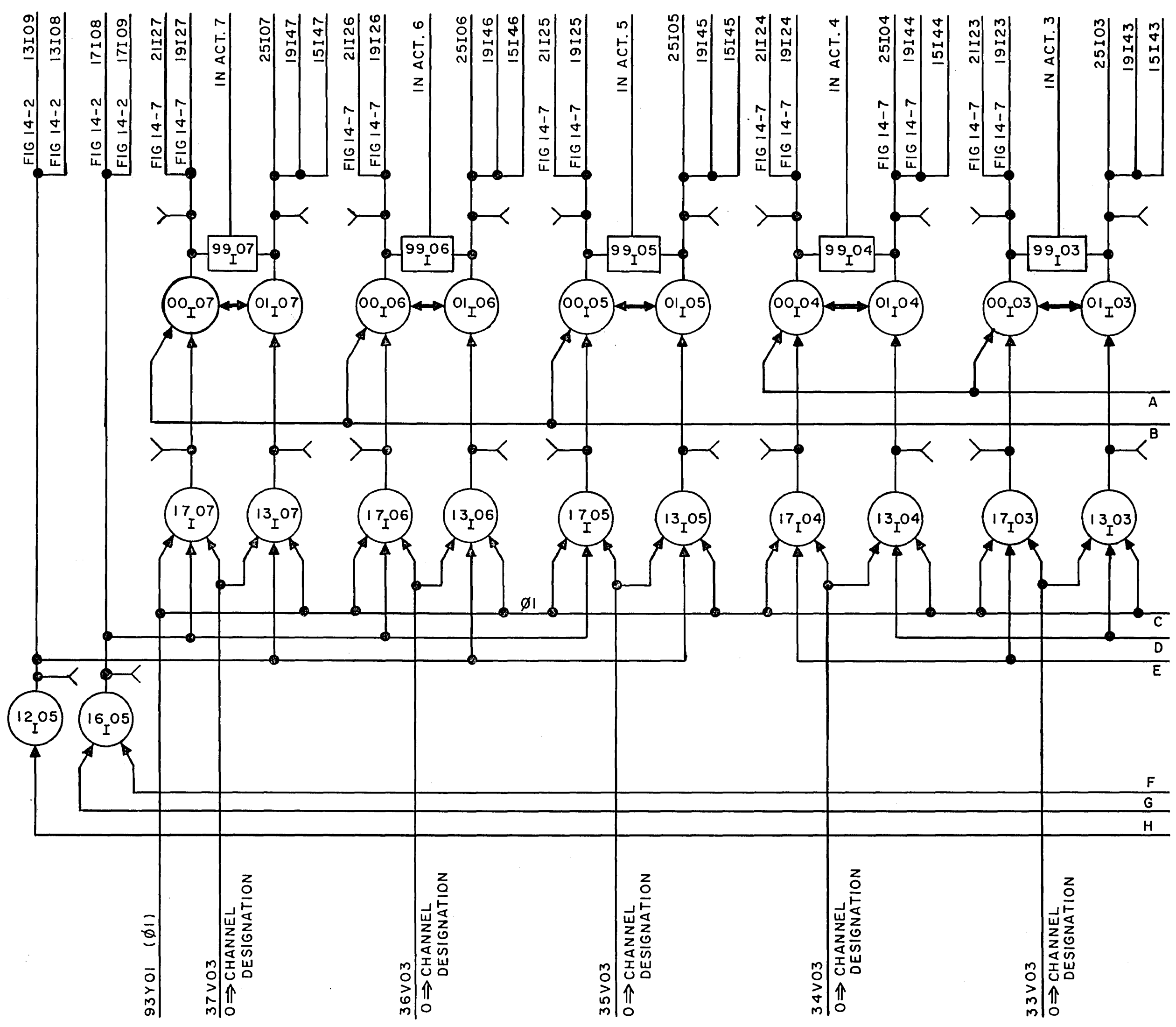
With the Input Active flip-flop cleared, enables are provided from $^{00}I^{--}$ to the input monitors. If the Input Active flip-flop had been originally set, using an instruction with f = 75, a provision was made to utilize the monitor at the completion of the buffer (Completion means all the words as defined by the buffer have been transmitted). The monitor generates an internal interrupt which is processed by I/O during Scan. The "0" output from $^{00}I^{--}$ is an enable to the $^{19}I^{--}$ circuits of the monitor to enter the request into subpriority. It is also an enable to the $^{21}I^{--}$ circuits of the monitor to enter the request into priority. However, no interrupt is generated when the Input Active flip-flops are cleared by the instruction with f = 66.

INPUT REQUEST GATES

When an external equipment has data on the computer input lines, it informs the computer by sending a signal called an Input Data request. This request is entered into the computer by Input Request gates (fig. 14-3) to be processed by I/O. Each channel has an associated Input Request gate.

When the MASTER CLEAR switch is depressed the "1" output from $^{80}N^{16}$ in the command enable circuit (not shown) forces the output from $^{10}I^{40}$ (fig. 14-3) to a "0". This "0" output is inverted to a "1" output by $^{11}I^{40}$, to set the flip-flop in anticipation of a waiting input request. (In the actual circuit, all of the input request flip-flops are set in this manner.)

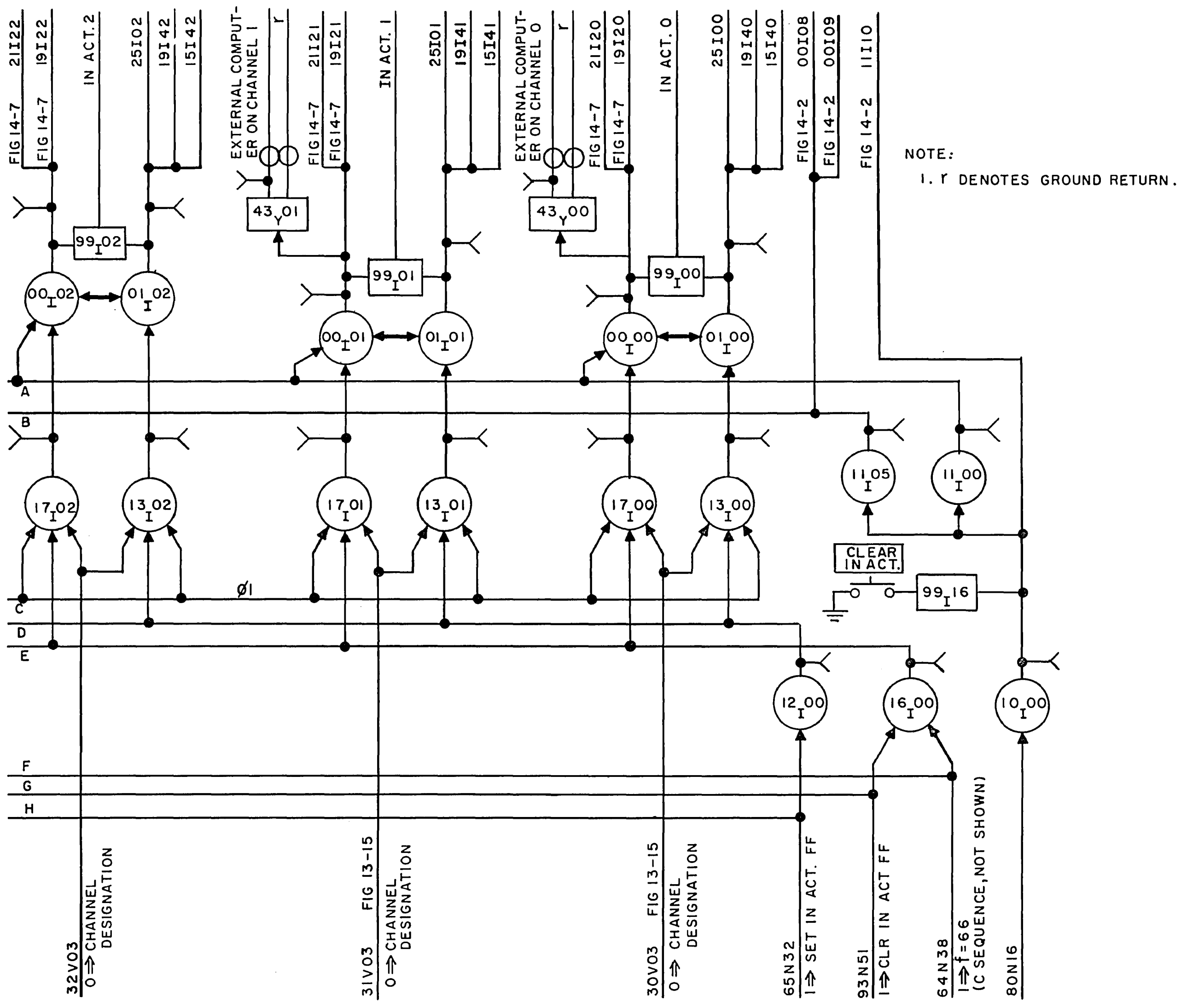
Under normal operating conditions, prior to the completion of a previous input request, the designated input request flip-flop is reset or cleared. When the request line drops, the input to the $^{11}Y^{--}$ circuit becomes a "0" indicating the end of a request. The output from the input



amplifier is now a "0" that is inverted to a "1" output by the ^{13}I-- circuit. This "1" output sets the Input Active flip-flop associated with the active channel.

When a request is transmitted to the computer, the signal on the input line becomes a "1". The output from the amplifier is a "1" forcing the output from the ^{13}I-- circuit to a "0". If the channel on which the request appears is active, its Input Active flip-flop is set. Thus, the inputs to the ^{15}I-- circuit are "0's" from the Request flip-flop, the Active flip-flop, and the ^{13}I-- circuit. The ^{15}I-- circuit is enabled and transmits a "1" signal to subpriority informing it of the Input request.

Since at any time there may be input requests on more than one channel, a signal must also be transmitted to priority to determine the channel to be honored. This is the function of the ^{19}I-- circuit (only one circuit shown in figure 14-3). Three of the enable signals required by ^{19}I-- are the same as those that enabled the ^{15}I-- circuits. In addition, a signal from the I/O timing chain is required. At time T83 in the e-designator timing chain (during Scan) the output from ^{89}N^{24} (fig. 13-6) is a "1" forcing the output from one of the ^{18}I-- circuits to a "0". At phase 4, the ^{19}I-- circuits (only one shown) are fully enabled by this "0" output from ^{18}I-- circuits and transmit a "1" signal to priority



124.187

Figure 14-1.—Input active FF's 0-7.

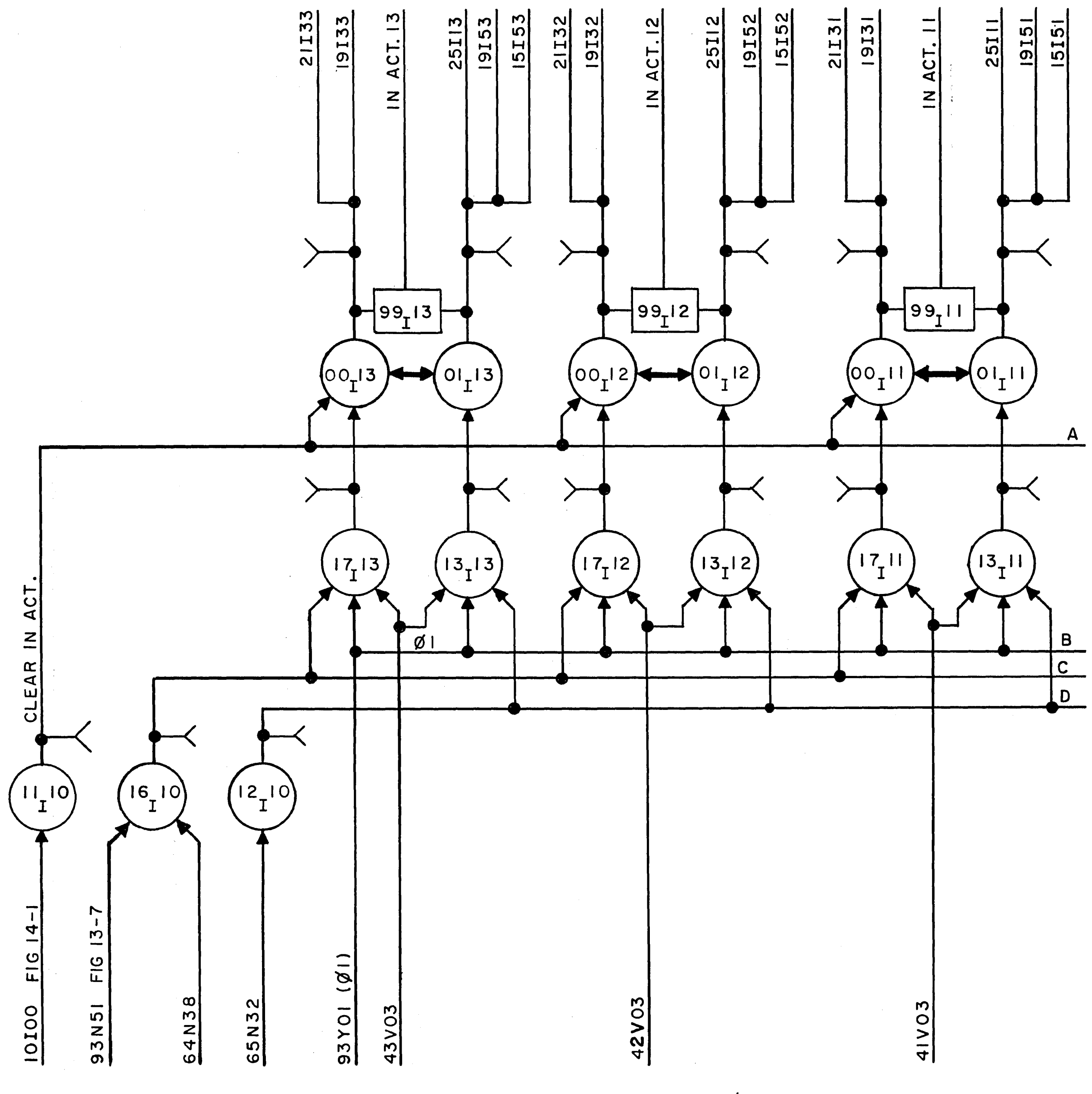
to set the designated flip-flop. The selection of the channel for the Input mode is then made.

It is a requirement of the system that for each word buffered into the computer there must be a separate request. To accomplish this, a provision is made to reset the input request gates following each request.

The command to reset the gate flip-flop (fig. 14-3) is generated by the I/O timing chain (fig. 13-7) during Disable I/O$_1$. This input is applied as a "0" input signal to the reset gate in figure 14-3. The channel designation (from the I/O translator as discussed in chapter 13) is also applied as a "0" input from the designated channel. The resulting "1" output from the reset gate at phase 1 resets the associated gate flip-flop.

INPUT DATA GATED AMPLIFIERS

Data entered into the computer from external equipment is gated in through input data gated amplifiers (fig. 14-4). Each channel has 30 input amplifiers associated with it so that parallel data transfer is realized. Accommodation for three bit positions (00 thru 02) for each of the

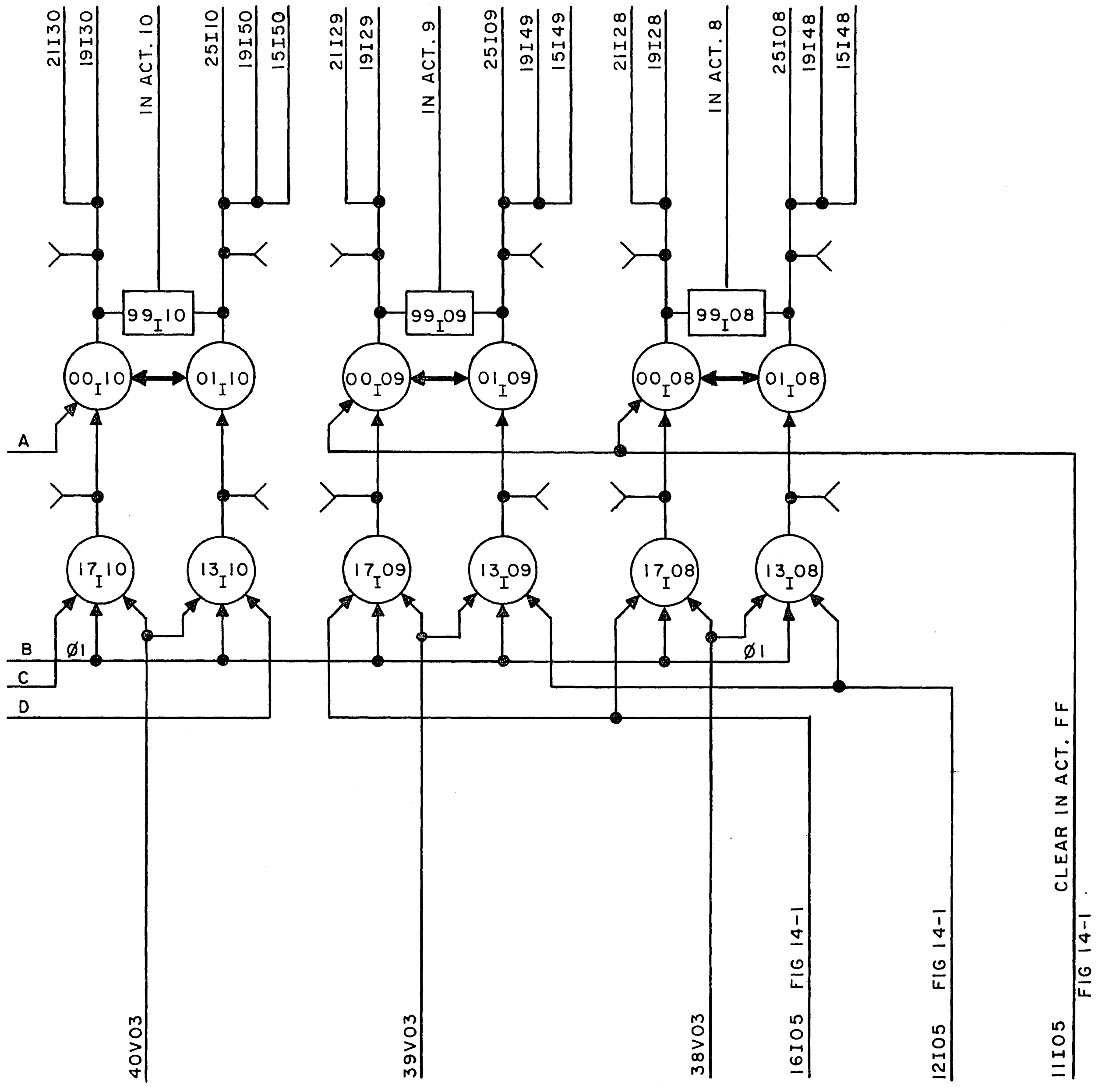


14 channels is illustrated in figure 14-4 (Inverters $^{20}C^{00}$ and $^{21}C^{00}$ are used for bit-position 00; $^{20}C^{01}$ and $^{21}C^{01}$ for bit position 01; and $^{20}C^{02}$ and $^{21}C^{02}$ for bit position 02.) The diagram for all other bit positions is not shown.

One word (30 data bits) is transmitted to the computer at any one time. The data is entered into the Z register before being stored in memory (see figure 8-1).

When the I/O section is in an Input mode, the data is placed on the lines by the external equipment prior to the sending of the Input request. This allows the data lines to stabilize before reading the data in (sampling) by the computer.

In order that the input data can exercise control over the output of the input amplifiers --Y--, (fig. 14-4), the gate signal from the I/O translator must be present. The manner by which

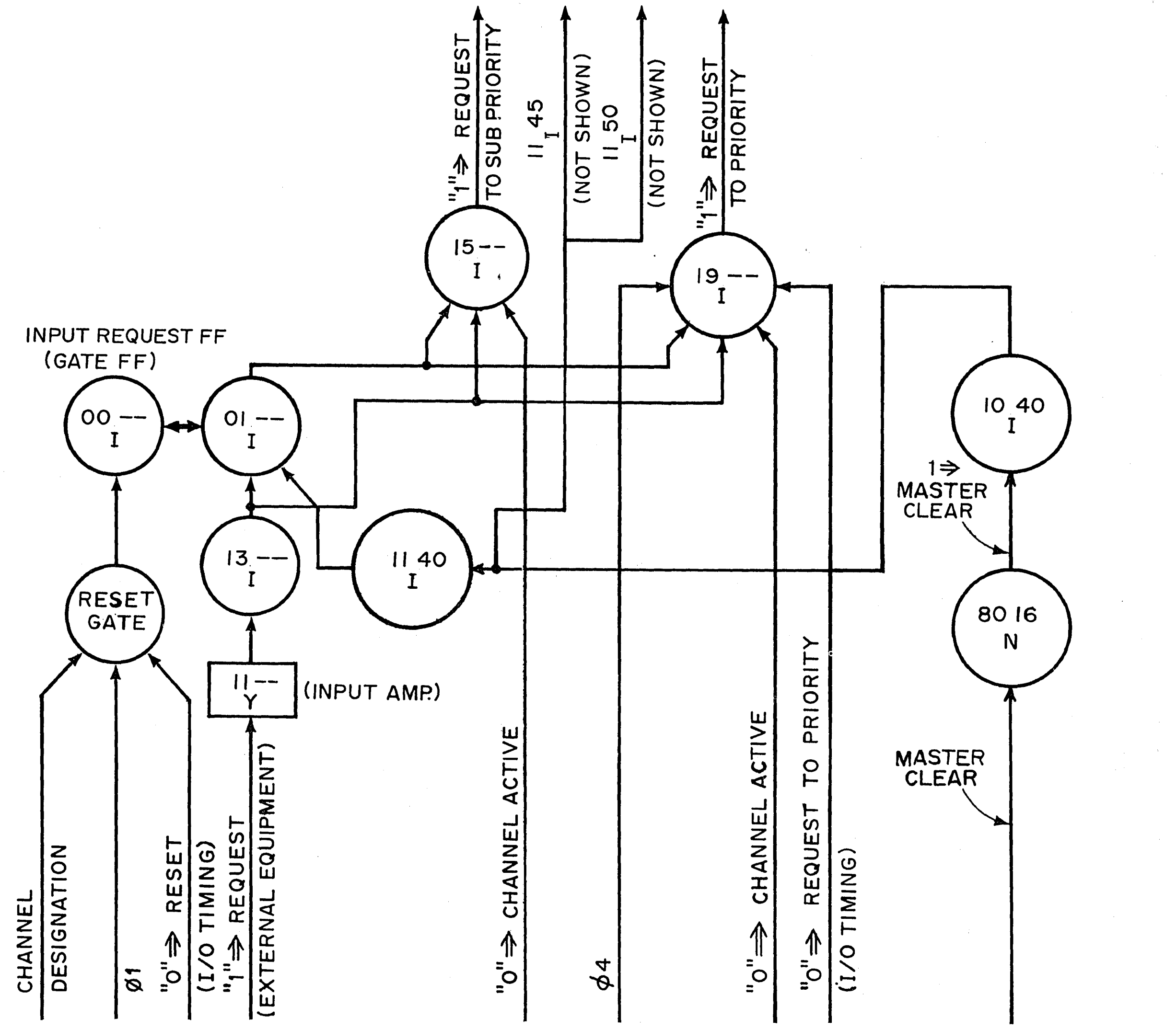


124.188

Figure 14-2.—Input active FF's 8-13.

this input is used is best illustrated using figure 14-5.

When the translator is not loaded with the active channel number, the output from the translator to the amplifier is a "1" that holds the input amplifier output at a "0". Regardless of the status of the data lines at this time, the output from the inverters (20C-- and the 21C-- in



124.189

Figure 14-3.—Input request gate.

figure 14-4) is a "1" that disables the Z register gate circuits (not shown). When the translator is loaded with the active channel number, the output from the translator to the input data gated amplifier is a "0". The data on the lines now determines the output from the amplifier. A "1" on the data line causes a "1" output from the amplifier forcing the output from a 20C-- or 21C-- circuit to a "0". This output is one of the enables required by a gate circuit in the Z-register. When the gate circuit is enabled, a "1" is loaded into Z. A "0" on the data line causes a "0" output from the amplifier. Since all other channels associated with the particular data line are inactive at this time, the output from all the amplifiers is "0". Now the 20C-- and 21C-- circuits have "1" outputs that disable the Z-register gates. Effectively, this operation loads a "0" into Z. Two inverters are used for any one bit position with each inverter having inputs from seven data channels and an output to a separate Z-register gate. This arrangement provides added assurance that input data will be properly transferred.

Data can be entered into the computer by main computer control utilizing only the input amplifiers. The instruction is programmed as follows:

$$17 \quad \hat{j} \quad \hat{k} \quad b \quad y$$

The instruction, f = 17 (see table 10-2), is used by an Interrupt subroutine to enter the Interrupt code into the computer for evaluation. During the A sequence, $\hat{j}$ is sent to the I/O translator to designate the channel which has the data available. The output from the translator provides the gate signal to the input amplifiers. The data lines are sampled and the data entered into the Z-register at phase 1. The command enable is operated in the D sequence.

INPUT ACKNOWLEDGE

Once the computer has sampled the data lines of an input channel, a control signal (Input Acknowledge signal) is sent to the external equipment. This control signal informs the external equipment that the data lines have been sampled. The control signal remains on the line for 14.8 microseconds. A "duration time" circuit is utilized for this purpose.

When the computer is master cleared the "1" output from $^{80}N^{20}$ (in the command enable circuits, not shown) forces the output from $^{10}I^{60}$ (fig. 14-6) to a "0". This "0" output enables $^{11}I^{60}$, $^{11}I^{64}$, $^{11}I^{67}$, and $^{11}I^{70}$ at phase 2 to clear the Input Acknowledge flip-flops I^{60} thru I^{73}. With the flip-flops cleared there is no signal on the input acknowledge line. If the associated manual clear indicator-pushbutton (IN ACK) is depressed, the output from $^{10}I^{60}$ is clamped at a "0". This enable signal will again be used to clear the 14 flip-flops.

At the same time that the input request gates are master cleared (as discussed earlier), a signal is applied to the duration timing circuit (not shown). This action ensures that the duration timing circuit is activated. The circuit will continue to operate for 14.8 μs before it returns to its stable state.

When the computer is in an Input mode and has sampled the data lines, the I/O translator has already been loaded with the designated channel number. The "0" output from the translator (see fig. 13-15) is one of the enables required by the ^{13}I-- circuit (fig. 14-6). The command to send the Input Acknowledge signal is generated by the I/O timing chain (fig. 13-7). During I/O_2 at time $T84_2$, the output from $^{96}N^{31}$ is a "1" forcing the output from the ^{12}I-- circuits (fig. 14-6) to a "0". At phase 1, the channel designated ^{13}I-- circuit is fully enabled to set the Input Acknowledge flip-flop. With this flip-flop set, the input to control line driver ^{10}Y-- is a "1". The Input Acknowledge signal is now on the line. At the same time, the "0" output from $^{12}I^{70}$ is used to enable the Signal Duration timing chain (not shown).

The Input Acknowledge signal is also used during the execution of an instruction (see table 10-2) programmed as follows:

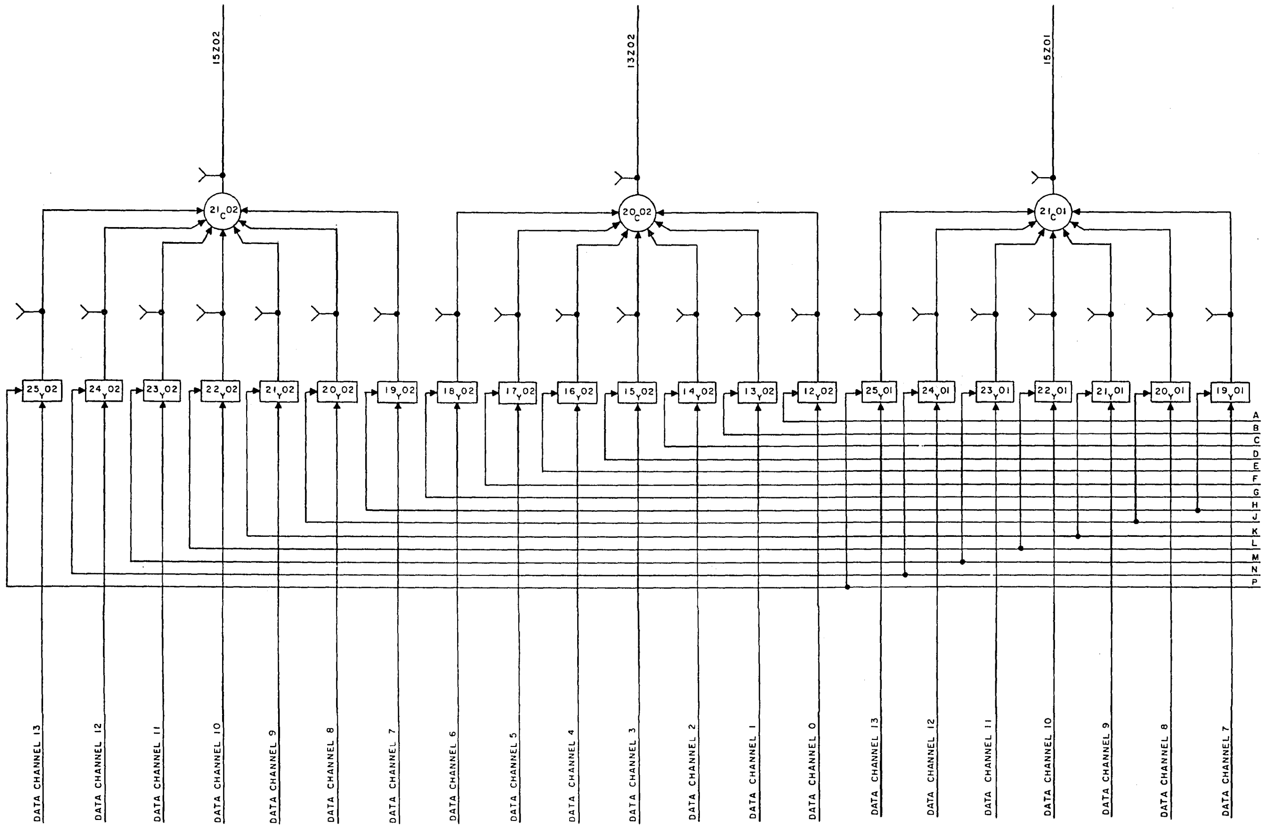
$$17 \quad \hat{j} \quad \hat{k} \quad b \quad y$$

Using this instruction the computer is to store information from the channel designated by $\hat{j}$. During the D sequence a Command signal is generated to send input acknowledge (f = 17). The "1" output produced in the D sequence is inverted to a "0" by the ^{12}I-- circuits (fig. 14-6) and applied to the ^{13}I-- circuits. The ^{13}I-- circuit to which the translator output is also applied is enabled at phase 1 to set the associated Input Acknowledge flip-flop. The Input Acknowledge signal is on the line and the same timing sequence takes place as for normal Input mode.

An instruction with f = 17 is intended for use in the computer's reply to an interrupt. Consequently, it is not synchronized with the Input Buffering process. Therefore, the execution of (n) sequential instructions, with f = 17 on the same channel, places (n) sequential Input Acknowledge signals on the Input Acknowledge line associated with that channel. It generates a signal which is (n) x 14.8 microseconds wide on that Input Acknowledge line. It must be understood then that the execution of a 17 instruction on a given channel while an input buffer is in progress can and does, it most cases, seriously interfere with the buffered transfer of data. The execution of any other instruction between two 17 instructions allows the Input Acknowledge line to return to the "0" state for a time consistent with I/O timing considerations before it must rise a second time.

INPUT MONITORS

The input monitors are used to generate an Internal interrupt at the completion of an Input buffer. (Completion means that all the words, as defined by the Input buffer, have been transmitted.)



When the computer is master cleared, a "1" input is applied to $^{10}I^{20}$ from the command enable circuit (not shown) via $^{80}N^{16}$. The "0" output of $^{10}I^{20}$ causes $^{11}I^{20}$ and $^{11}I^{25}$ (fig. 14-7) and $^{11}I^{30}$ (not shown) to provide "1" inputs to the zero sides of the Input Monitor flip-flops (I^{20} thru I^{27}). This input clears the flip-flops for channels 0 through 7. This action can also be accomplished by depressing the IN MON pushbutton (see figure 10-8).

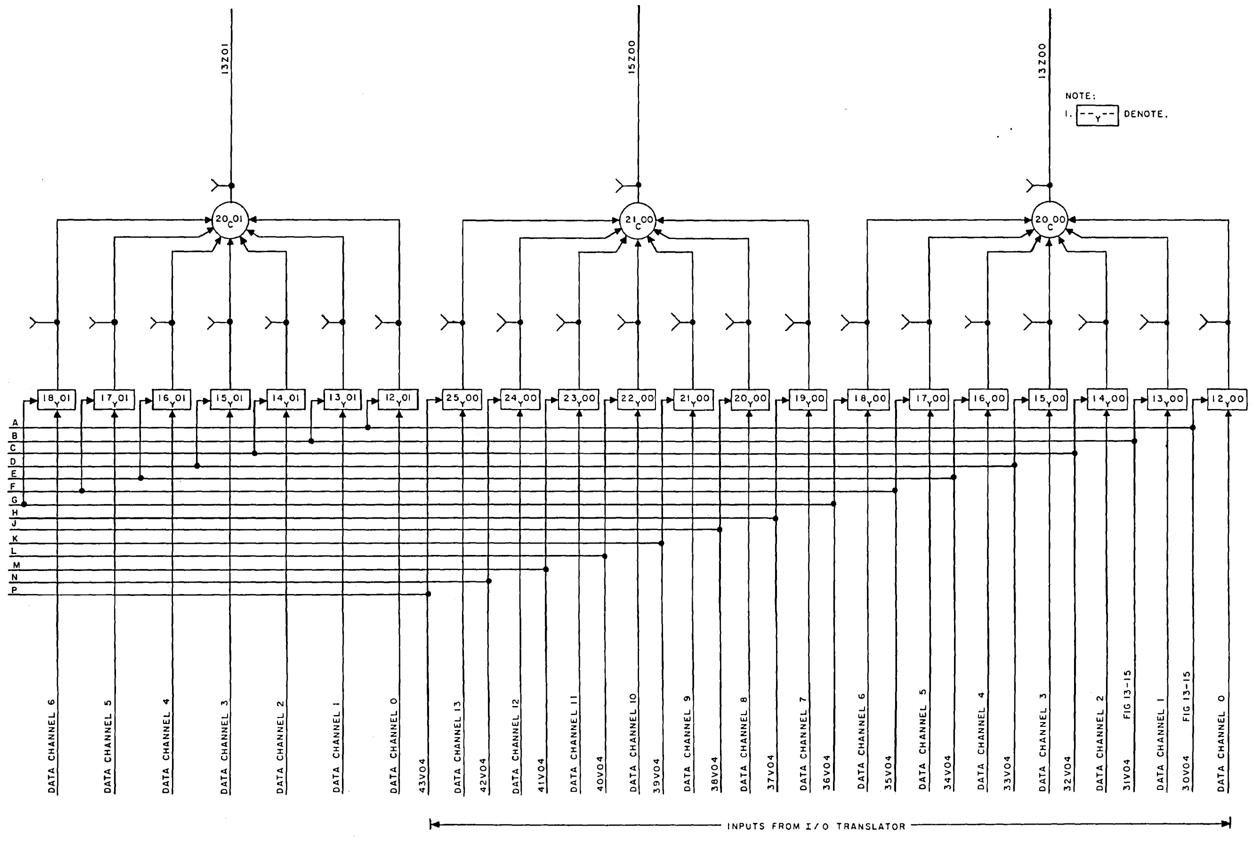
When any one of the 14 Input-Monitor flip-flops (only eight shown) is cleared, no interrupt is generated on that channel.

A particular Input Monitor flip-flop is initially set by the main program using an instruction as follows:

$$75 \quad \hat{j} \quad \hat{k} \quad b \quad y$$

During the A sequence, $\hat{j}$ is transmitted to the I/O translator to designate the specified channel. During the D sequence (not shown) a 1 output from an inverter ($^{65}N^{24}$, not shown) forces the output from ^{12}I-- to a "0". Thus, at phase 1 the designated ^{13}I-- circuit is enabled to set the particular Input Monitor flip-flop.

When the buffer is completed, a command from the I/O timing chain clears the Input Active flip-flop associated with the particular channel. The "0" output from ^{00}I-- (Input Active flip-flop figures 14-1 and 14-2) coincident with the "0" output from ^{01}I-- (Input Monitor flip-flop figure 14-7) enables a ^{19}I-- circuit. The "1" output from this circuit enters an Internal Interrupt In request into subpriority for processing. Now during Scan at time T83 in the I/O timing chain



124.190

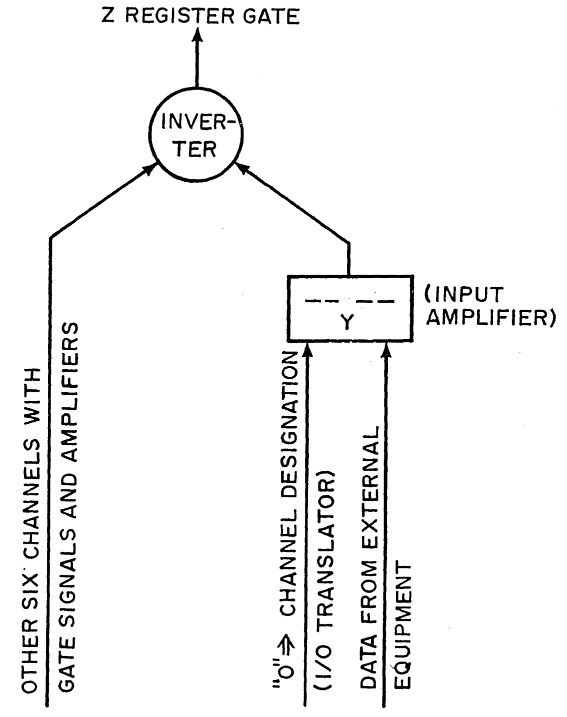
Figure 14-4.—Input data gated amplifier 00-02.

fig. 13-6), the command is generated to load priority. At this time $^{93}N^{24}$ is enabled and its "1" output forces the output from the ^{20}I-- circuits (fig. 14-7) to a "0". This enable, together with the two signals that enabled ^{19}I-- circuit, enables a ^{21}I-- circuit at phase 4. The "1" output from the selected circuit sets the associated Channel Designator flip-flop in priority (figures 13-12 and 13-13). The interrupt is now processed in the same manner as an external interrupt.

During the A sequence at time L^{61} (fig. 10-14), the command is generated to clear the circuit that initiated the interrupt (clear monitors or interrupt gates). This signal, together with the translation of the outputs from V^{24} and V^{25} in the storage register (fig. 13-16), enables $^{23}V^{24}$. The "1" output from $^{23}V^{24}$ forces the output from a ^{16}I-- circuit (fig. 14-7) to a "0", one of the enable signals required by a ^{17}I-- circuit to produce a clear signal input to the flip-flop.

It is also during the A sequence at time T13 that the Interrupt channel is loaded into the I/O translator (fig. 13-14). The translator output circuits (fig. 13-15) provide another enable to the designated ^{17}I-- circuit (fig. 14-7). At phase 1, this circuit is fully enabled to clear the Input Monitor flip-flop and remove the Interrupt signal.

It is possible to prevent the initial use of the monitor by programming; it is also possible


Z REGISTER GATE
INVERTER
Y
(INPUT AMPLIFIER)
OTHER SIX CHANNELS WITH GATE SIGNALS AND AMPLIFIERS
"0" → CHANNEL DESIGNATION (I/O TRANSLATOR)
DATA FROM EXTERNAL EQUIPMENT

124.191
Figure 14-5.—Data input circuit.

to prevent the monitor from generating an interrupt by programming. The monitor is cleared or kept cleared by the main program using either of the instructions as follows:

$73\ \hat{j}\ \hat{k}\ b\ y$ (See table 10-2.)

$66\ \hat{j}\ \hat{k}\ b\ y$

In both cases, $\hat{j}$ is loaded into the I/O translator during the A sequence to designate a specific channel. An instruction with f = 73 is used to set an input channel active (set the Input Active flip-flop) during the D sequence. At this time the "1" output from $^{65}N^{36}$ in the D sequence timing chain (not shown) forces the output from ^{16}I-- to a "0". At phase 1 the designated ^{17}I-- circuit is enabled to clear the Input Monitor flip-flop in case it had been previously set. Thus, an input monitor is not used on the designated channel.

An instruction with f = 66 is used to terminate an input buffer at any time before its completion, thus preventing the monitor from generating an interrupt. The enable for this instruction is produced during the C sequence (not shown). The input from $^{64}N^{37}$ in the C sequence is a "1" to the ^{16}I-- circuits. Since an enable used to clear the designated Input Active flip-flop is also generated at this time (see f = 66 instruction in table 10-2), it is conceivable that a false interrupt could be generated if the monitor were not cleared. At phase 1, the designated ^{17}I-- circuit (fig. 14-7) is enabled to clear the Input Monitor flip-flop and prevent the interrupt from being generated.

OUTPUT CIRCUITS

The circuits utilized by I/O to perform an output function consist of control circuits and the data-handling resisters. Before data can be transmitted from the computer to an external equipment, Output Active flip-flops (shown later) must be set by computer control. When the external equipment can accept data, it informs the computer by sending a control signal. The computer places the data on the lines and sends a control signal to inform the external equipment to sample the data lines. Under conditions as determined by main computer control and programming, an Internal Control signal can be generated to inform the computer when a buffer is completed.

The following discussion refers particularly to channels 2 through 13. All explanations concerning channels 0 and 1 are covered later.

OUTPUT ACTIVE FLIP-FLOPS

Each of the 14 output channels has an associated Output Active flip-flop (0^{00} thru 0^{13}) which must be set before data transmission to external equipment can be initiated. The flip-flops associated with channel 0 through 3 are shown in figure 14-8. These circuits are representative of all Output Active flip-flops.

When the computer is master cleared, the "1" output from $^{80}N^{18}$ (a command enable not shown) forces the output from $^{10}O^{00}$ to a "0". This "0" output is inverted to a "1" output by $^{11}O^{00}$ (fig. 14-8), and by $^{11}O^{05}$, and $^{11}O^{10}$ (not shown) to clear the Output Active flip-flops. Also, depressing the manual clear associated indicator-pushbutton (OUT ACT fig. 10-8) clamps the output from $^{10}O^{00}$ at a "0". This signal is inverted by the ^{11}O-- circuits to clear the

Output Active flip-flops in the same manner as just described.

The control of initially setting an output channel active is a function of the main computer program. The two instructions that may be used are in the form as follows:

$74 \quad \hat{j} \quad \hat{k} \quad b \quad y$ (See table 10-2.)

$76 \quad \hat{j} \quad \hat{k} \quad b \quad y$

With f = 74, an output channel is set active without a monitor. With f = 76, an output channel is set active with a monitor.

During the A sequence $\hat{j}$ is transmitted to the I/O translator (as described in chapter 13) which provides the Channel Designation signal. During the D sequence (circuits not shown) the output from an inverter $^{65}N^{37}$ (not shown) is a "1" forcing the output from ^{12}O-- to a "0". At phase 1 the designated ^{13}O-- circuit is enabled to set the Output Active flip-flop. When a flip-flop is set to indicate an active output channel enable signals are provided to certain circuits (explained later) in the I/O section.

When the external equipment sets its output request line, the request is entered into the computer by the Output Request gates. However, in order for the request to be processed completely by I/O, it must be entered into subpriority and priority. The "0" output from the ^{01}O-- side of the selected output active flip-flop is used to enable the ^{15}O-- circuits of the output request gates (fig. 14-9) to enter the request into subpriority. Also, it is used to enable the ^{19}I-- circuits of the request gates to enter the request into priority.

Another output from ^{01}O-- side of the output active flip-flop is used in a special application, which is a function of main computer control. The instruction is programmed as follows:

$63 \quad \hat{j} \quad \hat{k} \quad b \quad y$

During the A sequence, $\hat{j}$ is transmitted to the I/O translator providing the Channel Designation signal. The "0" output from the translator together with the "0" output from ^{01}O-- is utilized by a check circuit, (Test I/O Active not shown). In this application, an active output channel is being referenced to satisfy a jump condition, f = 63 (see table 10-2). The output from one of the inverters in the Test I/O active circuit is a "1" in accordance with the "0" inputs from the translator and the output Active flip-flop. If the designated channel is active, the output from the Test I/O active circuit is a "0" to a jump evaluation circuit (not shown) which satisfies the jump. If the designated channel is not active, the output from the Test I/O active circuit to the jump evaluation circuit is a "1" which does not satisfy the jump.

When the Output buffer has been completed, the Output Active flip-flop is cleared. During Disable I/O_1 at time $T88_1$ (fig. 13-7) the output $^{89}N^{51}$ is a "1". The "1" output from this circuit forces the output from $^{16}O^{02}$ (fig. 14-8) to "0" that indicates completion of the buffer. The translator at this time holds the number of the designated channel. At phase 1, a ^{17}O-- circuit is enabled by the "0" inputs from the translator and a ^{16}O-- circuit. The "1" output from ^{17}O-- thus clears the designated Output Active flip-flop indicating the completion of the output buffer.

The buffer can also be terminated before completion by the main program. The instruction is programmed as follows:

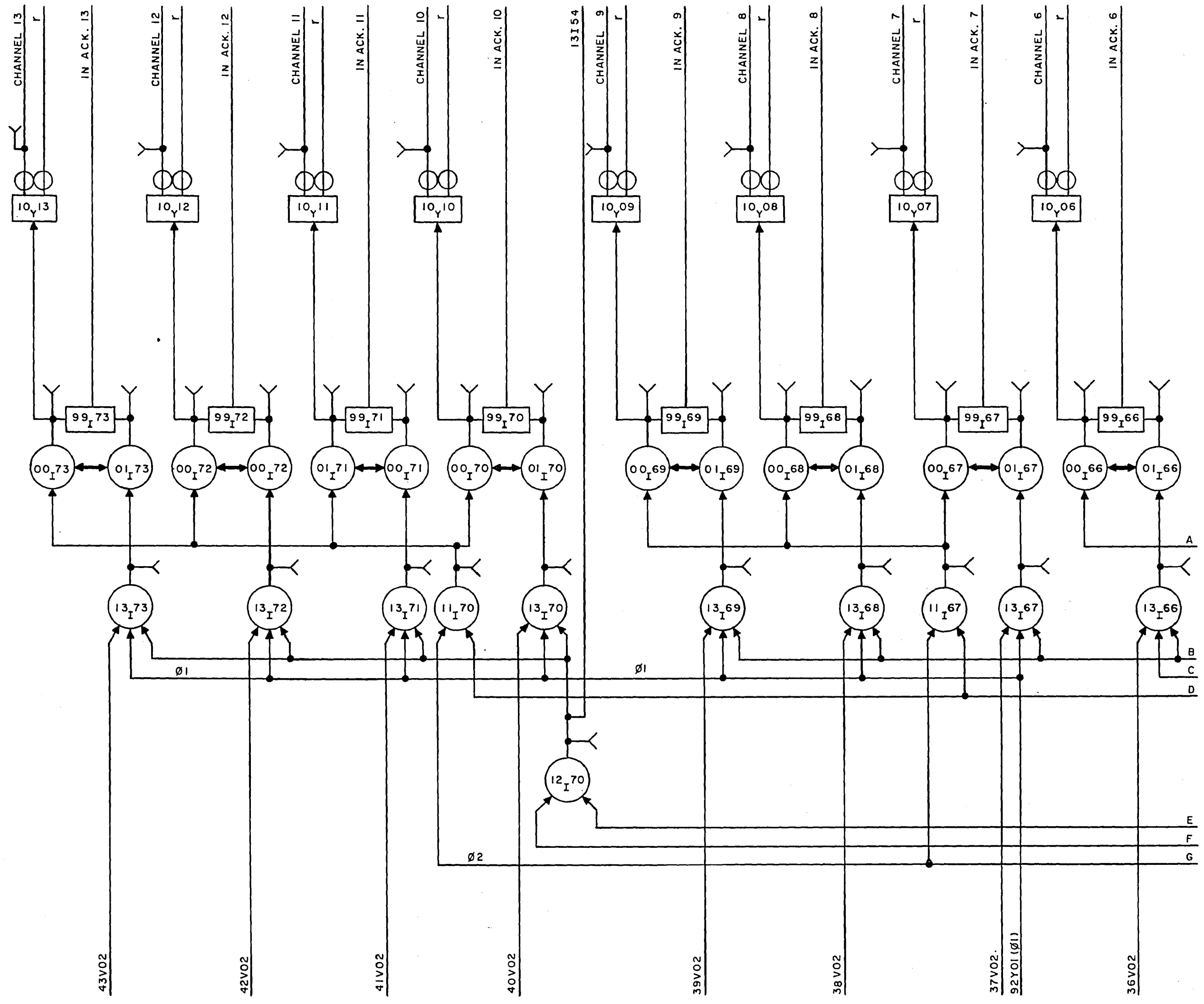
$67 \quad \hat{j} \quad \hat{k} \quad b \quad y$ (See table 10-2.)

The translator is loaded during the A sequence with $\hat{j}$. During the C sequence the output from an inverter $^{64}N^{36}$ (not shown) is a "1" forcing the output from ^{16}O-- (fig. 14-8) to a "0". At phase 1 the designated ^{17}O-- circuit is enabled to clear the Output Active flip-flop.

With the Output Active flip-flop cleared, enables are provided from ^{00}O-- side of the output active flip-flops to the output monitors shown later). If a given output Active flip-flop had been originally set using an instruction with f = 76, a provision was made to utilize the monitor at the completion of the buffer. The monitor generates an internal interrupt that is processed by I/O as any other Interrupt request. The "0" output from ^{00}O-- is an enable to the monitor to enter the request into subpriority. It is also an enable to enter the request into priority. No interrupt is generated when the Output Active flip-flops are cleared by an instruction with f = 67.

OUTPUT REQUEST GATES

When external equipment can accept data from the computer, it informs the computer by sending a control signal called an Output Data request. This request is entered into the computer by the output request gates (fig. 14-9)

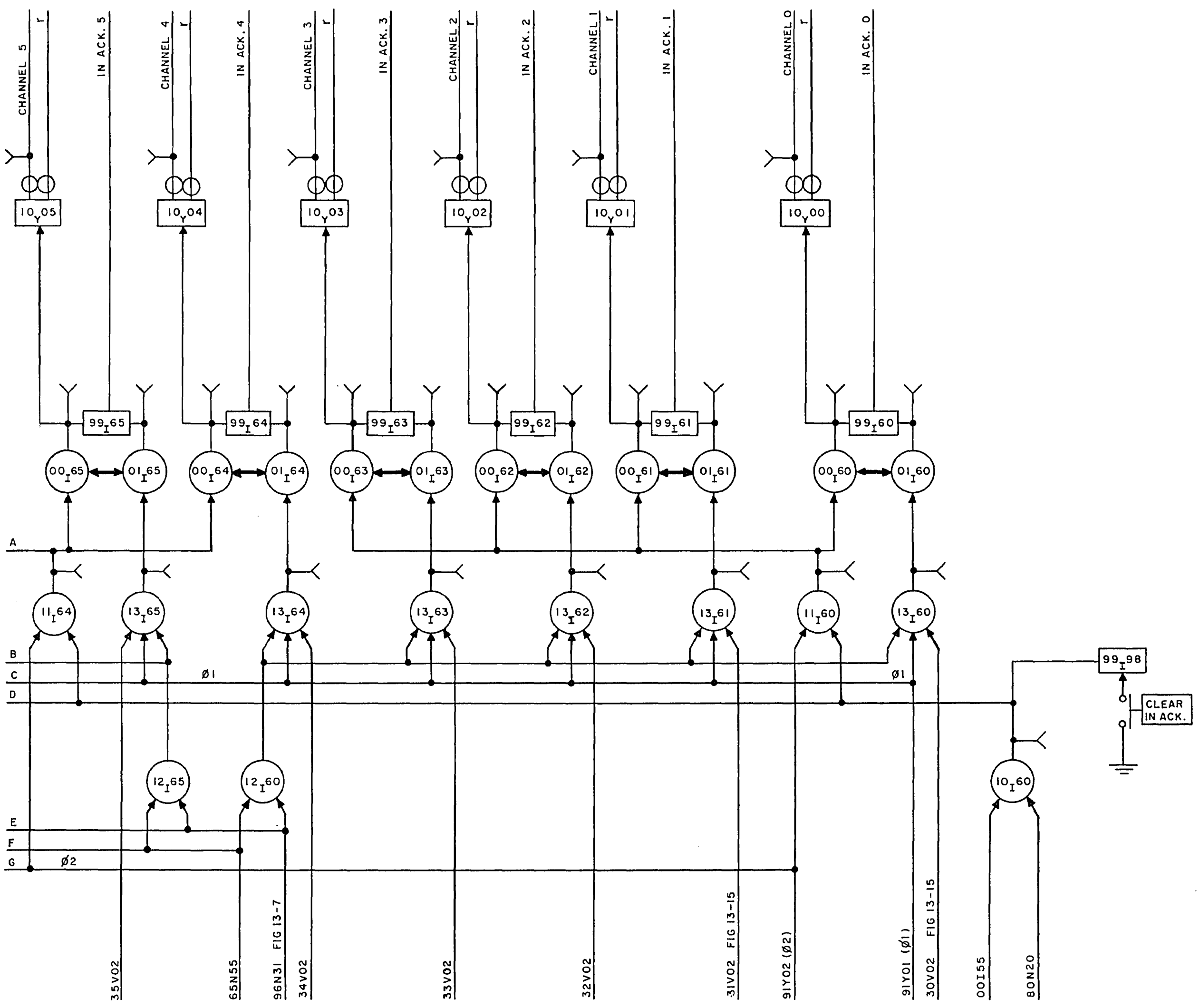


to be processed by I/O. Each channel (except channels 0 and 1) has an associated output request gate.

When the computer is master cleared, the "1" output from $^{80}N^{18}$ (a command enable circuit, not shown) forces the output from $^{10}O^{42}$ to "0". This output is inverted by $^{11}O^{42}$, $^{11}O^{47}$, and $^{11}O^{50}$ to a "1" output setting all the Output Request flip-flops in anticipation of a waiting request. Under normal operating conditions, prior to the completion of the previous output request, the designated flip-flop would have been reset or cleared. When the request line drops, the input to the ^{31}Y-- circuit becomes a "0". The output from the input amplifier is also a "0" that is inverted to a "1" output by the ^{13}O-- circuit. This "1" output sets the flip-flop associated with the particular channel.

When a request is transmitted to the computer, the signal on the line becomes a "1". The output from the amplifier is also a "1" forcing the output from the ^{13}O-- circuit to a "0". If the channel on which the request appears is active, its Output Active flip-flop is set. Now the associated ^{15}O-- circuit has all "0" inputs and transmits a "1" to subpriority to inform it of the Output Data request.

Since at any time there may be output requests on more than one channel, a signal must be transmitted to priority to determine the channel to be honored. This is the function of

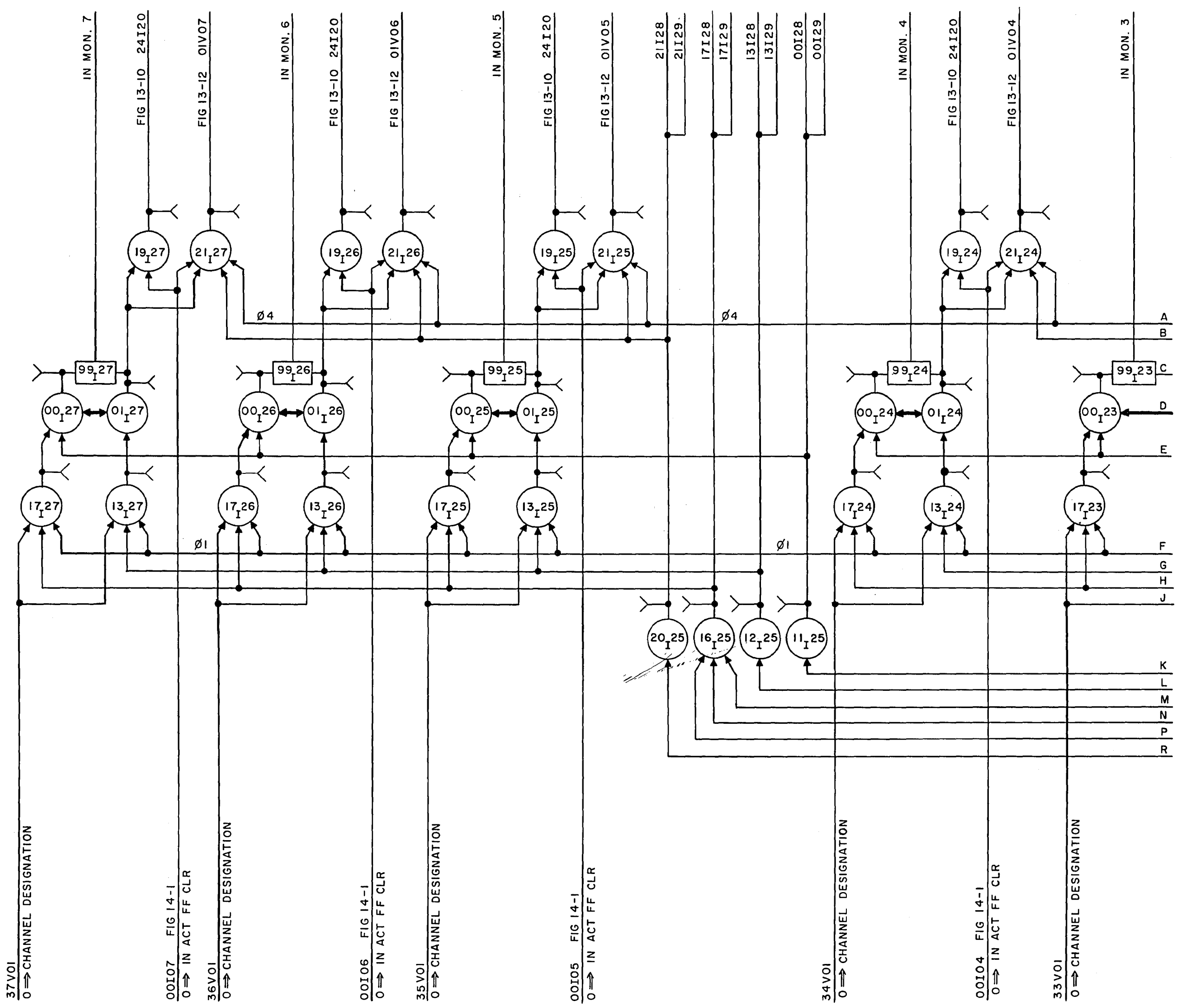


124.192

Figure 14-6.—Input acknowledge FF's.

the ^{19}O-- circuits. Three of the enable signals required by ^{19}O-- are the same three that enabled ^{15}O--. In addition, a signal from the I/O timing chain is required. During Scan at time T83 (fig. 13-6) the output from $^{97}N^{24}$ is a "1" forcing the output from ^{18}O-- to a "0". Then, at phase 4, the ^{19}O-- circuit is fully enabled and transmits a "1" signal to priority to set the designated flip-flop. The selection of the channel for the Output mode is made at this time.

Since it is a requirement of the system that for each word buffered out of the computer there be a separate request, a provision is made to reset the Output Request Gate circuit after each request. The command to reset the flip-flop is generated by the I/O timing chain during Disable I/O$_2$ at time T90$_2$. At this time the output from $^{93}N^{54}$ (fig. 13-7) is a "1" forcing the output from $^{16}O^{--}$ (fig. 14-9) to a "0". This "0" signal is one of the enables required by the Reset circuit, $^{17}O^{--}$. The other enable is the "0" output (from the translator) that designates the active channel and the channel to be reset. At phase 4 the designated $^{17}O^{--}$ circuit is enabled to reset the flip-flop.

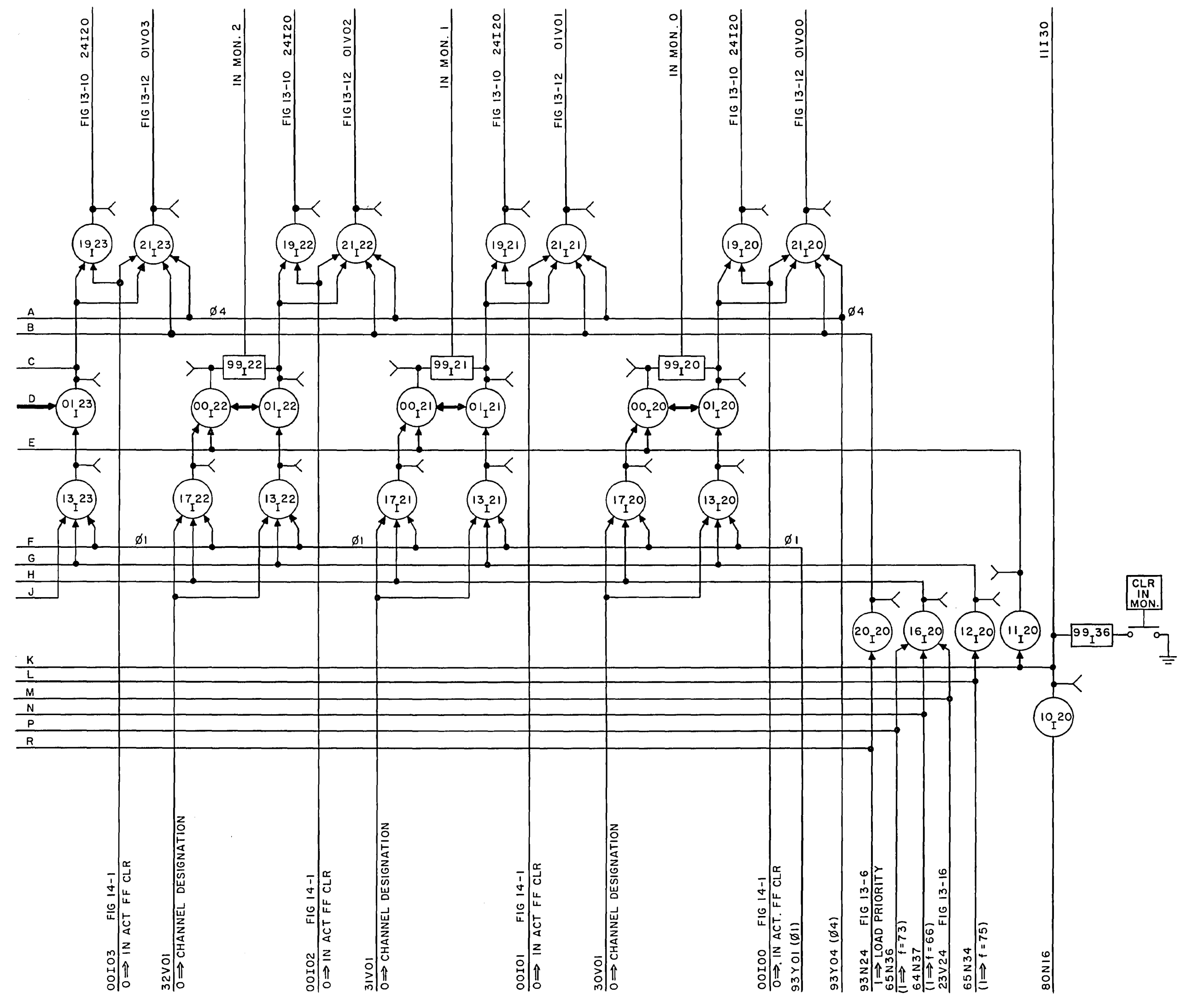

IN MON. 7
FIG 13-10 24I20
FIG 13-12 01V07
IN MON. 6
FIG 13-10 24I20
FIG 13-12 01V06
IN MON. 5
FIG 13-10 24I20
FIG 13-12 01V05
21I28
21I29
17I28
17I29
13I28
13I29
00I28
00I29
IN MON. 4
FIG 13-10 24I20
FIG 13-12 01V04
IN MON. 3
Ø4
Ø1
A
B
C
D
E
F
G
H
J
K
L
M
N
P
R
37V01 0⇑CHANNEL DESIGNATION
00I07 FIG 14-1 0⇑ IN ACT FF CLR
36V01 0⇑CHANNEL DESIGNATION
00I06 FIG 14-1 0⇑ IN ACT FF CLR
35V01 0⇑CHANNEL DESIGNATION
00I05 FIG 14-1 0⇑ IN ACT FF CLR
34V01 0⇑CHANNEL DESIGNATION
00I04 FIG 14-1 0⇑ IN ACT FF CLR
33V01 0⇑CHANNEL DESIGNATION

OUTPUT DATA REGISTER

The C_0 register handles the data transfer from the computer to external equipment. It consists of 30 stages to permit the parallel transfer of one 30 bit word for each output request. Figure 14-10 shows stages for bit positions 00-04, and are representative of all stages.

The output from the C_0 register is capable of driving the input amplifiers on 12 channels. The output from each 00C-- flip-flop drives three data line drivers. Each of these drivers is capable of driving four input amplifiers in external equipment. The data line drivers are utilized so that 32Y-- drives channels 2 through 5, 33Y-- drives channels 6 through 9 and 34Y-- drives channels 10 through 13.

When the computer is master cleared, the "1" output from 80N20 (from the command enable circuits, not shown) forces the output from 08C00 to a "0". This "00" output is inverted by 09C00 to a "1" output forcing the output from 10C00 (and 10C15 not shown) to a "0". This output is utilized at phase 4 to enable the 11C-- circuits to clear the register. When the associated manual clear indicator-pushbutton (C_0 see fig. 10-8) is depressed, the output from 08C00 is clamped at a "0". The same sequence then takes place as for master clear. Note that an indicator-pushbutton is associated with each bit position of the register.



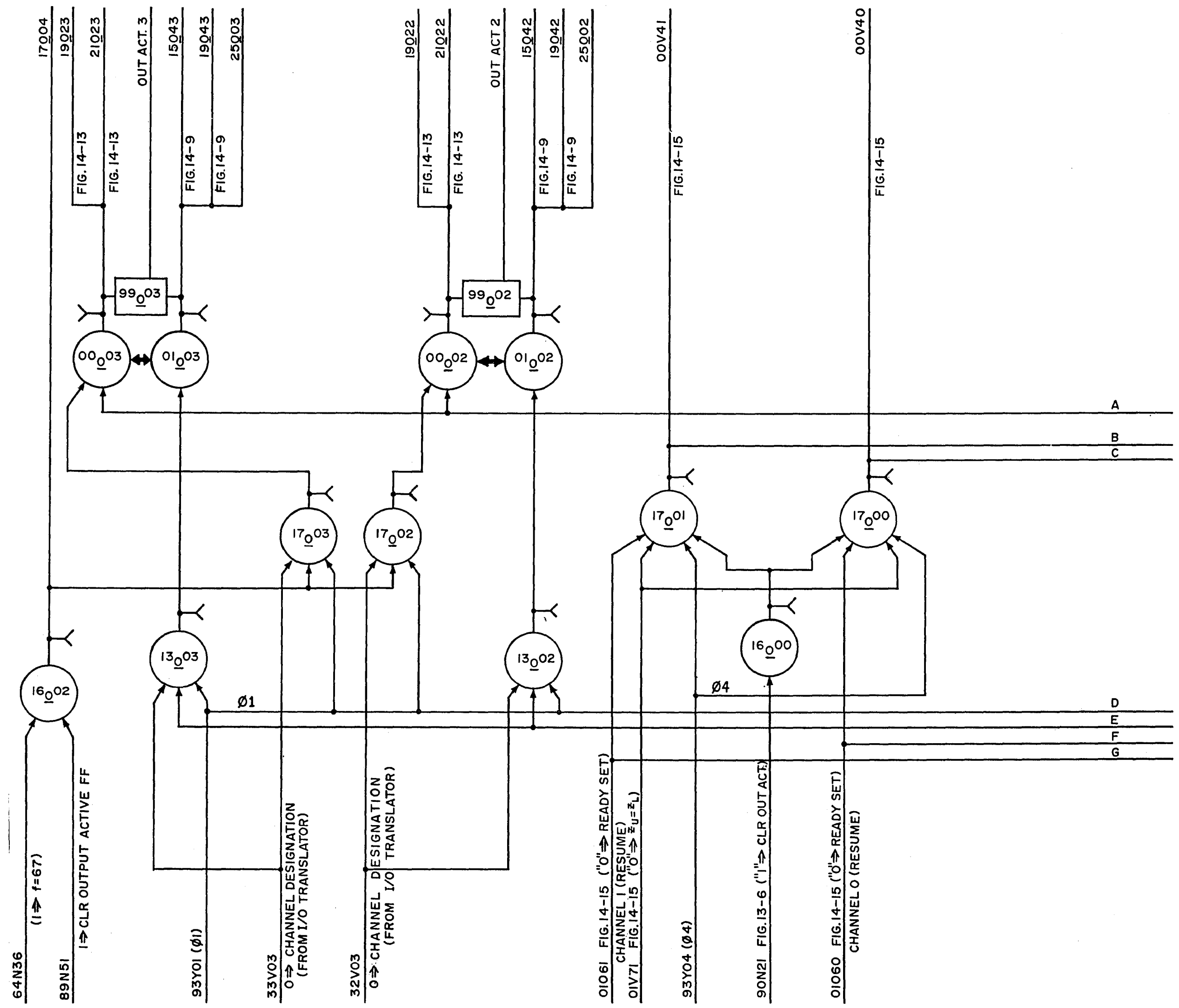
124.193

Figure 14-7.—Input monitor 0-7.

During a normal output buffer, the C_0 register is cleared and loaded by commands from the I/O timing chain. During I/O_2 at time $T83_2$, the output from $^{87}N^{24}$ (fig. 13-6) is a "1" forcing the output from $^{10}C^{00}$ (fig. 14-10) and $^{10}C^{15}$ (not shown) to a "0". This output is utilized at phase 4 to enable the ^{11}C-- circuits to clear the register.

During I/O_2 at time $T84_2$ (fig. 13-7) the command is generated to load the C_0 register (Z C_0). The output from $^{92}N^{31}$ and $^{93}N^{31}$ is a "1" forcing the output from the ^{12}C--circuits (fig. 14-10) to "0". This output is an enable to the ^{13}C-- circuits.

The data input to the C_0 register comes from corresponding bit positions in the Z-register.



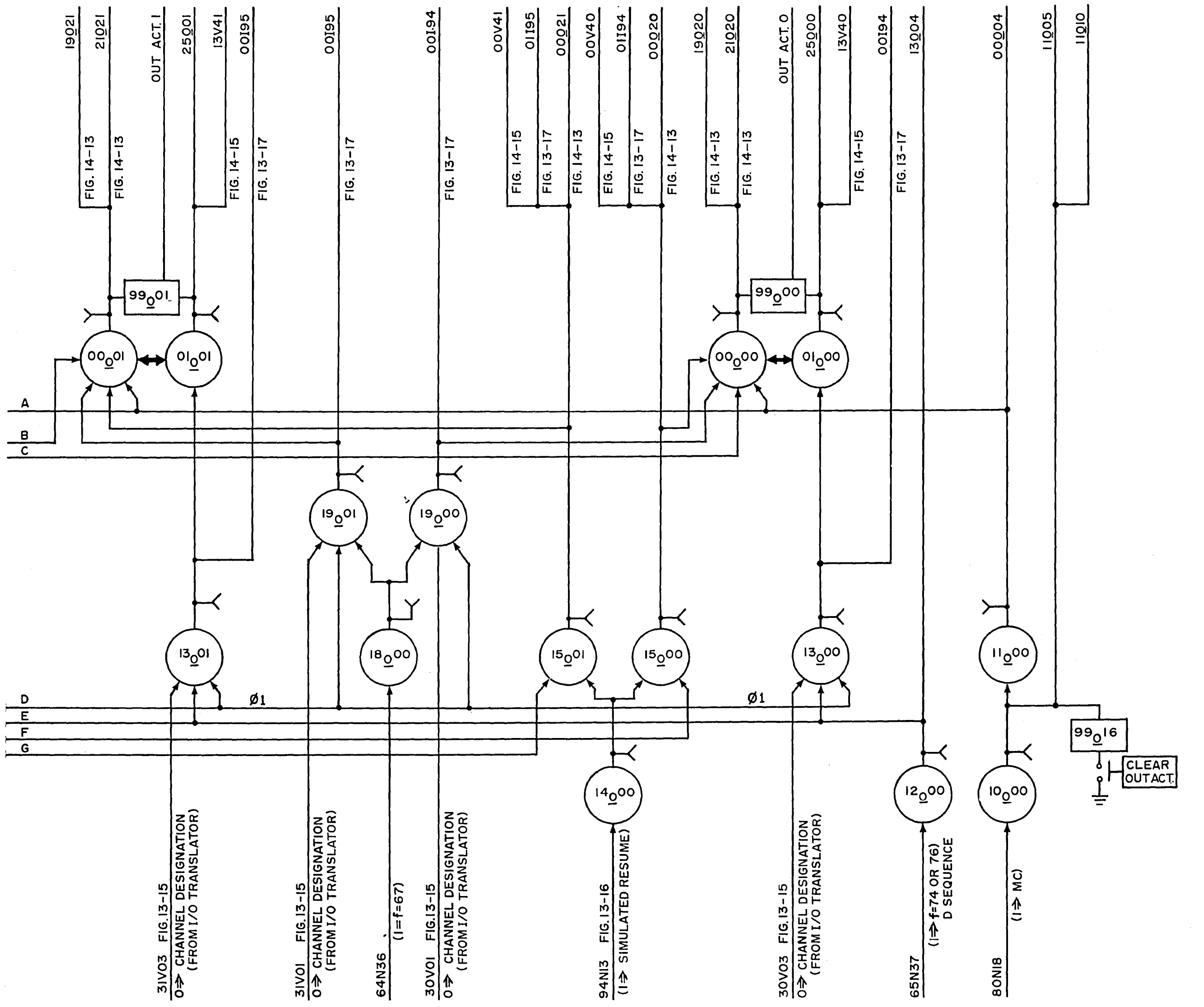
The circuits are such that if the input from Z is a "0", a "1" is to be loaded into C_0, and if the input from Z is a "1" a "0" is loaded into C_0.

Which of the ^{13}C-- circuits are enabled at phase 4 depends upon the data input from Z (having previously been read into Z from memory). When a ^{13}C-- circuit is enabled, the corresponding flip-flop is set; if a ^{13}C-- circuit remains disabled, the corresponding flip-flop remains cleared.

The output from the ^{00}C-- side of the flip-flop drives the data line drivers so that a "1" output appears as a "1" on the line and a "0" output appears as a "0" on the line. The information is placed on the data lines of all channels, but is only received by the equipment with its request line set. The information remains on the lines until the next time the C_0 register is used, either by another output buffer or by an f = 13 instruction.

The C_0 register is also utilized under control of the main program by an instruction as follows:

$$13 \quad \hat{j} \quad \hat{k} \quad b \quad y$$



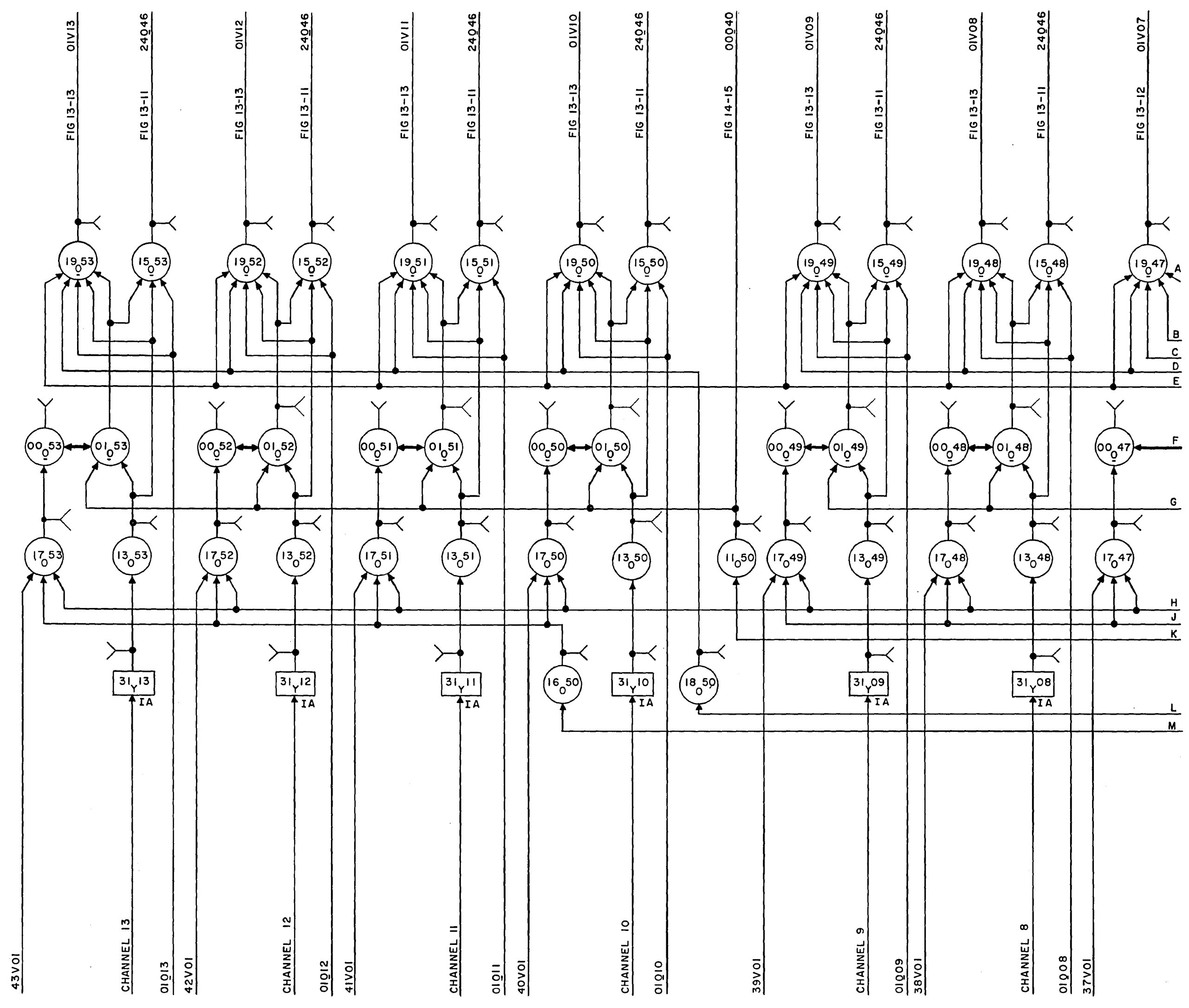
19021
21021
OUT ACT. 1
25001
13V41
00195
00194
00V41
01195
00021
00V40
01194
00020
19020
21020
OUT ACT. 0
25000
13V40
13004
00004
11005
11010
FIG. 14-13
FIG. 14-15
FIG. 13-17
99₀01
99₀00
00₀01
01₀01
00₀00
01₀00
19₀01
19₀00
13₀01
18₀00
15₀01
15₀00
13₀00
11₀00
14₀00
12₀00
10₀00
99₀16
CLEAR OUTACT.
A
B
C
D
E
F
G
Ø1
31V03 FIG.13-15 0→ CHANNEL DESIGNATION (FROM I/O TRANSLATOR)
31V01 FIG.13-15 0→ CHANNEL DESIGNATION (FROM I/O TRANSLATOR)
64N36 (1=f=67)
30V01 FIG.13-15 0→ CHANNEL DESIGNATION (FROM I/O TRANSLATOR)
94N13 FIG.13-16 (1→ SIMULATED RESUME)
30V03 FIG.13-15 0→ CHANNEL DESIGNATION (FROM I/O TRANSLATOR)
65N37 (1→f=74 OR 76) D SEQUENCE
80N18 (1→ MC)

124.194

Figure 14-8.—Output active FF's 0-3.

This instruction is used by the computer to send a function code to external equipment. The code is transmitted on the data lines of channels 2 through 13.

The enables required to clear and load the C_0 register are generated during the D sequence. At the proper time the output from an inverter $^{65}N^{54}$ (not shown) is a "1" that forces the output from $^{10}C00$ (fig. 14-10) and $^{10}C15$ (not shown) to a "0". This output is utilized at phase 4 to enable the 11C-- circuits to clear the register. During the following phase 4 times, the conditions are established to load the register as follows: The output from $^{65}N^{51}$ in the D sequence (not shown) is a "1" forcing the output from the ^{12}C-- circuits (fig. 14-10) to a "0". Thus, the register is loaded with data from the Z-register at phase 1 in the same manner as for normal output. The data is received by the external equipment on the channel on which the control signal is transmitted.

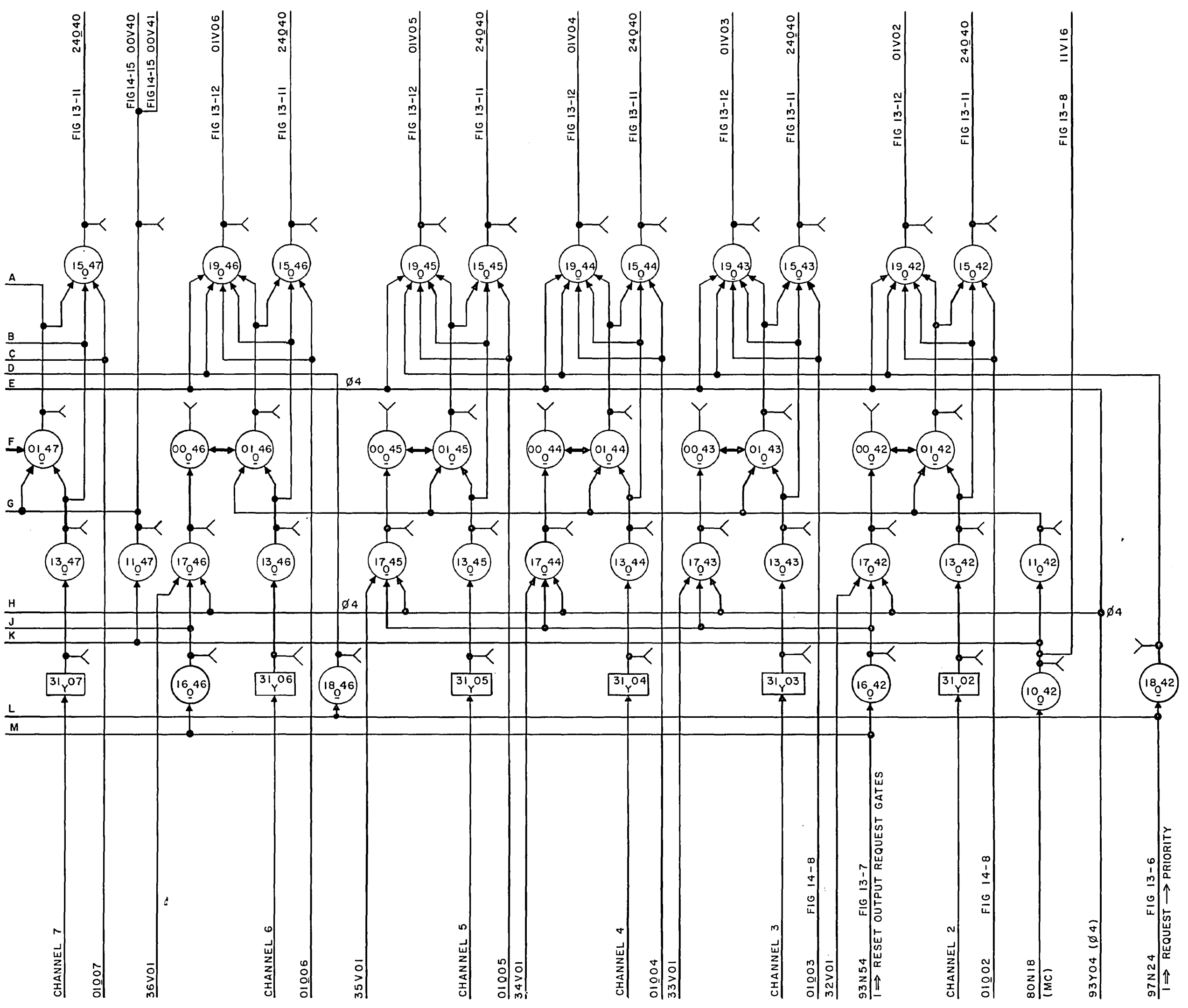


OUTPUT ACKNOWLEDGE AND EXTERNAL FUNCTION

After the computer has set the data lines, a control signal is sent to the external equipment 4.4 microseconds later. In the case of a normal output, the control signal informs the external equipment that the data lines are ready for sampling. In the case of an external function, the control signal informs the external equipment to sample the data lines. The flip-flop associated with each channel (fig. 14-11) performs the dual function of providing enable signals for both the Acknowledge and Function control outputs. These circuits are only associated with channels 2 through 13.

When the computer is master cleared, the input to $^{10}O^{62}$ (via an intercomputer logic circuit), is a "1" forcing the output from $^{10}O^{62}$ to a "0". At phase 4 the output is utilized to enable the ^{11}O-- circuits to clear the flip-flops. When the flip-flops are cleared, there is no signal on the output line.

Also, as the computer is master cleared, the "1" output from $^{80}N^{16}$ in the command enable



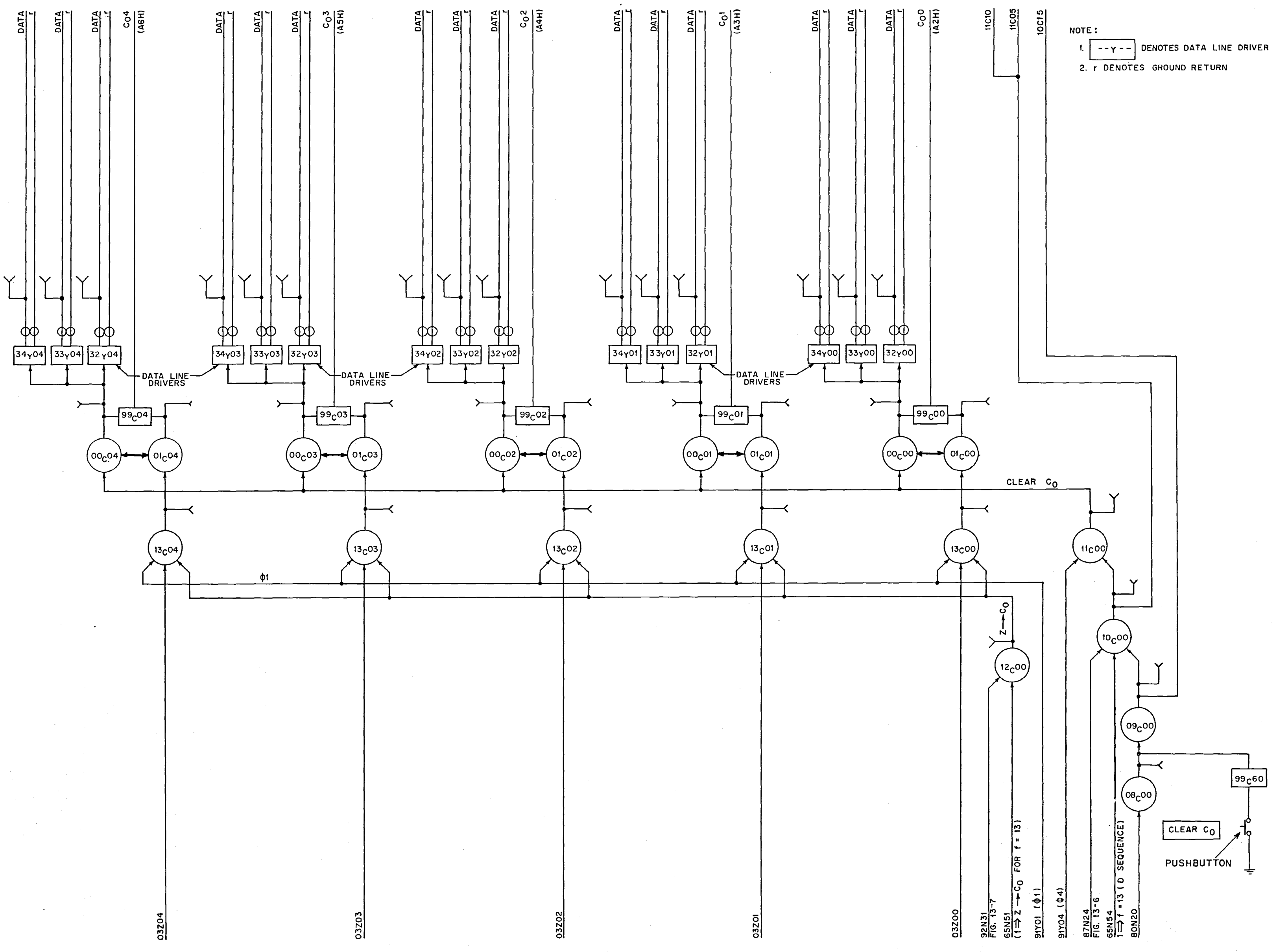
124.195

Figure 14-9.—Output request gates 2-13.

circuits, (not shown) sets Function Control flip-flop O^{83}, clearing the external function. If the manual clear button (OUT ACK (output acknowledge) see fig. 10-8) is depressed, the input to 10^{O62} (fig. 14-11) is again a "1". This produces a "0" output to 11^{O62}, 11^{O66}, and 11^{O70} that enables the clearing of the flip-flops at phase 4. When the associated manual clear button (EXTERNAL FUNCTION) is depressed, the output from 01^{O83} (fig. 14-11) is clamped at a "0" to clear the function. Each of the 12 flip-flops has an associated set indicator-pushbutton (OUT ACK). In this case the applicable indicator positions are 2 through 13. The Control flip-flop also has an associated set indicator-pushbutton (EXTERNAL FUNCTION).

When the computer is master cleared, the Output Acknowledge Duration Timing circuit (not shown) is set up in the same manner and by the same signal as the Input Acknowledge circuit (described earlier).

When the computer is in an Output mode (not intercomputer) and has set the data lines, the I/O translator (fig. 13-15) holds the designated



124.196

Figure 14-10.—C_0 register (00-04 bit positions).

channel number. The "0" output from the translator is one of the enables required by the ^{13}O-- circuit (fig. 14-11) to set the associated flip-flop.

The command to send the Output Acknowledge signal (an indication that data has been placed on the output lines) is initiated by the I/O timing chain (fig. 13-7). During Disable I/O$_2$ at time T90$_2$ the output from ^{94}N^{54} is a "1" forcing the output from the ^{12}O-- circuits to a "0". At phase 4, the channel designated ^{13}O-- circuit is enabled to set the flip-flop.

The "0" output from the ^{01}O-- side of the flip-flops is one of the enables required by the ^{19}O-- circuit. The other enable must come from Control flip-flop O^{83}. When ^{94}N^{54} is enabled, its "1" output sets O^{83}. The "1" output from ^{00}O^{83} now forces the output from the ^{18}O-- circuits to a "0" which is applied to the ^{19}O-- circuits. The designated ^{19}O-- circuit is enabled and the "1" input to Control Line driver ^{30}Y-- places a "1" on the line. The Acknowledge signal is thus set.

Since the signal must remain set for a period of 14.4 microseconds, a duration timing circuit is utilized. At the same time that the flip-flop ^{00}O^{83} is set, the "0" output from ^{12}O^{66} is used to start the duration timing chain (not shown).

At the proper time, a "1" signal is applied to ^{10}O^{62} (fig. 14-11) from the duration timing circuit (via an inverter ^{00}O^{55}, not shown). The "0" output from ^{10}O^{62} is applied to the ^{11}O-- circuits, causing them to be enabled during phase 4. The "1" outputs from the ^{11}O-- circuits clear all flip-flops, thereby removing the output acknowledge signal from the lines.

As aforesaid, the flip-flops that enable the Acknowledge signal are also utilized to send the Function Control signal. Since the external function comes under the control of the main program, the instruction is programmed as follows:

$13\quad \hat{j}\quad \hat{k}\quad b\quad y$ (See table 10-2.)

By the use of this instruction, the computer sends a function code to external equipment. The channel on which the code is sent is designated by $\hat{j}$.

During a normal output mode, the time between the setting of the data lines and the sending of the Acknowledge signal is 4.4 microseconds. However, this delay is strictly a function of the I/O timing chain.

During the execution of an f = 13 instruction, the I/O timing chain is not used and this necessitates the use of a special time delay circuit (fig. 14-12).

The normal static condition for the delay network is all flip-flops (L47 thru L^{50}) cleared. The External Function code is set on the data lines. When an inverter ^{59}T^{33} (in the D sequence not shown) is enabled, the "1" output from this circuit sets L^{47} of the delay network. The circuit operation is such that 4.4 microseconds later, L^{50} is set. The "1" output from 00L^{50} clears control flip-flop O^{83} (fig. 14-11) and forces the output from the ^{12}O-- circuits to a "0". The ^{13}O-- circuit designated by the translator output is now enabled at phase 4 to set its associated flip-flop. The "1" output from ^{01}O^{83}, meanwhile, forces the the output from the ^{20}O-- circuits to a "0". The designated ^{21}O-- circuit is enabled. The "1" input to the Control Line driver, ^{41}Y--, places a "1" on the line. The Function Control signal is now set.

Since this signal must also remain on the line for 14.4 microseconds, the Duration Timing sequence is utilized in the same manner as for a normal Output Acknowledge. An external function requires no return control signal from the external equipment.

OUTPUT MONITORS

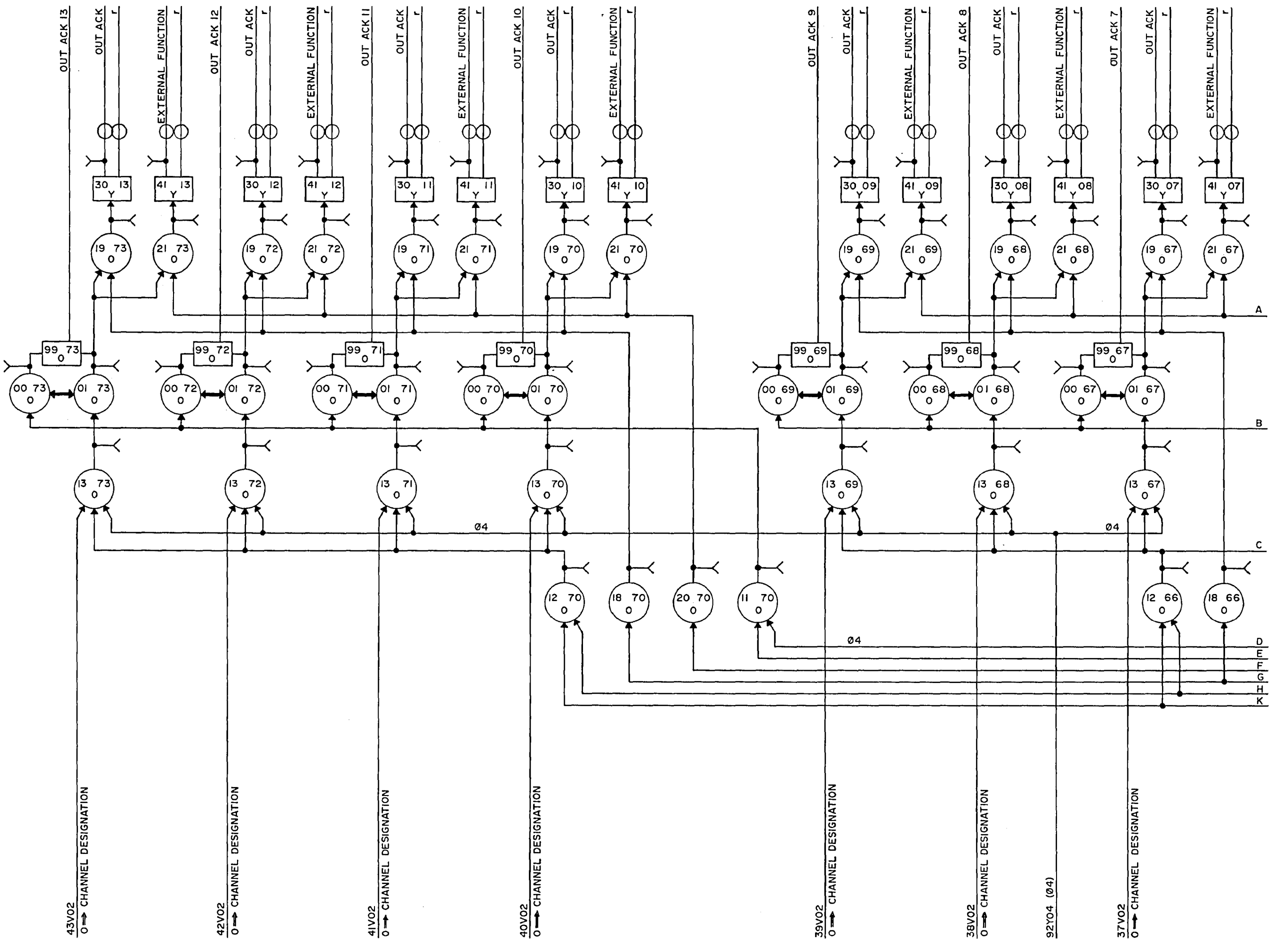
The output monitors (fig. 14-13) are used to generate an internal interrupt at the completion of an output buffer. This means that all the words defined by the Output buffer have been transmitted. Only five of the 14 Output Monitor flip-flops (O^{20} thru O^{24}) are shown.

When the computer is master cleared, the "1" output from ^{80}N^{20} (a command enable, not shown) forces the output from ^{10}O^{20} to a "0". This "0" output is inverted to a "1" output by the ^{11}O-- circuits to clear the monitor flip-flops. The flip-flops can also be cleared by the associated manual clear button (OUT MON, see fig. 10-9) which, when depressed clamps the output from ^{10}O^{20} to "0". No interrupt can be generated when the flip-flops are cleared.

The monitor is initially set by the main computer program using an instruction as follows:

$76\quad \hat{j}\quad \hat{k}\quad b\quad y$ (See table 10-2.)

During the A sequence, $\hat{j}$ is loaded into the translator. During the D sequence an inverter, ^{65}N^{33} (not shown) produces a "1" input to ^{12}O^{20} which forces the output from this circuit

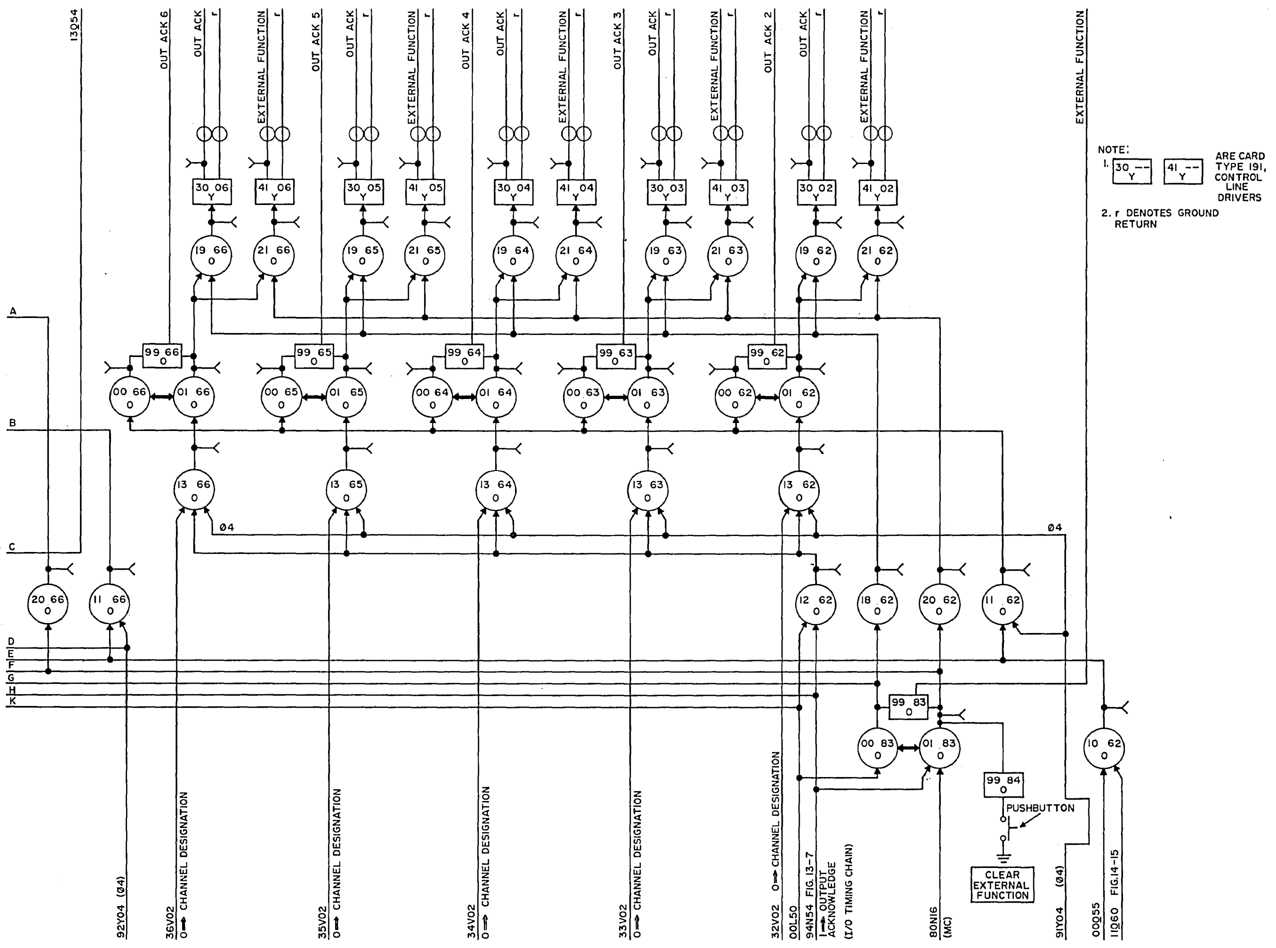


to a "0". This signal serves as an enable to the 13O-- circuits. At phase 1, the designated 13O-- circuit is enabled to set the Output Monitor flip-flop.

When the buffer is completed, a command from the I/O timing chain clears the Output Active flip-flop associated with the particular channel. The "0" output from the 01O-- (monitor flip-flop) coincident with the "0" output from 00O-- (output active flip-flop) enables a 19O-- circuit. The "1" output from 19O-- enters an Internal Interrupt Out request into subpriority for processing.

Now during Scan at time T83 (fig. 13-6) the command is generated to load priority. At this time the output from 96N24 is a "1" forcing the output from the 20O-- circuits to "0". This enable, together with the two signals that enabled the 19O-- circuit, enables a 21O-- circuit at phase 4. The "1" output from this circuit sets the associated channel flip-flop in priority. The interrupt is now processed in the same manner as an external interrupt.

At time L61 (fig. 10-14) in the A sequence, the command is initiated to clear the circuit that generated the interrupt. This signal, together with the translation of the outputs from V24 and V25 in the storage register (fig. 13-16), enables 24V24. The "1" output from 24V24 forces the output from 16O-- (fig. 14-13) to a "0", one of the enable signals required by a 17O-- circuit. It was also during the A sequence that the designated interrupt channel had been loaded into the translator. At phase 1 the designated 17O-- circuit is enabled to clear the Output Monitor flip-flop and remove the Interrupt signal.



124.197

Figure 14-11.—Output acknowledge FF and external function.

It is possible to prevent the initial use of the monitor by programming. It is also possible to prevent the monitor from generating an interrupt by programming. The monitor is cleared or kept cleared by the main program using either of the instructions as follows:

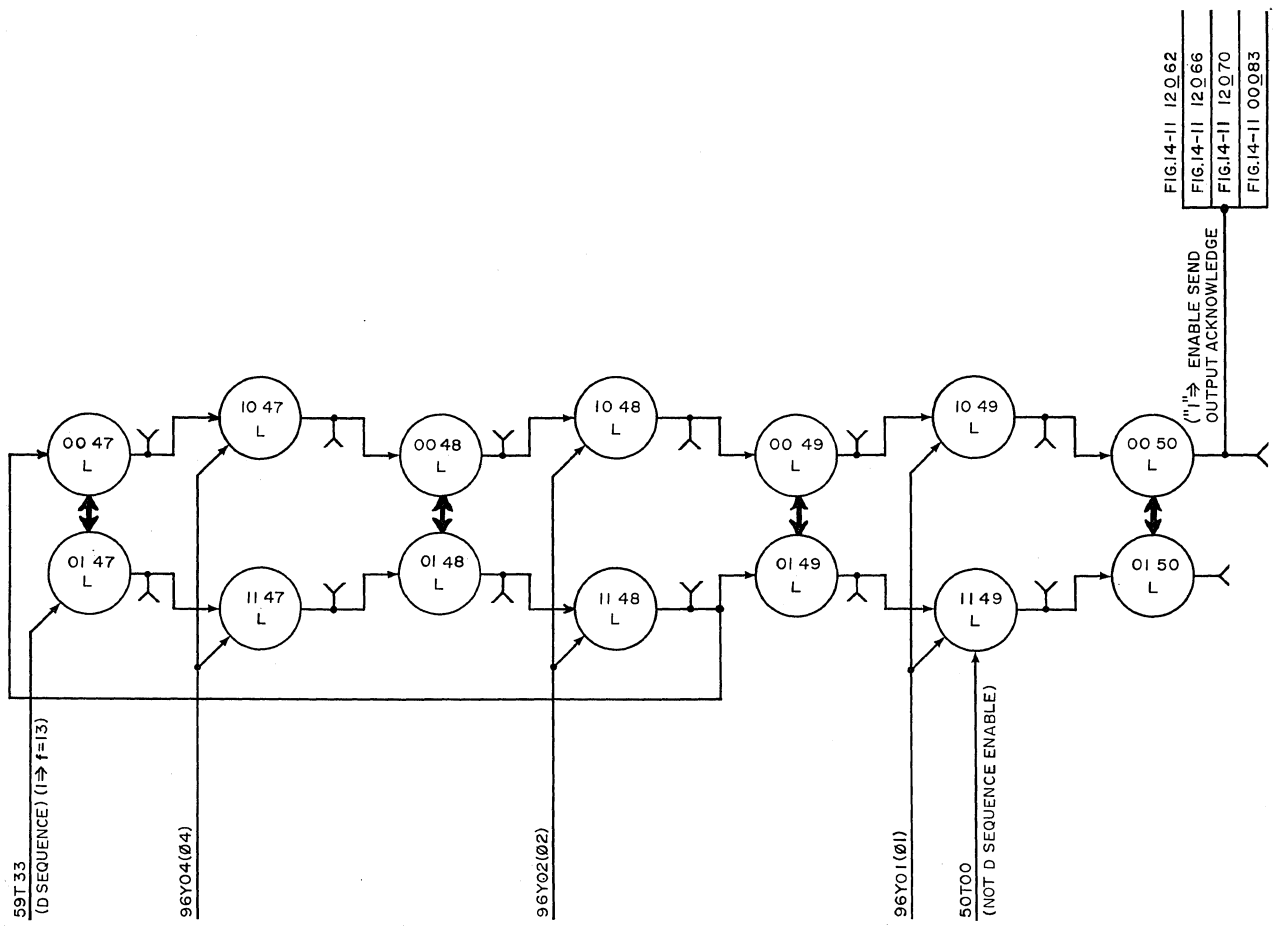
$$74 \quad \hat{j} \quad \hat{k} \quad b \quad y \qquad \text{(See table 10-2.)}$$

or

$$67 \quad \hat{j} \quad \hat{k} \quad b \quad y$$

In both instructions, $\hat{j}$ is loaded into the I/O translator during the A sequence to designate a specific channel. An instruction with f = 74 is used to set an output channel active (set the Output Active flip-flop) during the D sequence. At this time the "1" output from $^{65}N^{35}$ in the D sequence (not shown) forces the output from ^{16}O-- to a "0" that is an enable to the ^{17}O-- circuits. At phase 1, the designated ^{17}O-- circuit is enabled to clear the Monitor flip-flop in case it had been previously set.

An instruction with f = 67 is used to terminate an output buffer at any time before its completion. The enable for this instruction is initiated during the C sequence with the output from $^{64}N^{35}$ being a "1". Since an enable at this time is also used to clear the designated Output Active flip-flop, a false interrupt could be generated if the monitor were not cleared. At phase 1 the designated ^{17}O-- circuit is enabled to clear the Monitor flip-flop to prevent the generation of an interrupt.



124.198

Figure 14-12.—Delay external function control.

INTERCOMPUTER COMMUNICATIONS

The transmission of data and control signals between computers presents problems not encountered in the normal transmissions between a computer and other external equipment. These problems necessitate the use of special circuits for operations concerned with channels 0 and 1. Also, in intercomputer communications, new terminology is involved with regard to the control signals. In previous topics, the Acknowledge signal was always sent from the computer to the external equipment; the Request signal was always sent from the external equipment to the computer. Now it is the Ready signal that is sent to the external computer and the Resume signal that is sent from the external computer.

Since computers can process data at the same rate, channels 0 and 1 and an Out-on 2 request have the lowest priority in I/O. This is necessarily true because it is conceivable that inter-computer data transfer could lock out the other external equipment, if channels 0 and 1 were assigned a high priority. Likewise, in order to prevent one external computer from locking out the other, data transmission is carried out on a time-share basis. In this way, data transmission is alternated automatically between each of the two external computers.

A timing problem occurs again when the computer receiving the data does not send the Resume signal because of some malfunction. This would cause the Ready signal to remain on the line and prevent the transmission of data to the other computer. To prevent this, a time limit is imposed that allows the Ready signal to remain on the line for a maximum period of 32 to 64 seconds. The time element involved may seem extremely long in comparison to computer time; but the external computer uses the same priority logic with regard to input channels, since it may also communicate with other external equipment.

I/O CONTROL

The e-designator (figures 13-6 and 13-7) and the t-designator (figure 13-8) generate the enable signals required to buffer data on channels 0 and 1. In most cases, enables are also required from the circuits that are peculiar to both those channels.

The utilization of the t-designator differs in one respect from the normal usage as explained. The Overflow flip-flop (V^{65}) normally indicates, when set, that the Real-Time Clock count has exceeded the capacity of the lower-half of address 00036. When used in conjunction with an output buffer on channel 0 or 1, V^{65} provides enables to limit the maximum amount of time that the Ready signal is set on either of the two channels.

REAL-TIME CLOCK

The function of the Real-Time Clock is to provide the computer with the means to reference real time (actual time). The oscillator furnishes output pulses at a rate of 4096 cps. This output is applied to a logical countdown circuit which furnishes pulses to subpriority at a rate of 1024 cps. In time, this means that when the 2^{10} position of the Z-register holds a "1", one second has elapsed. The Real-Time Clock provides a particularly useful function in the computer by measuring and limiting the amount of time it takes to buffer out a word on channel 0 or 1.

When power is applied to the computer, there is always an output from the Real-Time Clock oscillator. Whether the output is utilized or not depends on the position of the DISCONNECT RTC control (fig. 10-9). If the DISCONNECT RTC is selected, the Real-Time Clock request circuit in subpriority is disabled. When this circuit is disabled, clock address 00036 cannot be updated (advanced).

INPUT

When the computer is to receive data from an external computer, the Input Data request is processed in the normal manner according to priority considerations. The explanations for an input request on channel 0 or 1 is the same as that already described.

In order that an external computer can check for an active input channel, the Input Active flip-flops (fig. 14-1) on channels 0 and 1 (I^{00} and I^{01}), have associated control line drivers ($^{43}Y00$ and $^{43}Y01$). When the channel is active, the output from the ^{00}I-- side of the associated flip-flop is a "1" which is the input to the line driver. The output from the ^{43}Y-- circuit is a "1", which is an indication of an active channel to the external computer. This output from the ^{43}Y-- circuit is one of the inputs to a gated amplifier in the external computer.

The external computer checks for the active channel by using an f = 13 instruction and $\hat{j} = 0$ or $\hat{j} = 1$ (see table 10-2). It is the output from the translator that is the enable to the gated amplifier. If the interrogated channel is active as indicated by a "1" output from the amplifier, the external computer can set up an output buffer.

The action of all other input circuits, as explained, is the same for an input mode on channels 0 and 1.

OUTPUT

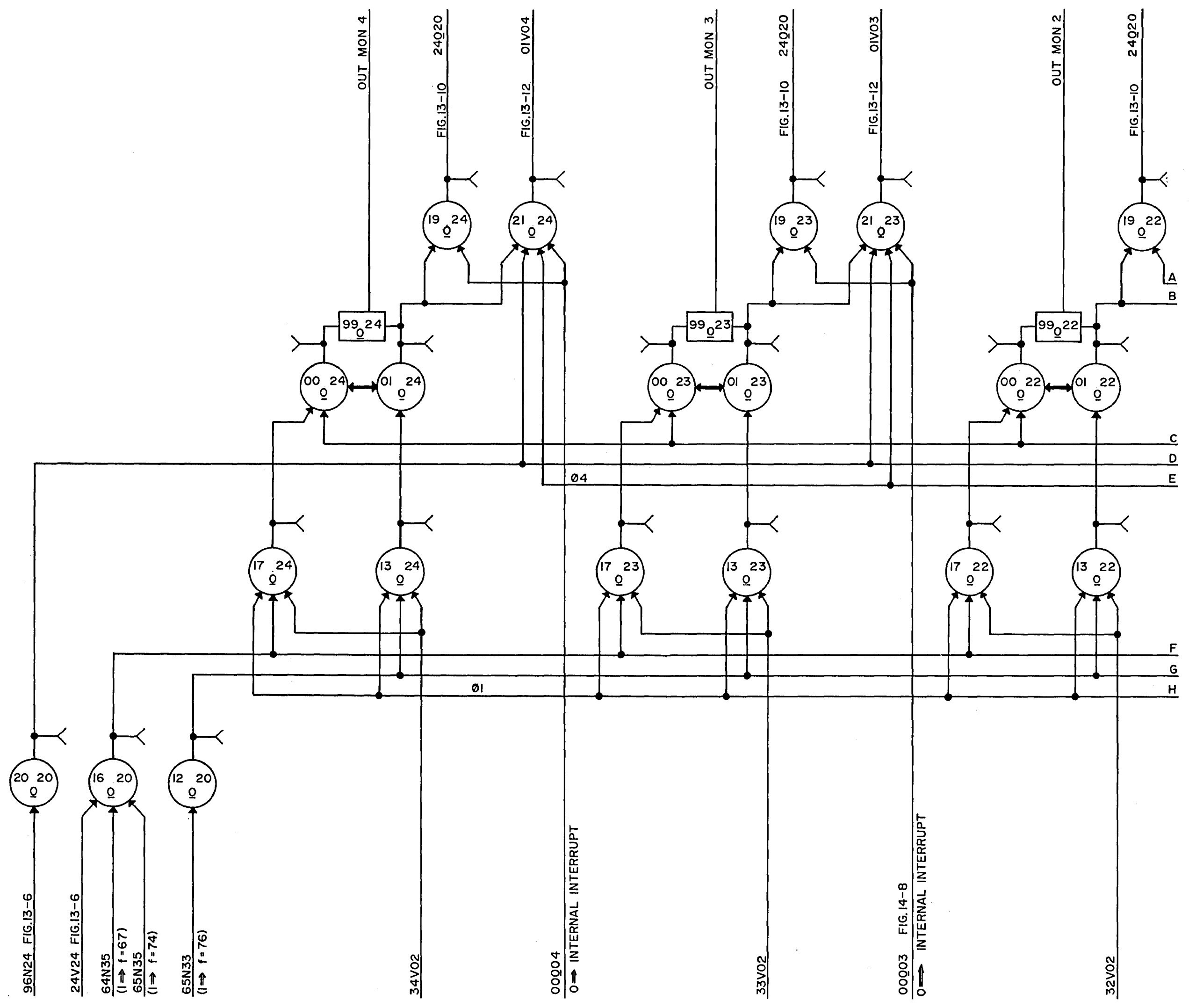
Before the computer sets up an output buffer on channel 0 or 1, it normally checks the status of the Input Active flip-flops in the external computers. The instruction is programmed as follows:

$$13 \quad \hat{j} \quad \hat{k} \quad b \quad y$$

The $\hat{j}$ designator is loaded into the translator during the A sequence to designate the channel to be checked. It must equal "0" or "1" for this use. If the input Active flip-flop on the designated channel is active, there is a "1" output from the control line driver. As seen in figure 14-14, the inputs to the amplifiers are a "0" input from the translator for the designated channel and a "1" input from the control line for an active channel. With these inputs, the output from the ^{44}Y-- circuit is a "1" forcing the output from $^{61}H^{24}$ to "0". This "0" output informs the computer of an active channel and also provides an enable in the D sequence to skip the next instruction.

OUTPUT ACTIVE FLIP-FLOPS

The Output Active flip-flops (fig. 14-8) on channels 0 and 1 are set active by an instruction using f = 74 or 76 (see table 10-2). The

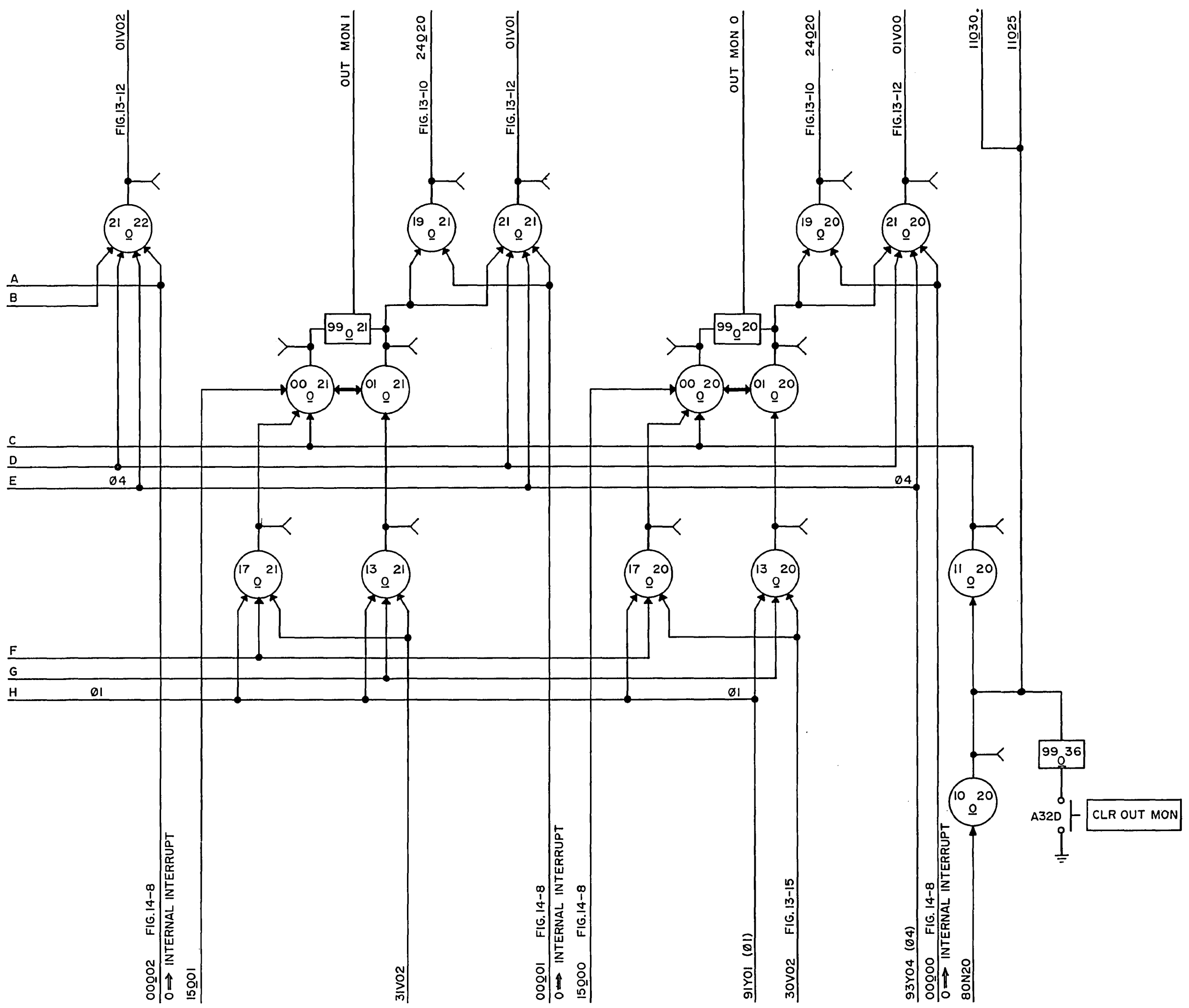


"1" output from an inverter $^{65}N^{37}$ (in the D sequence not shown) forces the output from $^{12}O^{00}$ (fig. 14-8) to "1", one of the enables required by a $^{13}\hat{O}$-- circuit. The ^{13}O-- circuit designated by j in the instruction (input from the I/O translator) is enabled at phase 1 to set the Output Active flip-flop.

Another output from ^{13}O-- is used to clear I^{94} or I^{95} (fig. 13-17) or to clear any previous external interrupt on channel 0 or 1.

With the Output Active flip-flop (fig. 14-8) set, the "0" output to $^{13}V^{4-}$ (fig. 14-15) is one of the enables required to load the Working register. This register sets up the time-share transmission of data between channels 0 and 1 (described later). The other "0" output from $01O$-- (fig. 14-8) is utilized during an instruction with f = $\underset{\wedge}{63}$ (see table 10-2). If the channel designated by j is active, the output from an inverter in the Test I/O Active circuitry (not shown) is a "0". This output satisfies the jump condition as specified in the instruction. If the channel designated by $\hat{j}$ is not active, the output from the Test I/O active circuitry is a "1" that does not satisfy the jump condition.

The computer can set the data lines by loading the C_1 register. It sends the Ready signal on the designated channel and awaits the return of the Resume signal. If the Resume signal is received within the specified time, the computer



124.199

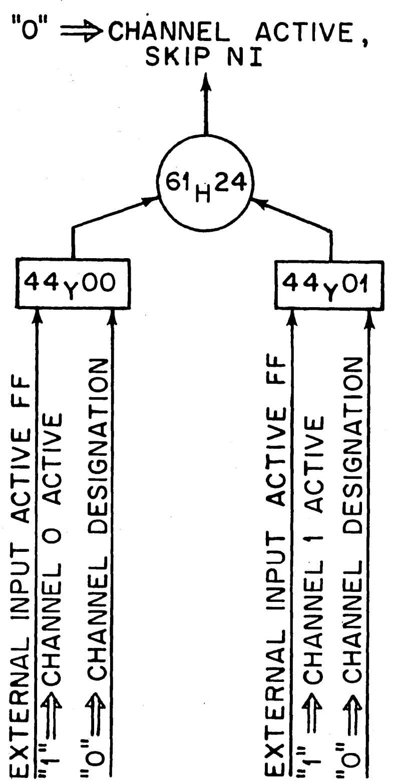
Figure 14-13.—Output monitors, 0 thru 4.

sets up an output transmission on the other intercomputer channel, if it is active. The data transmissions continue normally until the buffer has been completed.

When the initial and terminal addresses of the buffer are equal, V^{71} (the Disable intercomputer Buffer flip-flop) is set by an enable from the I/O timing chain. At time $T90_2$ (fig. 13-7) during Disable I/O_2, the Ready signal is sent on the designated channel. If the Resume signal is received in less than 64 seconds, a normal buffer termination takes place.

At time T82 (fig. 13-6), during any Scan cycle, the output from $^{90}N^{21}$ forces the output from $^{16}O^{00}$ (fig. 14-8) to "0". At phase 1 the designated ^{17}O-- circuit is enabled. The "1" output from ^{17}O-- clears the Output Active flip-flop and the associated flip-flop in the Working register.

If the Output Active flip-flop had been initially set using an instruction with f = 76, an internal interrupt is generated at the normal completion of the buffer. The "0" output from ^{00}O-- is an enable to the ^{19}O-- circuit (fig.



124.200

Figure 14-14.—Circuit for test input buffer active. (f = 13, $\hat{j}$ = 0 or 1).

14-13) of the output monitor to load the Interrupt request into subpriority. This "0" output is also an enable to the ^{21}O-- circuit of the Output monitor to load the Interrupt request into priority.

The buffer can be terminated before completion and without generating any interrupt by the main program. If an instruction is used with f = $_{\wedge}$67, (see table 10-2) the channel designated by j is cleared by a command signal from the C sequence. The "1" output from ^{64}N^{36} (an inverter in the C sequence not shown) forces the output from ^{18}O^{00} (fig. 14-8) to a "0" one of the enables required by ^{19}O--. At phase 1, the designated ^{19}O-- circuit is enabled to clear the Active flip-flop and the Monitor flip-flop (fig. 13-17) to prevent the generation of an internal interrupt. Also an output from ^{19}O-- is used to clear I^{94} or I^{95} (fig. 13-17) preventing the generation of a simulated external interrupt.

If, because of a malfunction, the Resume signal is not received within the specified time, the Active flip-flop is cleared and a simulated external interrupt is generated. The Ready signal is set on the line with 06-set. The "0" output from 06- is one of the enable signals required by ^{15}O--. If the Resume is not received, at time T81$\hat{x}$, the output from ^{94}N^{13} is a "1" that indicates a simulated Resume. The "1" output from ^{94}N^{13} forces the output from ^{14}O^{00} to a "0" that enables the designated ^{15}O-- circuit. The "1" output from ^{15}O-- clears the active flip-flop, clears the monitor to prevent the initiation of an Internal interrupt, clears the associated flip-flop in the Working register, and sets I^{94} or I^{95} to initiate a simulated External interrupt.

WORKING REGISTER

The working register, comprising flip-flops V^{40}, V^{41}, and V^{58} functions as a time-share circuit for data transmissions on channels 0 and 1. If both channels are active, transmissions are automatically alternated between them. Associated with the register is a loading flip-flop (V^{50}) that enables and disables the loading of the register. The register can only be loaded after both flip-flops are cleared.

When the computer is master cleared, the output from ^{11}O^{47} is a "1" (see figure 14-9). This "1" output clears both working Register flip-flops (fig. 14-15) prior to the loading cycle. The Loading flip-flop (V^{58}) is cleared by a signal from the I/O timing chain (fig. 13-6). Each time ^{13}T^{83} is enabled, its "1" output clears V^{58} (fig. 14-15). When V^{58} is cleared, the register cannot be loaded. In order to load the register, ^{13}V^{58} must be enabled. Two of the required enables are from the Register flip-flops, V^{40} and V^{41}. Both these flip-flops must be cleared. The command to load the register is generated by the I/O timing chain at time T80 (fig. 13-6) during Scan. This is the same time that the Command signal is initiated to clear subpriority and priority. The output from ^{10}V^{50} is a "0" (see figure 13-10) that enables ^{13}V^{58} at phase 1. The "1" output from ^{13}V^{58} sets V^{58} providing an enable to the ^{13}V-- circuits. The other enable to these circuits is from the output Active flip-flops (fig. 14-8). At phase 3, either one or both of the ^{13}V-- circuits are enabled, depending on the status of the Output Active flip-flops.

In order to illustrate the time-share function of the register, assume both channels to be active. Under this condition, both V^{40} and V^{41}

are set by the "1" output from the ^{13}V-- circuits. The output from $^{15}V^{40}$ is now forced to a "0" by the "1" inputs from both $^{00}V^{40}$ and $^{00}V^{41}$. This "0" output is one of the enables required by $^{13}V^{50}$ (fig. 13-11) to load the Out-on-2 request into subpriority. At the same time the request is loaded into subpriority, the "1" output from $^{13}T^{83}$ (fig. 13-6) clears V^{58}. This removes the enable from the ^{13}V-- circuit. When the request is processed by subpriority, enables are generated to load the request into priority to determine channel priority. The "0" outputs from $^{01}V^{40}$ and $^{01}V^{41}$ are required enables to the ^{17}V-- circuits (fig. 13-12) in priority. When the higher priority channel is designated, in this case channel 1, a signal is sent from priority to clear the associated flip-flop in the Working register. Therefore a "1" output from $^{19}V^{01}$ clears V^{41}, but does not clear V^{40}. On the next loading cycle, $^{13}V^{58}$ is disabled by the "1" input from $^{00}V^{40}$. The next Out-on-2 request must be done on channel 0 since V^{40} is the only flip-flop set. The sequence of events is repeated with V^{40} being cleared by the "1" output from $^{19}V^{00}$ (fig. 13-12). Now, on the next loading cycle, $^{13}V^{58}$ can be enabled because both V^{40} and V^{41} are cleared. This time sharing function of the Working register is only applicable when channels 0 and 1 are both in use.

When I/O is processing Out-on-2 requests, the Resume signal must be recieved before another request can be loaded into subpriority. So if the last word buffered on channel 0 or 1 caused the initial and terminal addresses to be equal, the buffer would be terminated. Again, however, before the Output Active flip-flop can be cleared, the Resume signal must be received. In the meantime, it is conceivable that the Working register could have been cleared and then loaded on the next loading cycle. This could possibly set up an Out-on-2 request on a channel where the buffer has been completed. To prevent this, the Ready signal enables a ^{17}O-- circuit (fig. 14-8) before an Out-on-2 request can be processed. The "1" output from ^{17}O-- clears the flip-flop in the Working register associated with the now inactive channel. With V^{40} or V^{41} (fig. 14-15) cleared, no Out-on-2 request is processed for that particular channel.

Now assume that only one channel is set active (channel 1). In the Working register, V^{41} is the only flip-flop set during the loading cycle. When the Out-on-2 request is loaded into priority, the "1" output from $^{19}V^{01}$ (fig. 13-12) clears V^{41}. The Working register is now completely cleared with V^{40} and V^{41} cleared. During a Scan cycle at time $T80_x$, the Working register is again loaded. Since channel 1 is the only channel active, V^{41} is the only flip-flop set. This Out-on-2 request is loaded into priority during Scan at time $T82_x$, after a Resume is received.

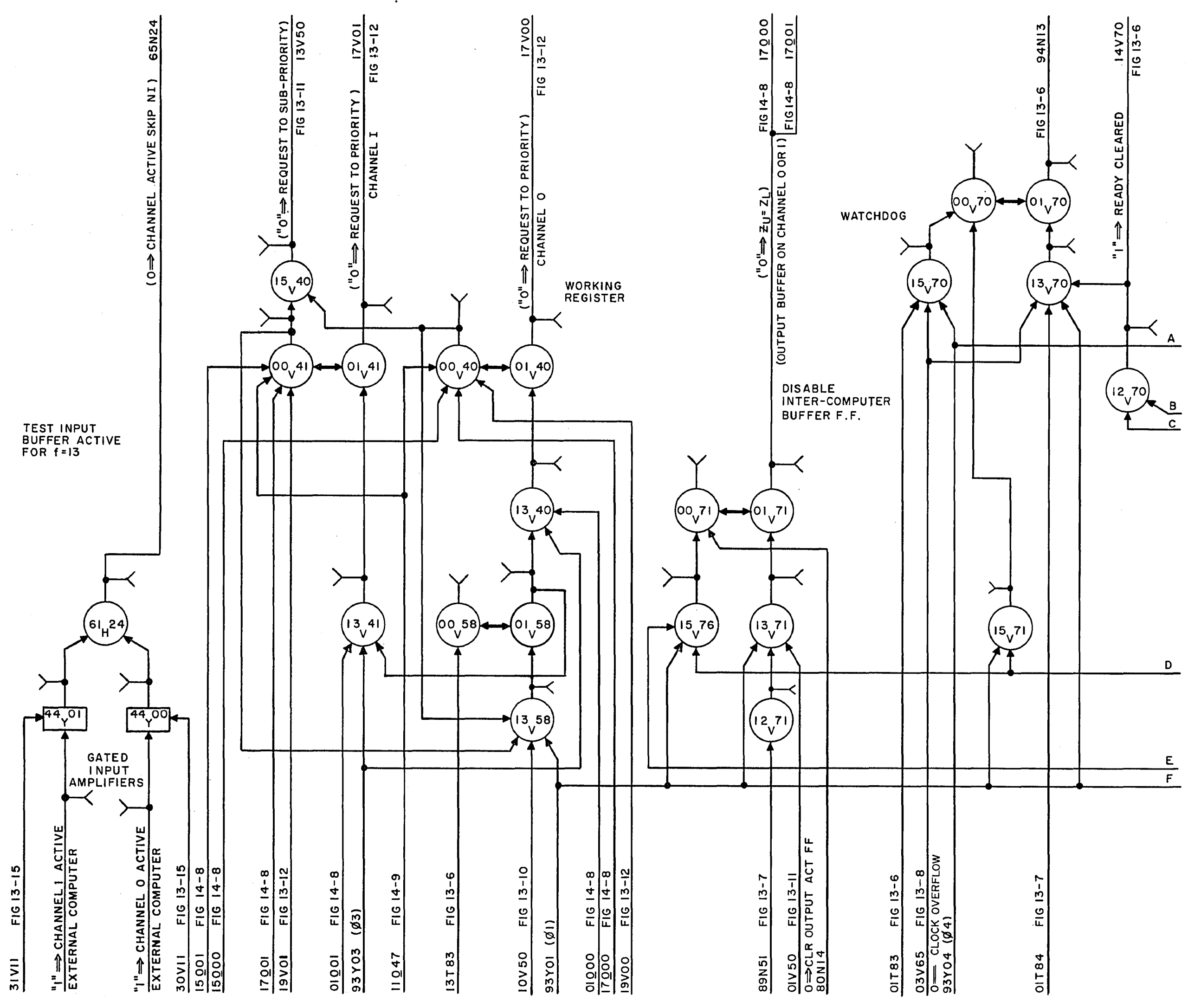
PRIORITY DETERMINATION

When an Out-on-2 request is to be loaded into subpriority, $^{13}V^{50}$ (fig. 13-1) must be enabled. Since Out-on-2 has the lowest priority of all requests, the inputs to $^{13}V^{50}$ from the other circuits in subpriority (discussed in chapter 13) must be "0's". This indicates that a request with a higher priority is not being processed. Also, the input from $^{12}V^{52}$ must be "0" indicating that the last request processed was not an output.

The output from $^{15}V^{40}$ in the Working register is "0" indicating that either channel 0 or 1, or both, are active. The command to load subpriority is generated by the I/O timing chain as discussed. If a Resume is set or if the Ready flip-flops are cleared, the output from $^{90}N^{21}$ (fig. 13-6) is a "1". This occurs during Scan at time $T82_x$. The "1" output from $^{90}N^{21}$ forces the output from $^{12}V^{50}$ (fig. 13-1) to a "0" enabling $^{13}V^{50}$ at phase 1.

The "1" output from $^{13}V^{50}$ sets V^{50}, the Out-on-2 flip-flop in subpriority. With V^{50} set, the "1" output from $^{00}V^{50}$ forces the output from $^{05}V^{50}$ and $^{03}V^{50}$ to "0". The "0" output from $^{05}V^{50}$ to the ^{17}V-- circuits in priority (fig. 13-12) enables the loading of the request to designate the channel. The "0" output to the ^{19}V-- circuits is one of the enables required to clear the designated flip-flop in the Working register. The "0" output from $^{03}V^{50}$ (fig. 13-11) to $^{13}V^{35}$ (fig. 13-14) in the translator is one of the enables required to set Mode flip-flop V^{35} (Out-on-2). The other "0" output from $^{13}V^{50}$ (fig. 13-11) to $^{17}O^{40}$ (fig. 14-15) is an enable used to clear the Resume flip-flop in preparation for the reception of the next Resume for the request being processed. The "0" output from $^{01}V^{50}$ (fig. 13-11) is used as an enable by $^{13}V^{71}$ which is fully enabled if the buffer is completed on the request being processed. The output from $^{13}V^{71}$ sets V^{71}, the Disable Buffer flip-flop for channels 0 and 1.

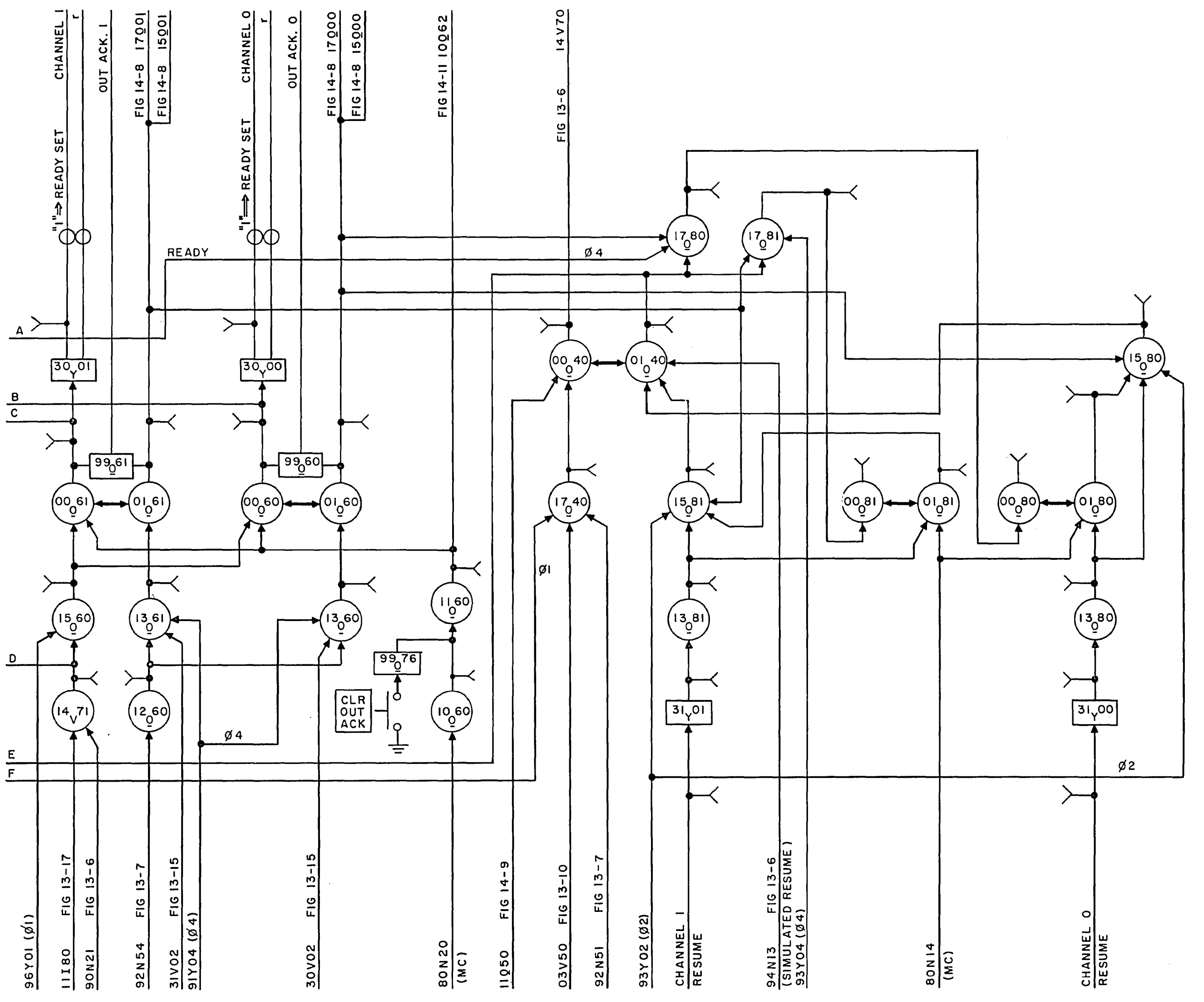
The "0" output from $^{01}V^{50}$ is inverted to a "1" output by $^{02}V^{50}$ to be used to set up the



special address in priority of the Output Buffer Control register. The inputs to $^{15}V^{04}$ and $^{15}V^{06}$ (fig. 13-12) set up the address 00120 for channel 0, and priority increments this address.

An Out-on-2 request is loaded into priority on a command from the I/O timing chain coupled with the enables from the Working register and subpriority. At times T83 (fig. 13-6) during Scan, the output from $^{90}N^{24}$ is a "1" forcing the output from $^{16}V^{00}$ to "0". At phase 4, either one or both of the ^{17}V-- circuits are enabled, depending on the status of the Working register (fig. 14-15). If both circuits are enabled, V^{01} and V^{00} (fig. 13-12) are both set. However, channel 1 has priority because the "1" output from $^{00}V^{01}$ to $^{15}V^{00}$ increments the special address to setup address 00121. If only $^{17}V^{01}$ was enabled to set V^{01}, the "0" output from $^{01}V^{01}$ would be inverted to a "1" output by $^{02}V^{01}$ to set V^{00}. This is required since the "1" output from $^{00}V^{00}$ provides a signal used to enable the Memory Access flip-flop. There is no output from V^{01} to the Memory Access flip-flop.

If only $^{17}V^{00}$ were enabled to set V^{00}, this would be the only flip-flop set, since the special address for channel 0 has already been set up. Outputs from V^{01} and V^{00} are also used to enable the clearing of the Working register.



124.201

Figure 14-15.—Intercomputer logic.

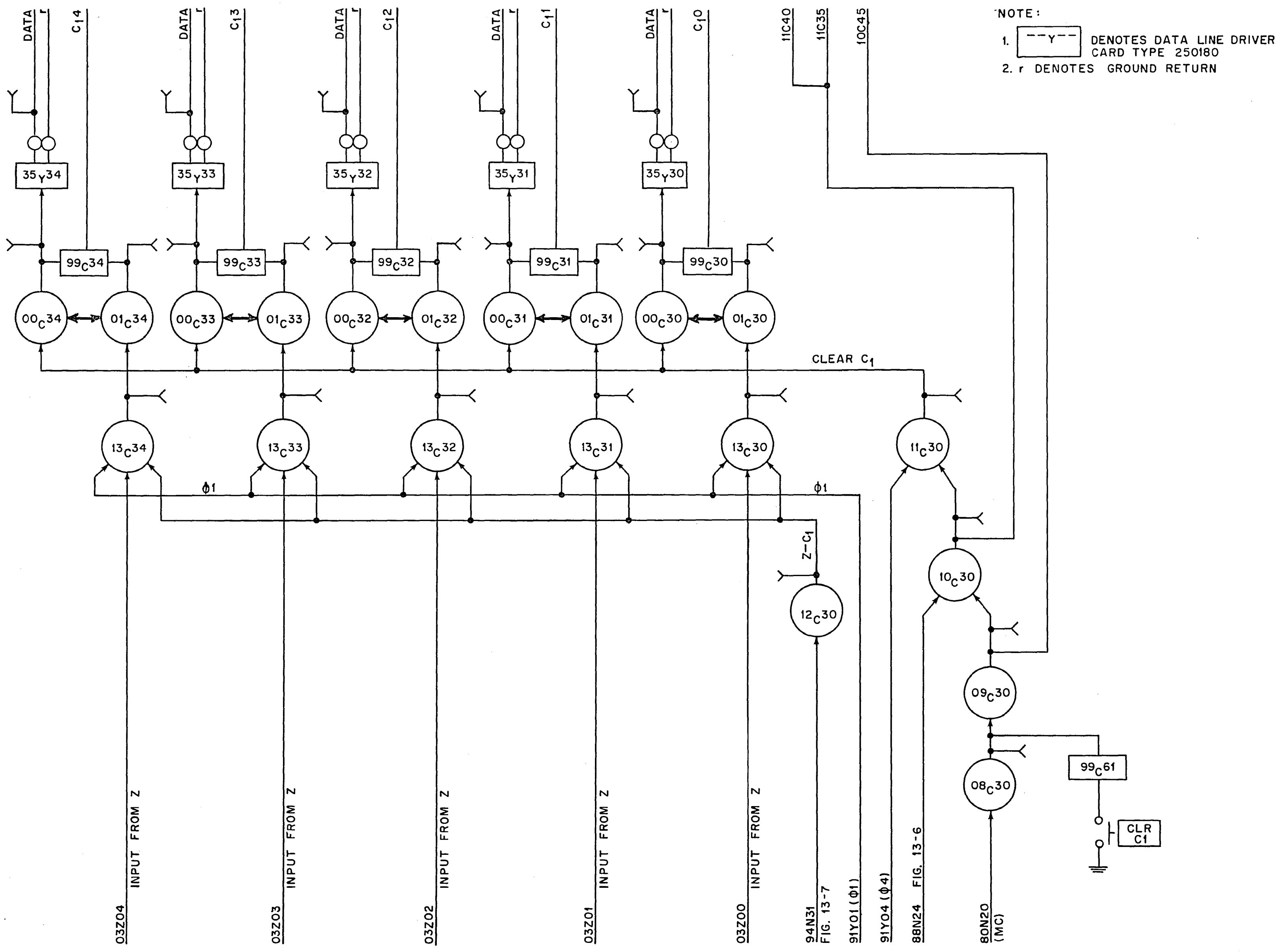
One of the enable inputs to the 19V-- circuits is from subpriority ("0" input from 05V50). If both V00 and V01 are set, 19V01 can be enabled by the "0" input from 01V01, but 19V00 is disabled by the "1" output from 02V01. If V01 is set and V00 is cleared, 19V01 can be enabled by the "0" input from 01V01, and 19V00 is again disabled by the "1" output from 02V01. If V00 is set and V01 is cleared, 19V01 is disabled by the "1" input from 01V01, and 19V00 can be enabled by the "0" inputs from 01V00 and 02V01. The command signal is initiated by the I/O timing chain at time T88.

The output from 90N51 is a "1" that forces the output from 18V00 to "0". At phase 1, the 19V-- circuit corresponding to the priority-designated channel is enabled. The "1" output from this circuit clears the associated flip-flop in the Working register.

The normal transmission from priority to the S-register and to the I/O translator is carried out as in the case of any output request.

OUTPUT DATA REGISTER

The C_1 register (see figures 8-10 and 14-16) handles the data transfer to external computers



124.202

Figure 14-16.—C_1 Register 00-04.

on channels 0 and 1. The register consists of 30 stages to allow the transfer of one word for each Out-on-2 request. The output from each ^{00}C-- circuit in the C1 register drives one data line driver, ^{35}Y--. Each data line driver is capable of driving two external input amplifiers (not shown).

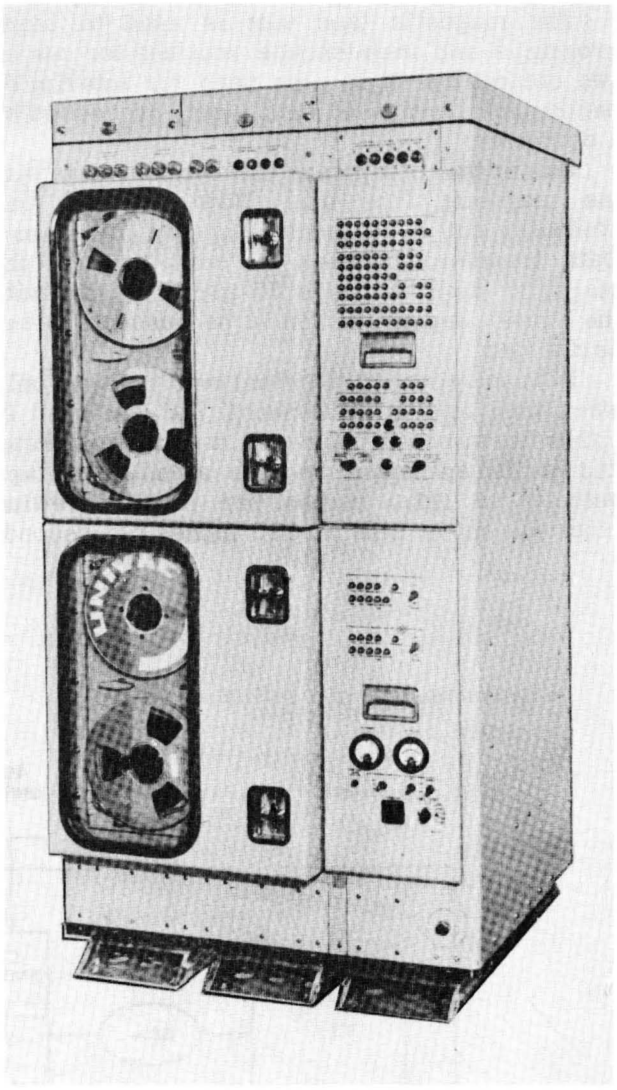
When the computer is master cleared, the "1" output from ^{80}N^{20} (fig. 14-16) forces the output from ^{08}C^{30} to "0". This "0" output is inverted by ^{09}C30 to a "1" output forcing the output from ^{10}C30 (and ^{10}C45 not shown) to "0". This output is utilized at phase 4 to enable the ^{11}C-- circuits to clear the register. When the associated button (C_1, see figure 10-8) is depressed, the output from ^{08}C^{30} (fig. 14-16) is clamped at "0". The clearing action is as for master clear. Associated with each bit position of the register is a set indicator-pushbutton (fig. 10-8).

During an Out-on-2 request, the C_1 register is cleared and loaded by commands from the I/O timing chain. During I/O_2 at time T83_2 (fig. 13-6) the output from ^{88}N24 is a "1" forcing the output from ^{10}C-- to "0". This output is utilized at phase 4 to enable the ^{11}C-- circuits to clear the register. During I/O_2 at time T84_2 (fig. 13-7) the command is initiated to load the register. At this time the output from ^{94}N31 and ^{95}N31 is a "1" forcing the output from the 12C-- circuits (only 12C30 is shown in fig. 14-16) to "0". This output is an enable to the ^{13}C-- circuits. The data input to the C^1 register is from the corresponding bits of the Z-register (fig. 8-1). A "0" input from Z causes a "1" to be stored in the C_1 register flip-flops. To load a "0" into C_1 requires a "1" input from Z.

At phase 1 which of the ^{13}C-- circuits is enabled depends on the data transmission from Z. When a Register flip-flop is set, the "1" output from ^{00}C-- drives the line drivers so that the output from ^{35}Y-- is a "1". When a Register flip-flop is cleared, the output from ^{35}Y-- is "0". The data is placed on the lines of channels 0 and 1, but is only received by the external computer to which the Ready signal is sent. The data remains on the lines until the C_1 register is cleared.

RECORDER REPRODUCER RD-243/USQ-20(v)

The Recorder-Reproducer RD-243/USQ-20(v) is shown in figure 14-17 and hereafter is referred to as the magnetic tape unit. This tape unit is a piece of peripheral equipment (an I/O device) for the NTDS computers.


124.203
Figure 14-17.—Recorder-Reproducer, RD-243/USQ-20(V).

DESCRIPTION

The magnetic tape unit (including the tape transports and control circuits), provides medium-speed, large-capacity auxiliary storage for a computer. Information is recorded on magnetic tape for later use by the computer or other data processing equipment. Control circuits interpret computer commands, causing a tape transport to move tape across its read/write head. By this action the Control circuits receive data from or transmit data to the computer until the computer-originated command is completed.

The magnetic tape unit is used to store programs and maintenance routines for one or two computers. It is also used for interim or permanent storage of data being processed by a computer.

One or two computers can communicate with the magnetic tape unit. However, only one computer can be in control at any given time. Data transfers between the computer and the magnetic tape unit are 30 parallel bits, with the rate dependent on type function being performed.

Data is recorded (written) onto the tape only when tape motion is forward. Data is read in either forward or backward tape motion. Read and write operation require a magnetic tape velocity of 112.5 inches per second; rewind functions move tape at 225 inches per second.

FUNCTIONAL SECTIONS

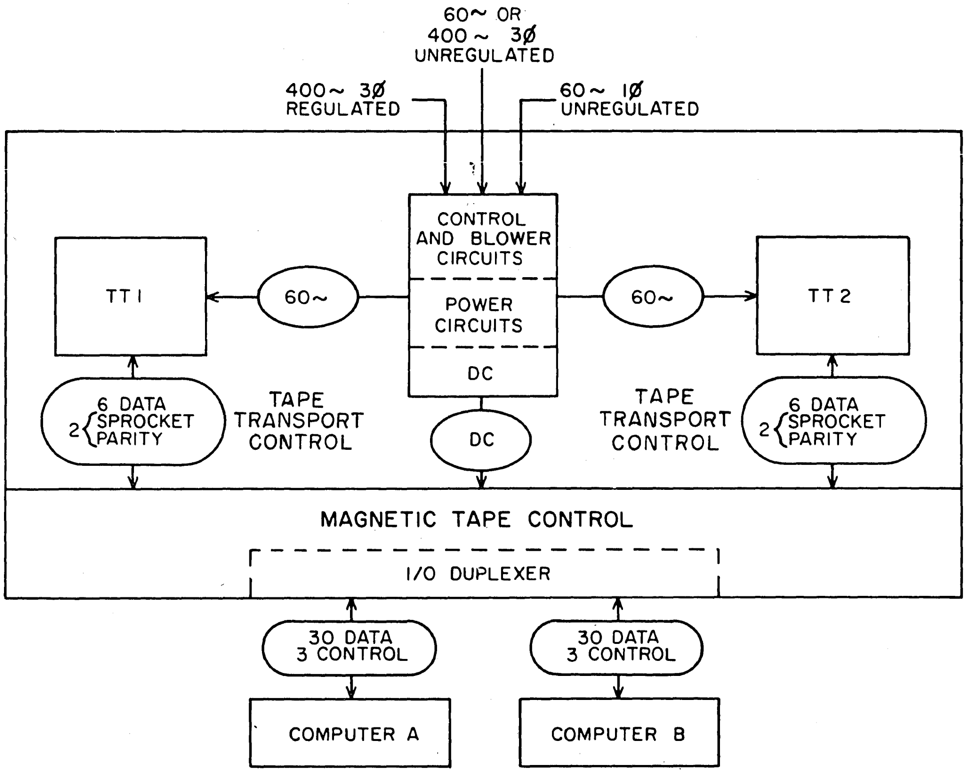
The magnetic tape unit is divided functionally into the following major sections (fig. 14-18): Magnetic Tape Control, (I/O duplexer), Tape Transports, and Power.

Duplexer

The duplexer ensures that only one computer is communicating with the magnetic tape system at any given time. It interprets requests for control by a computer to determine if the request can be met or not.

Magnetic Tape Control

Magnetic tape control (MTC) interprets computer external function words to determine the type of operation required. During operation,



124.204

Figure 14-18.—Block diagram, magnetic tape unit with computers.

it either breaks down (disassembles), or assembles 30-bit computer data words, depending on type of function specified. It also checks for data and timing errors, sending this information to the controlling computer at the end of a function.

Tape Transport Control

Tape transport control determines if the specified tape transport can perform the function requested by magnetic tape control. If the function is legal, tape motion is initiated; if not, tape transport control informs magnetic tape control that an improper condition exists and the function cannot be executed.

Tape Transport (TT1 and TT2)

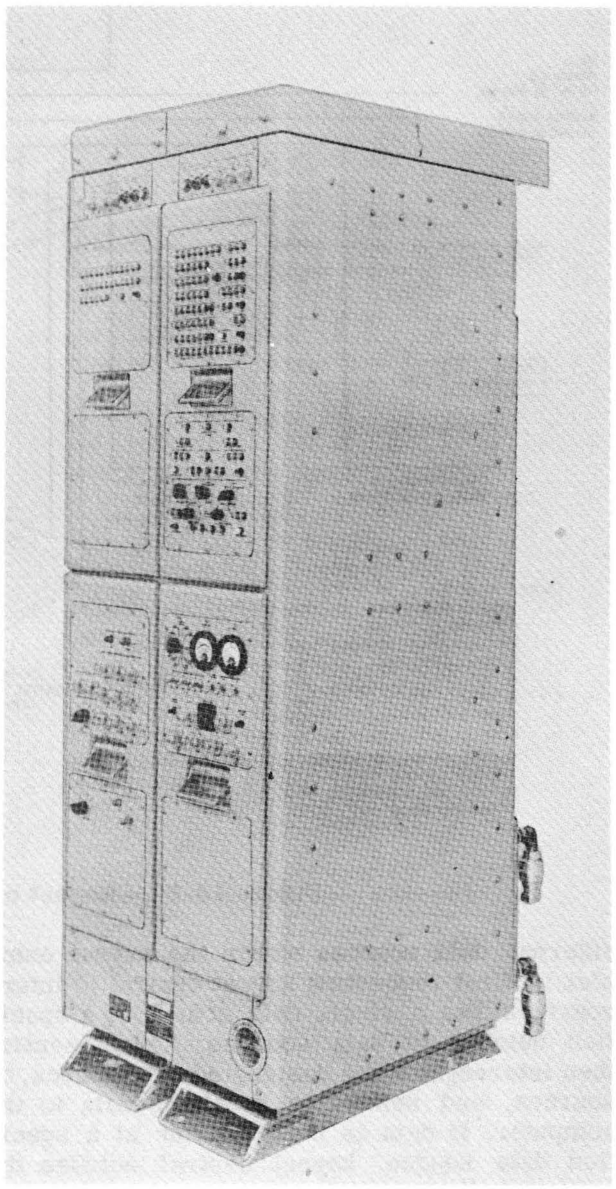
The tape transport moves tape across the read/write head as directed by tape transport control. Two tape speeds are provided (as discussed).

Power

Power and power control circuits provide regulated d-c voltages for control circuits.

DIGITAL DATA CONVERTER CV-1123/USQ-20(V)

The Digital Data Converter CV-1123/USQ-20(V) (fig. 14-19), serves as the control center of the NTDS data gathering system (hereinafter referred to as the keyset complex). Figure 14-20 illustrates the functional relationship of the equipment comprising the keyset complex. Each unit in fig. 14-20 (except those which may vary according to installation site) is identified by both its assigned designation and its colloquial name. For convenience, colloquial names will be used hereafter, whenever possible.


124.205
Figure 14-19.—Digital Data Converter (Keyset central) CV-1123/USQ-20(v).

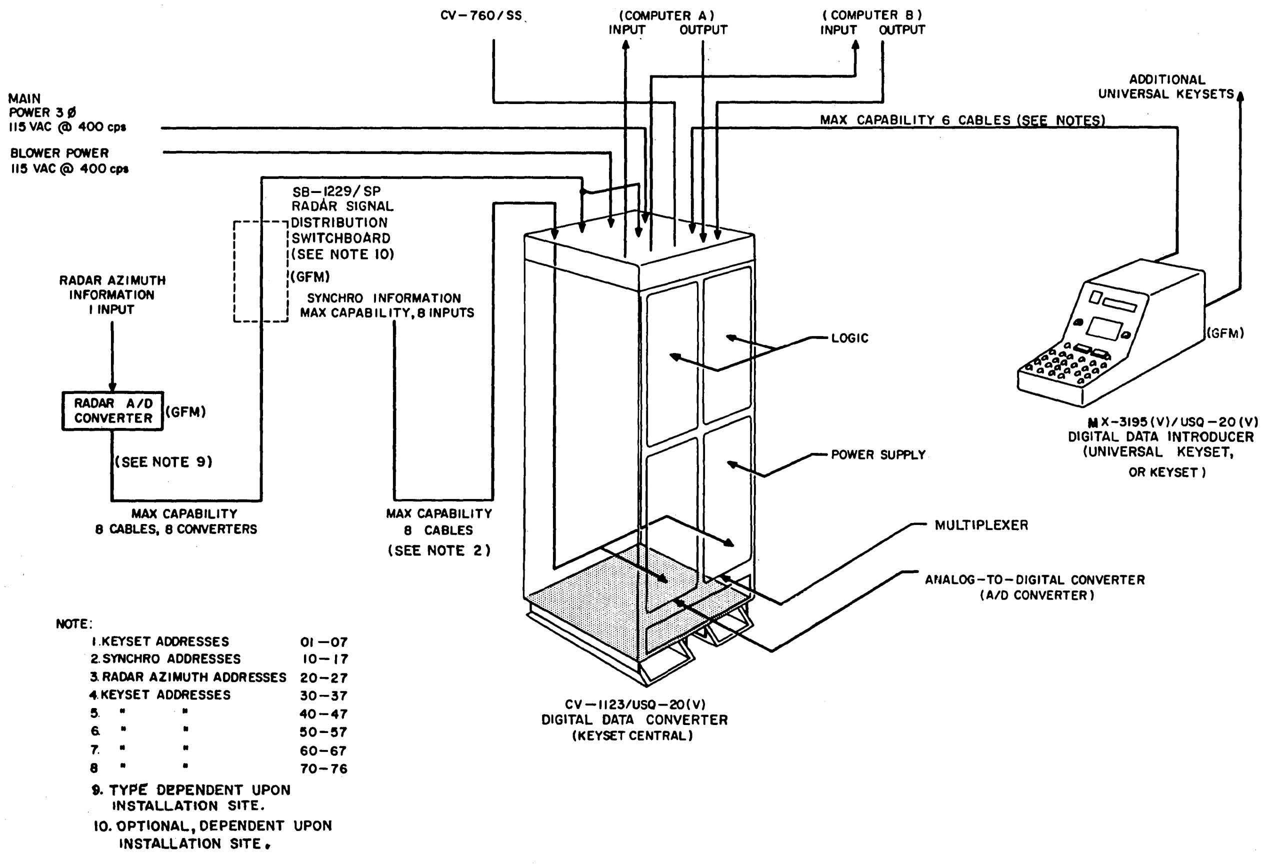
FUNCTIONAL DESCRIPTION

The keyset complex (also called keyset central) allows data to be entered into the computer from a maximum of 62 data sources, through the use of one computer input/output channel.

KEYSET COMPLEX CONTROL UNIT

Keyset central controls the flow of data from the keyset complex external data sources to a computer. Duplexing circuits allow keyset central to be used with either one or two computers.

Keyset central derives its colloquial name from the fact that within the keyset complex the unit is functionally located between the external data sources and the computer. In operation, the computer may receive data from any of the


CV-760/SS
(COMPUTER A) INPUT OUTPUT
(COMPUTER B) INPUT OUTPUT
MAIN POWER 3 Ø 115 VAC @ 400 cps
BLOWER POWER 115 VAC @ 400 cps
ADDITIONAL UNIVERSAL KEYSETS
MAX CAPABILITY 6 CABLES (SEE NOTES)
SB-1229/SP RADAR SIGNAL DISTRIBUTION SWITCHBOARD (SEE NOTE 10) (GFM)
SYNCHRO INFORMATION MAX CAPABILITY, 8 INPUTS
RADAR AZIMUTH INFORMATION 1 INPUT
RADAR A/D CONVERTER (GFM)
(SEE NOTE 9)
MAX CAPABILITY 8 CABLES, 8 CONVERTERS
MAX CAPABILITY 8 CABLES (SEE NOTE 2)
LOGIC
POWER SUPPLY
MULTIPLEXER
ANALOG-TO-DIGITAL CONVERTER (A/D CONVERTER)
(GFM)
MX-3195(V)/USQ-20(V) DIGITAL DATA INTRODUCER (UNIVERSAL KEYSET, OR KEYSET)
CV-1123/USQ-20(V) DIGITAL DATA CONVERTER (KEYSET CENTRAL)
NOTE:
1. KEYSET ADDRESSES 01-07
2. SYNCHRO ADDRESSES 10-17
3. RADAR AZIMUTH ADDRESSES 20-27
4. KEYSET ADDRESSES 30-37
5. " " 40-47
6. " " 50-57
7. " " 60-67
8. " " 70-76
9. TYPE DEPENDENT UPON INSTALLATION SITE.
10. OPTIONAL, DEPENDENT UPON INSTALLATION SITE.

124.206

Figure 14-20.—Keyset complex, relationship of units.

external data sources within the keyset complex by first requesting keyset central to interrogate either a single data source or a specified number of data sources. Keyset central then interrogates the designated data source, or sources, and sends the available data to the computer. If data is not available at a specified data source, keyset central notifies the computer that data does not exist at the data source.

EXTERNAL DATA SOURCES

Three types of data input equipment produce the data which is processed and entered into the computer via keyset central. These are universal Keysets, radar azimuth converters, and multiplex and analog-to-digital converters.

Universal Keysets

A maximum of 46 keysets may be used in the keyset complex. The keyset is a manually actuated device incorporating pushbutton control keys whereby the data is entered into the keyset. The data is stored within the keyset as a 30-bit data word until it is requested by keyset central upon demand of the computer.

Removable keyboard overlay panels and data display units allow the keysets to be adapted for various types of data entry. Keyset modification consists of fitting a keyset with the correct keyboard overlay panel and data display units, so as to reflect the type of data being entered into the keyset.

Radar Azimuth Converters

A maximum of eight Radar Azimuth converters may be used to supply data to keyset central (one converter unit serves one radar data source). Azimuth information from the radar set or synchro is originally in the analog form. The purpose of the Radar Azimuth converter is to convert the azimuth information from an analog form to a digital representation. Radar Azimuth information consists of 12 bits of data, namely bits 2^0 through 2^{11} (base 10). Data from this source is always available and may be sampled at any time (except when the data is being changed) by a computer acting through keyset central.

Multiplexer and Analog-To-Digital Converter

The multiplexer and A/D converter are located within the cabinet of keyset central.

The function of the multiplexer is to transmit to the A/D converter a single set of synchro stator voltages selected from one of its eight double-speed or single-speed synchro system inputs. The A/D converter then performs an analysis of the phase and amplitude relationship of the received synchro stator voltages and transforms the received analog information into a 16-bit binary representation of the synchro shaft angle being transmitted. Data from this source may be sampled at any time provided the original synchro reference voltage has attained the minimum amplitude for analysis by the A/D converter.

SIGNAL DATA CONVERTER, CV-760/SS

The Signal Data Converter CV-760/SS (fig. 14-21) is a special analog to digital devices used in the NTDS complex. The relationship between the CV-760/SS and the other parts of the NTDS system is shown in figure 14-22. All communication and power cables illustrated in figure 14-22 enter the top rear (not shown) of the equipment cabinet (fig. 14-21). Cooling water connections are made at the lower left rear of the cabinet.

The colloquial name for Signal Data Converter is the "video processor." The video processor serves as an NTDS peripheral equipment.

The video processor is an analog to digital converter which receives information from one or two surveillance radars, converts the information from analog to digital form, analyzes the information for true target echoes, and reports the azimuth and range location of detected targets to a computer which performs automatic tracking of the targets.

This model video processor will also censor areas of excess radar clutter and detect and report jammer bearings to the computer.

The video processor is used with Radar Set AN/SPS-12, Radar Set AN/SPS-37A, and Radar Trainer AN/SPS-T2. Raw video range data is sent directly to the video processor from the radars. Azimuth data is received in raw video form by an azimuth converter, converted to digital form, and sent through keyset central to the video processor. Keyset does not alter the information in transit from the azimuth converter to the video processor.

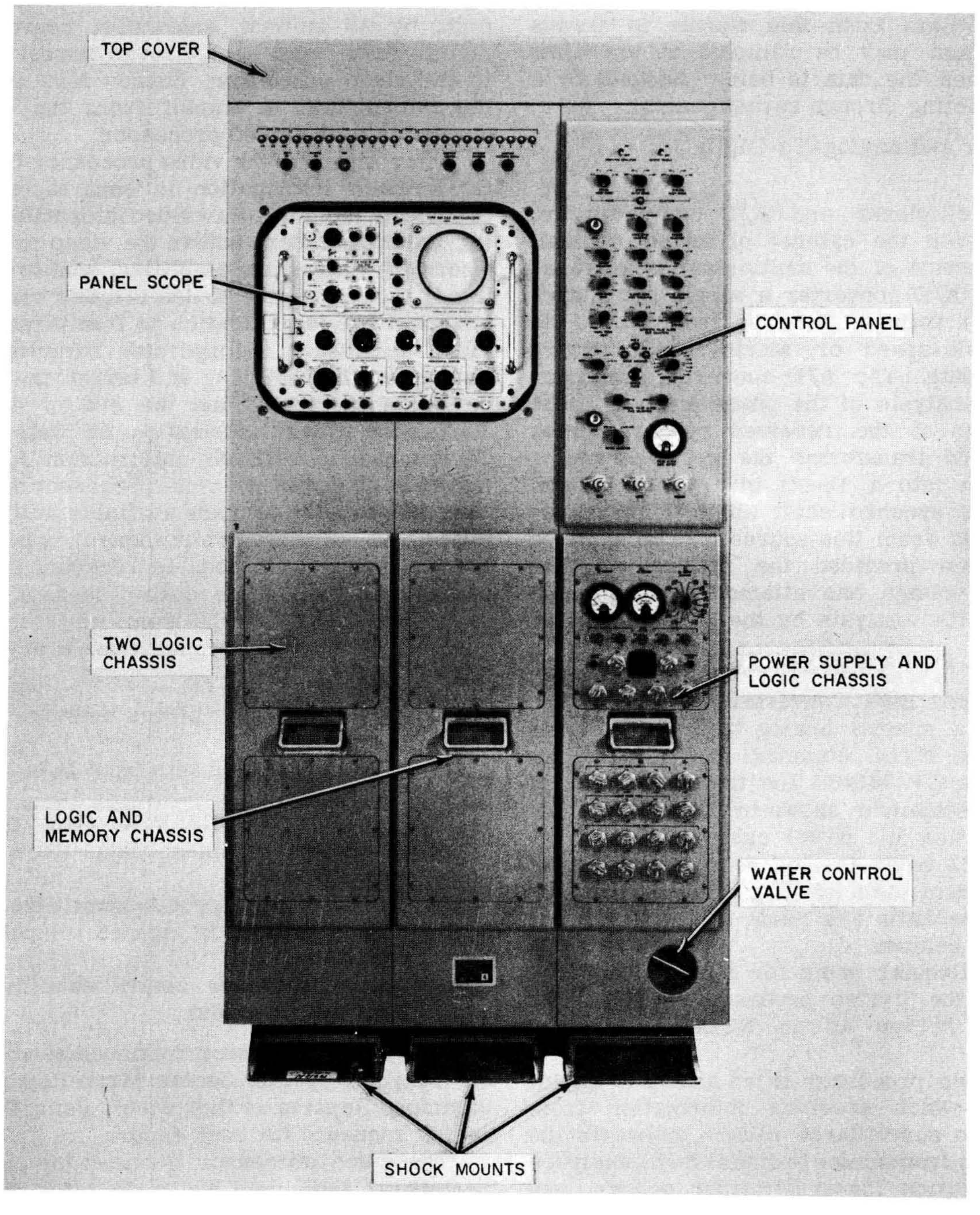
After entering the video processor the digital azimuth information is sent to the video processor memory. Raw video information coming from the radars enters the video processor where the echoes are amplified, sent to circuits (bins) corresponding to the target's range, and analyzed for identification as true targets. The video processor incorporates circuits which determine the beginning of a target, the continuation of a target, and the end of a target.

Target range information as well as the corresponding azimuth information for each target is stored in the video processor memory. This information is made available to the video processor Static register where it is held until the proper function code is received from the computer. At this time video processor control is turned over to the computer requesting control. The normal sequence for every report is as follows:

1) Video processor places message on data lines.
2) Video processor sets Input Data Request control line.
3) Computer detects Input Data Request.
4) Computer samples data lines at its convenience.
5) Computer sets Input Acknowledge.
6) Video processor senses Input Acknowledge.
7) Video processor clears data lines and Input Data Request.

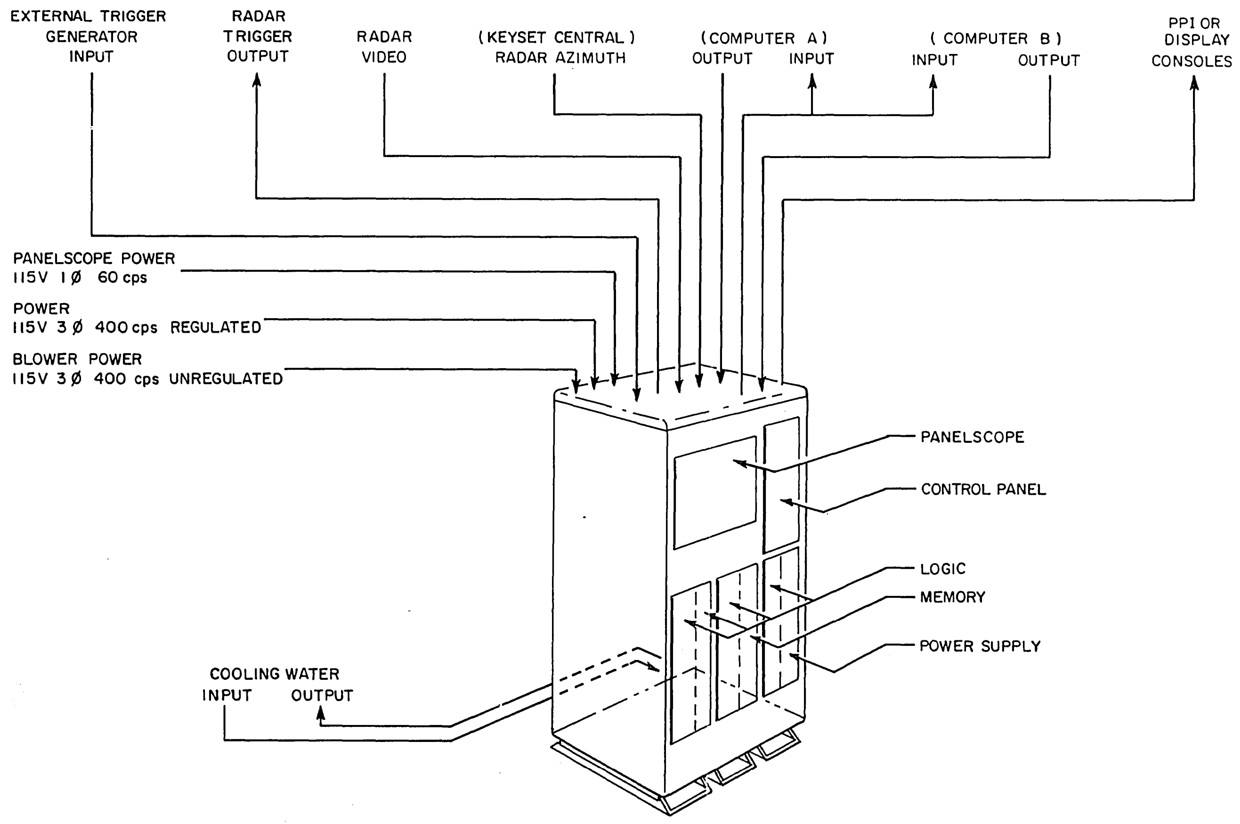
The video processor continues sending data on Target Reports, Sector Mark Reports and Jamming Reports as they occur, using the preceding sequence for each report.

The video processor is one of the pieces of peripheral equipment which operates in a duplexing mode to communicate with two computers. It contains a standardized duplexing logic circuit.



124.207

Figure 14-21.—Signal Data Converter, CV-760/SS front view.


EXTERNAL TRIGGER GENERATOR INPUT
RADAR TRIGGER OUTPUT
RADAR VIDEO
(KEYSET CENTRAL) RADAR AZIMUTH
(COMPUTER A) OUTPUT INPUT
(COMPUTER B) INPUT OUTPUT
PPI OR DISPLAY CONSOLES
PANELSCOPE POWER 115V 1Ø 60 cps
POWER 115V 3Ø 400 cps REGULATED
BLOWER POWER 115V 3Ø 400 cps UNREGULATED
PANELSCOPE
CONTROL PANEL
LOGIC
MEMORY
POWER SUPPLY
COOLING WATER INPUT OUTPUT

124.208

Figure 14-22.—Signal Data Converter, CV-760/SS relationship of units.

All reports to the computer consist of one-word transmissions. There are four types of reports generated by the video processor:

1) Sector Mark Report
2) Jamming Report
3) Target Report
4) Test Message Report

The Sector Mark Report will be generated and transmitted to the computer 32 times per antenna revolution. This divides a scan into 32 sectors of 11.25 degrees each. These sector marks are used by the computer to keep track of antenna position. No use of range is made with a Sector Mark Report. One bit of the report indicates which radar azimuth generated the report.

The Jamming Report is a report generated by the video processor and sent to the computer indicating the center bearing of jamming areas. The range sent is always zero and is not a measure of the range to the jammed area; therefore, it is not usable data. A bit of the report message indicates which radar set detected the jamming.

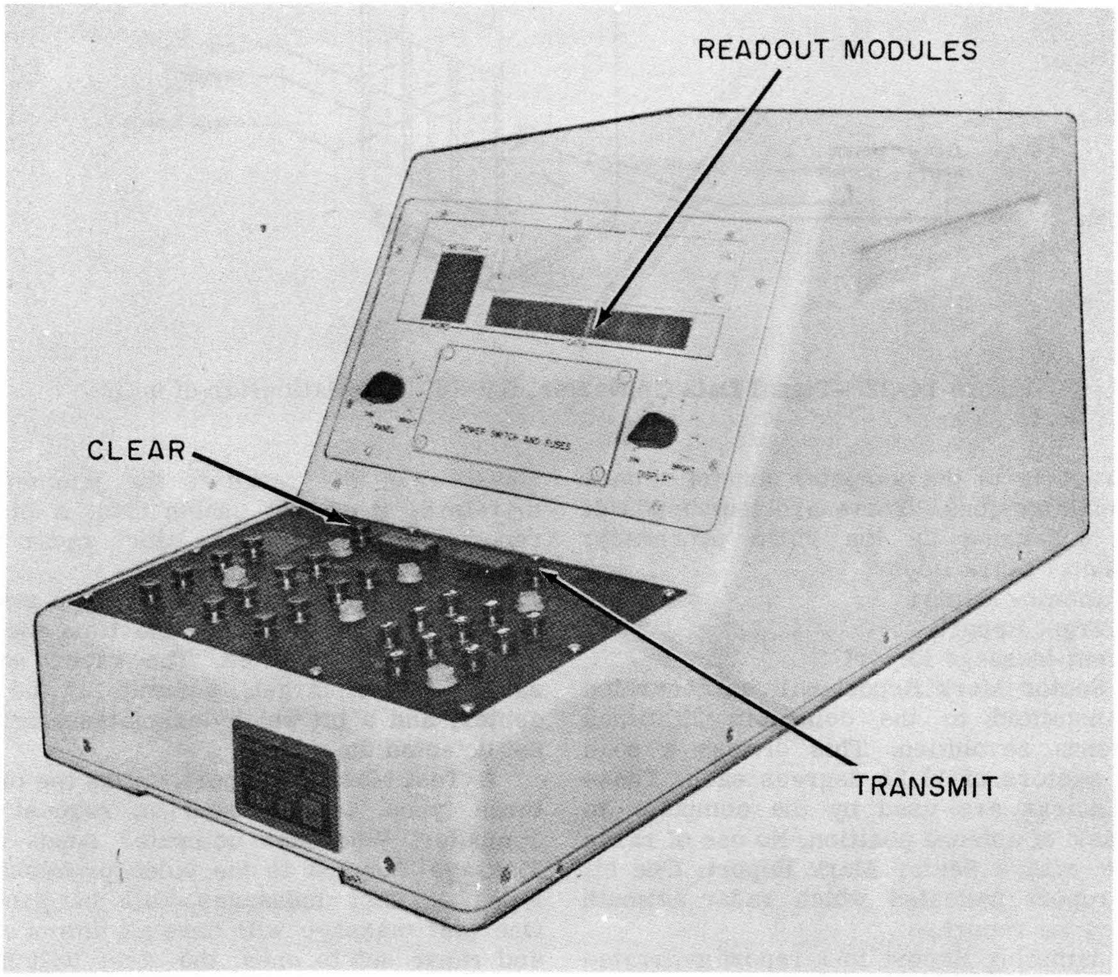
The Target Report is a message generated by the video processor at the time the end of the target is detected. The report contains range to the target, azimuth of the target center, and a bit which designates which radar set detected the target.

A Test Message Report, unlike the previous three types, is only sent on request by the computer. When the computer sends a Test Message Request to the video processor, two types of test messages will be generated. One test message will have all bits of azimuth and range set to ones; the other will have all bits of azimuth and range set to zeros. As in other reports, one bit will indicate which radar set is generating the report. One test message is generated at the time the antenna azimuth count is maximum and the other when the azimuth count is zero. The video processor sends a common radar trigger to the radars being used with the video processor. This allows the range sweeps to be synchronized with the Range Counter, and also enables the video processor to control pulse repetition frequency (PRF).

Processed video is also sent to plan-position indicator (PPI) repeaters and to display consoles. This output consists of raw video mixed with censor gate and unused range band information. As the video output may be from two different radars whose antenna are at different azimuths, information from only one radar at a time may be used by the PPIs.

DIGITAL DATA INTRODUCER MX-3195(V)/USQ-20(v)

The Digital Data Introducer MX-3195(V)/USQ-20(v) shown in figure 14-23, is used as a component part of the Naval Tactical Data System (NTDS). The Digital Data Introducer (colloquial name, universal keyset) is one of the equipment types used in the data gathering system (referred to as the keyset complex). Figure 14-20 illustrates the functional relationship of the equipment comprising the keyset complex. Each unit, except those which may vary according to installation site, is identified by both its assigned designation and its colloquial name. The keyset provides a means for manually entering data into the keyset



124.209

Figure 14-23.—Digital Data Introducer, MX-3195/USQ-20(V).

complex for eventual processing by keyset central and a computer.

Visual verification of each data entry is provided by the readout display comprised of seven panel-mounted projection-type readout modules. The readout display reproduces the code words inscribed on the keyboard overlay and/or the decimal digits appearing on the keyboard pushbutton key controls.

Data transfer is initiated by operating the keyset TRANSMIT control. However, actual data transfer does not occur until the computer requests keyset central to interrogate the keyset. During keyset interrogation, keyset central transfers the data from the keyset to its own Data Flow register (not shown) for temporary storage prior to transmission to the computer.

Removable keyboard overlays and readout modules permit the keyset to be adapted for handling various data subjects. Keyset modification consists of fitting a keyset with the keyboard overlay and readout modules which reflect the data being entered on the keyset.

COMPUTER SET CONTROL INTRODUCERS

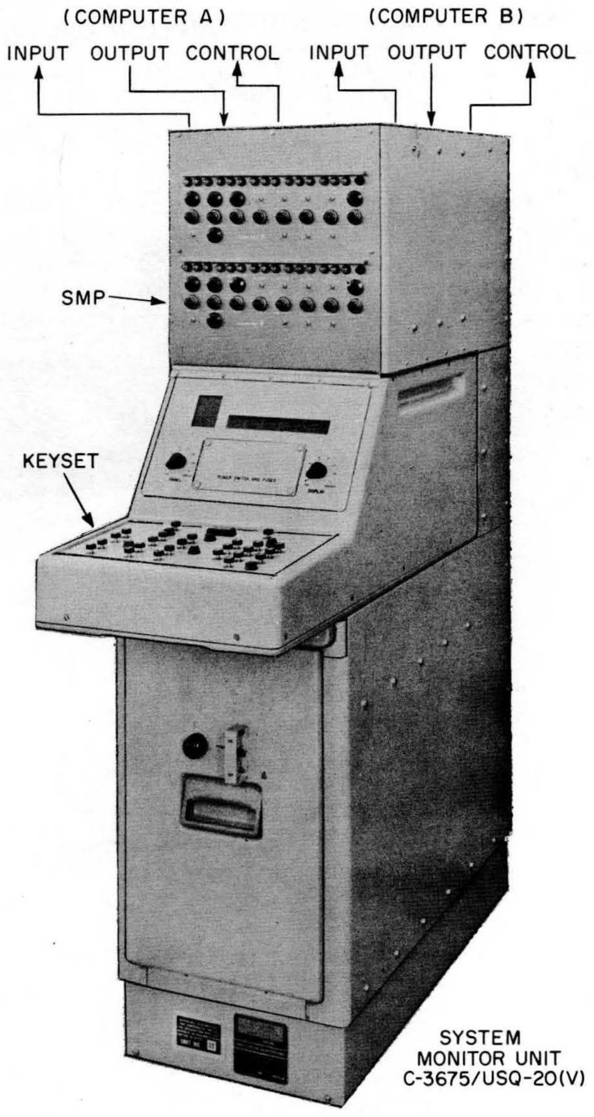
The Computer Set Control-Introducer types C-3675A/USQ-20(v) and C-3674A/USQ-20(v) (fig. 14-24) provide the means whereby one man may conveniently direct and monitor the performance of a Naval Tactical Data System installation consisting of a multi-computer complex and associated peripheral equipment.

The computer Set Control-Introducer occupies a regular NTDS operating position in the Combat Information Center. Type C-3675A/USQ-20(v) is used when the computer complex contains two computers; type C-3674A/USQ-20(v) (not shown) is used when the computer complex contains three computers. The two units are identical in size and operation, the exception being that the C-3674A/USQ-20(v) contains additional logic and control circuits in order to provide the control and monitoring facilities necessary in a three-computer complex.

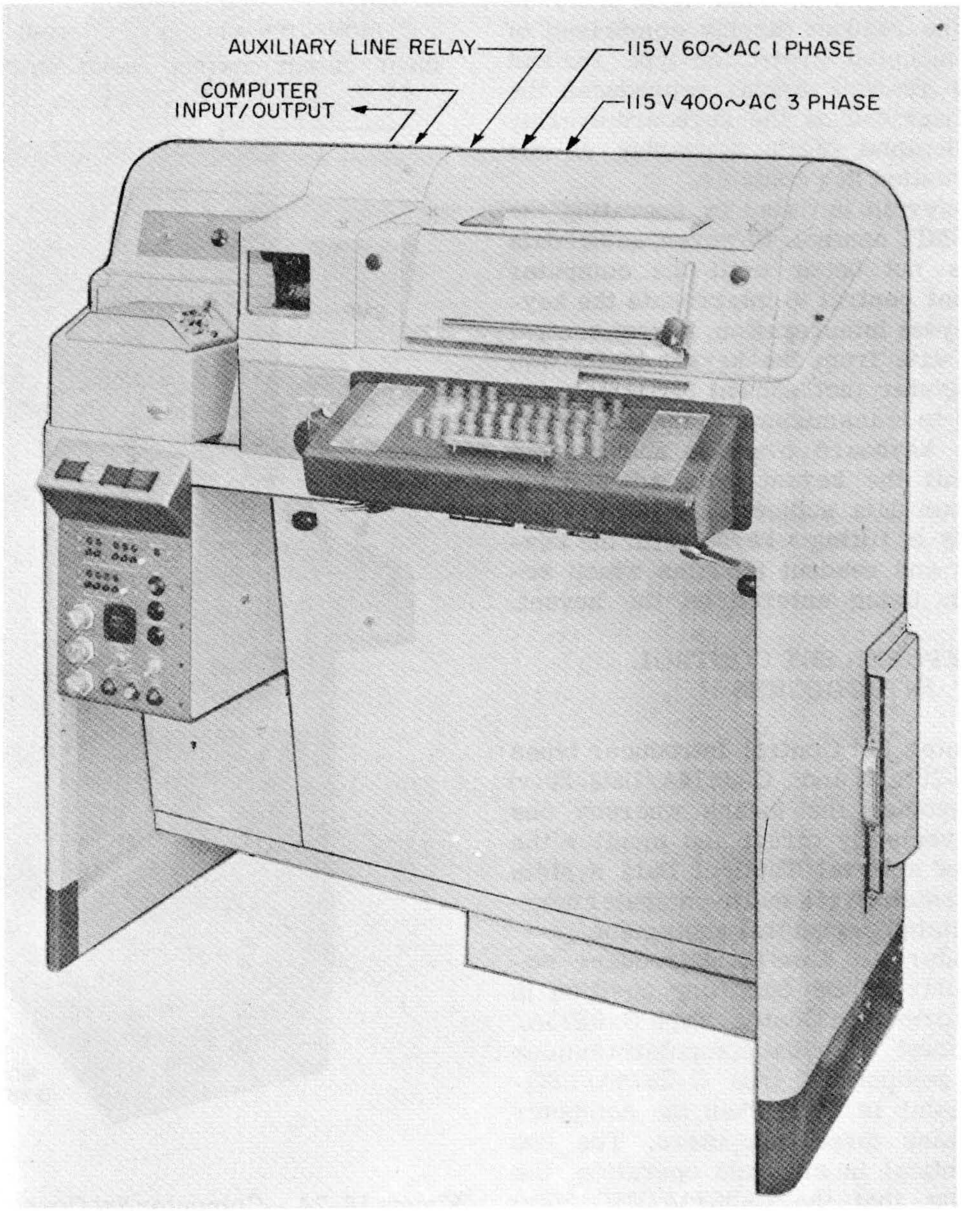
TELETYPEWRITER SET AN/UGC-6 OR AN/UGC-13

The purpose of the Teletypewriter Set AN/UGC-6 or AN/UGC-13 (fig. 14-25) is to provide input/output communications with a selected data processing computer. The teletypewriter set is connected to an interconnection panel (fig. 14-26) which is used to connect the input/output cables of the selected computer to the input/output cables of the teletypewriter set. The adapter contained in the teletypewriter set modifies data to provide compatibility between the computer and the teletypewriter unit.

If required, an auxiliary relay can be connected between the adapter and teletypewriter



124.210
Figure 14-24.—Computer Set Control Introducer (Two Computer), relationship of units



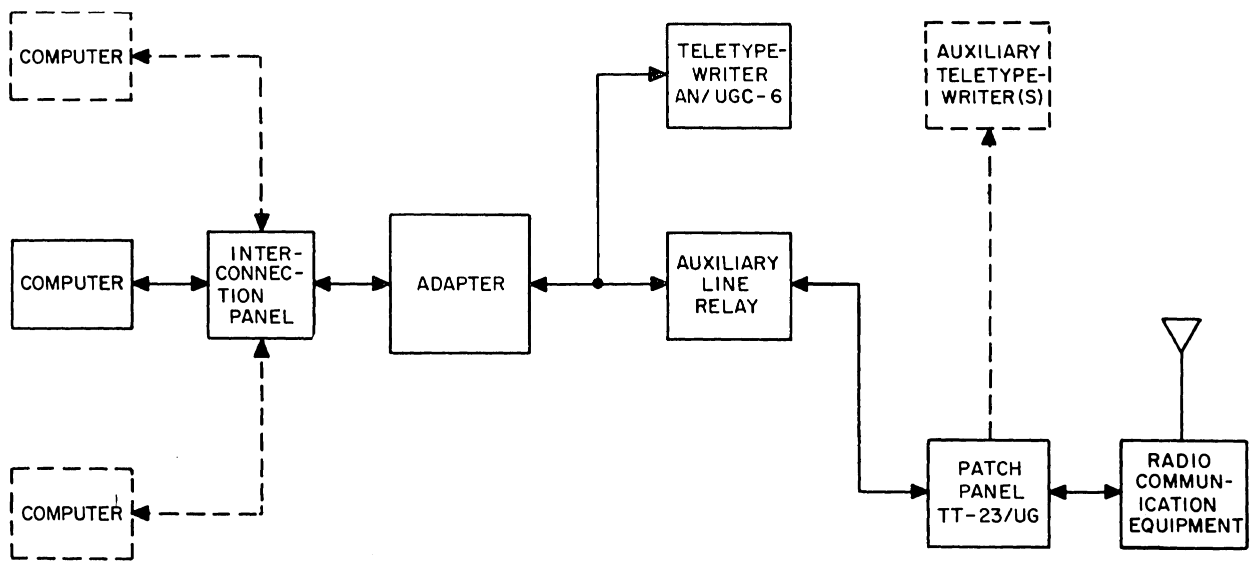
31.29

Figure 14-25.—Teletypewriter Set AN/UGC-6, with adapter

unit. This line relay permits radio link equipment and/or other teletypewriter units to be entered into the system.

DIGITAL-TO-ANALOG CONVERTER GROUP, AN/SYA-3

The Digital-To-Analog Converter Group, AN/SYA-3 (fig. 14-27), sometimes referred to as the IDAC Set (Interconnecting Digital Analog Converter) Mod 1, serves as the communication link between the Naval Tactical Data System and the Designation Equipment Mark 9 (DE Mk 9, discussed later) which is a part of the Weapon Direction System. The Control Indicator Units (which are located near the DE Mk 9 consoles) permit manual insertion of data into IDAC Mod 1



31.29

Figure 14-26.—Teletypewriter system.

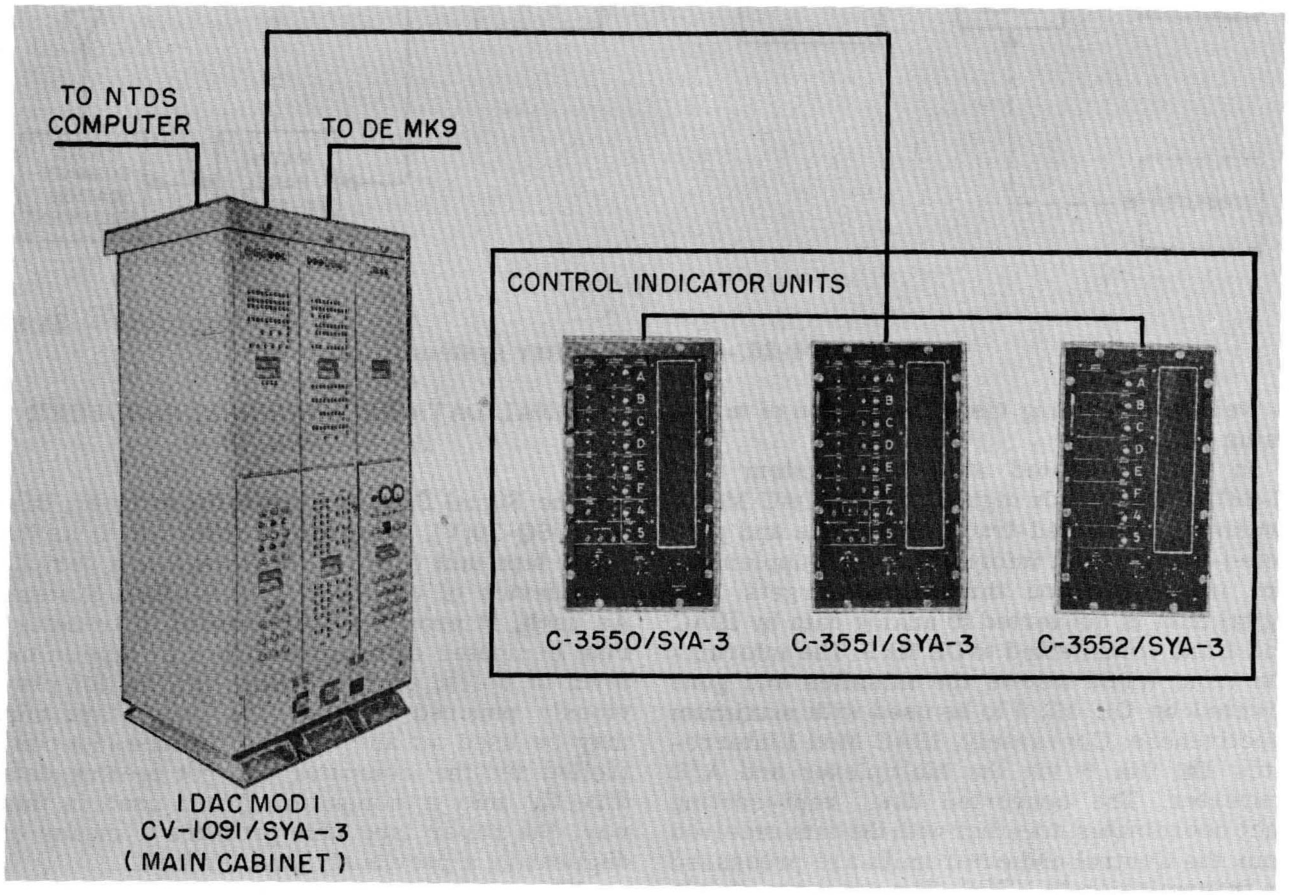
as well as providing visual indication of target status.

In the integrated shipboard system (fig. 14-28), NTDS sends digital data to IDAC Mod 1 containing predicted-track coordinate and predicted-track height information, and supplementary information on threatening targets. The digital data is converted to analog data by IDAC Mod 1 and transmitted to DE Mk 9. The information from NTDS allows the missiles and guns directed by DE Mk 9 to be used with maximum effectiveness. Conversely, IDAC Mod 1 interrogates DE Mk 9 via the Multiplexer and A/D Converter. The converted data, representing tract coordinates, together with the data received from the Control-indicator units are processed for transmission to NTDS. The exchange of data between NTDS and the DE Mk 9 via IDAC Mod1 is accomplished by the computer buffer mode. (A computer buffer consists of a group of data transmission words to or from the computer.) The computer control signals, function codes, interrupt codes, and the IDAC Mod 1 timing circuits govern the two-way flow of data.

SIGNAL DATA RECORDER-REPRODUCER RD-231/USQ-20(V)

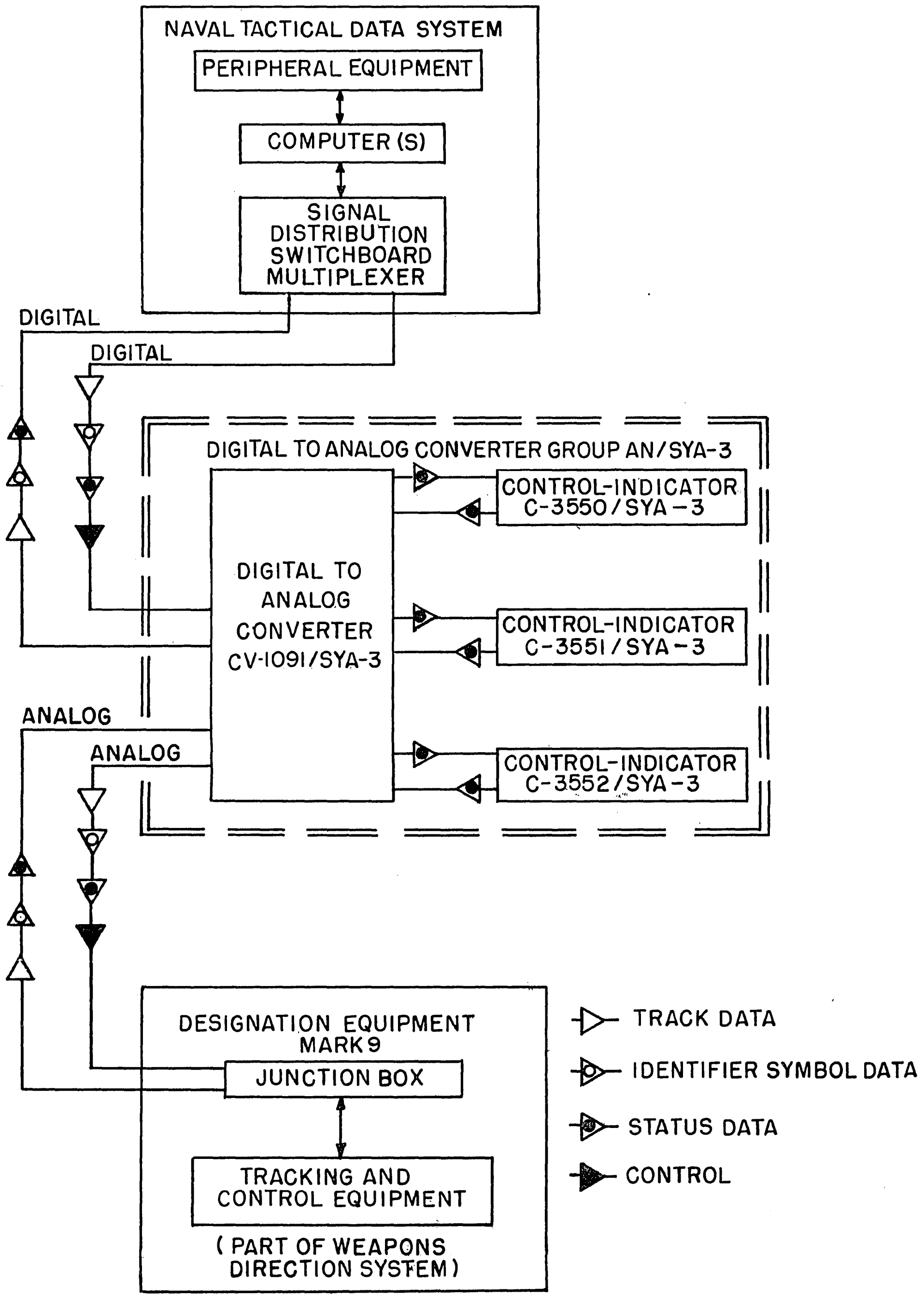
The Signal Data Recorder-Reproducer, RD-231/USQ-20(V), hereinafter referred to as the paper tape unit (fig. 14-29) is used as a peripheral device of the Naval Tactical Data System. As such, it provides both a means of transferring computer data onto a ribbon or tape in the form of holes, or for reading information previously punched on tape. The paper tape unit may be used as an "off-line" device (not controlled by the computer and not feeding data directly into a computer). When used in this way, the paper tape unit provides a means of duplicating a previously punched tape.

The paper tape unit contains all the functional elements of the equipment: a photoelectric reader, a tape winder and reel holder, a manual control panel (all shown in figure 14-29), a high speed punch, logic chassis, and power supply chassis (not shown). The unit also contains a cooling system and a connector panel for input, output, and power cables.



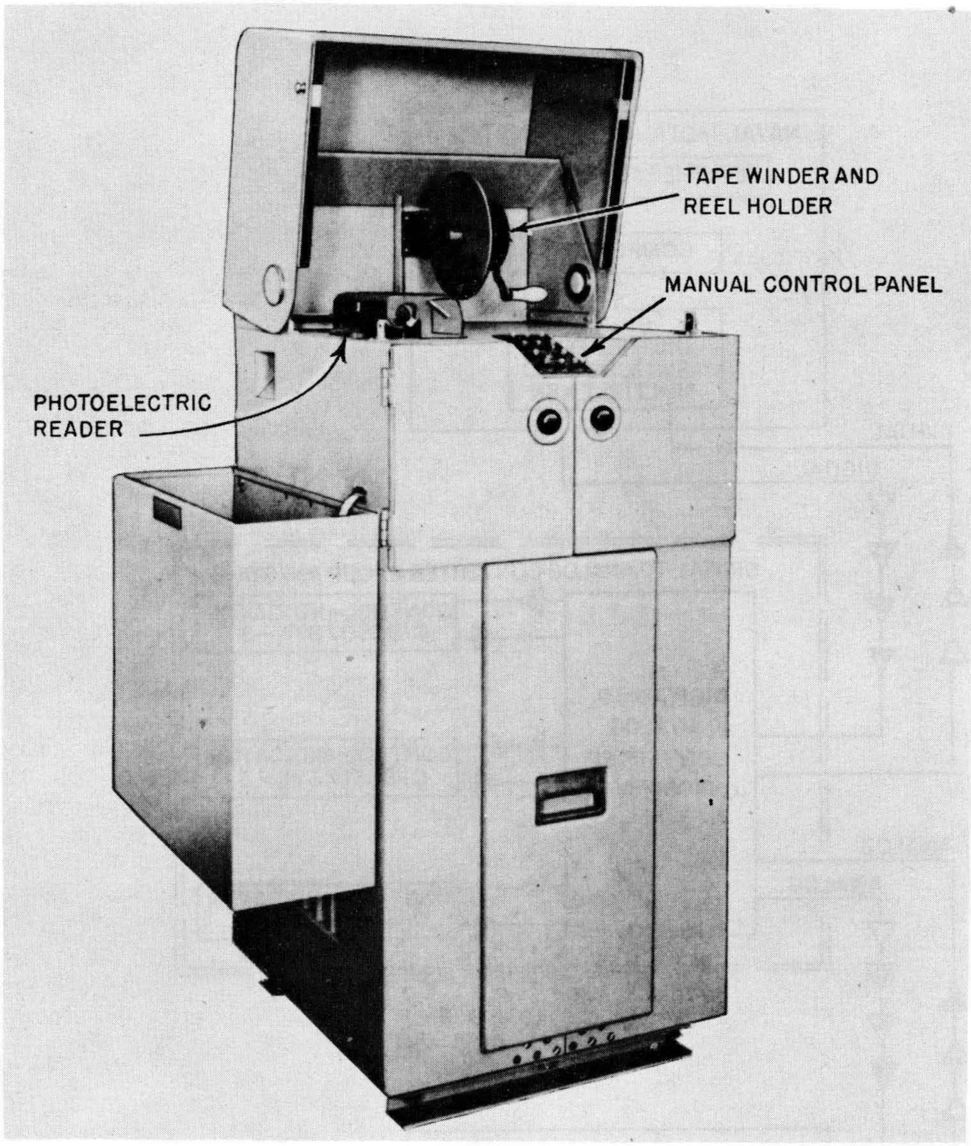
124.213

Figure 14-27.—IDAC Set Mod 1, unit identification.


NAVAL TACTICAL DATA SYSTEM
PERIPHERAL EQUIPMENT
COMPUTER (S)
SIGNAL DISTRIBUTION SWITCHBOARD MULTIPLEXER
DIGITAL
DIGITAL
DIGITAL TO ANALOG CONVERTER GROUP AN/SYA-3
DIGITAL TO ANALOG CONVERTER CV-1091/SYA-3
CONTROL-INDICATOR C-3550/SYA-3
CONTROL-INDICATOR C-3551/SYA-3
CONTROL-INDICATOR C-3552/SYA-3
ANALOG
ANALOG
DESIGNATION EQUIPMENT MARK 9
JUNCTION BOX
TRACKING AND CONTROL EQUIPMENT
(PART OF WEAPONS DIRECTION SYSTEM)
TRACK DATA
IDENTIFIER SYMBOL DATA
STATUS DATA
CONTROL

124.214

Figure 14-28.—Equipment-system integration drawing.



124.215

Figure 14-29.—Signal Data Recorder-Reproducer RD-231/USQ-20(V).

CHAPTER 15

OTHER NAVY COMPUTERS

The Navy is presently using computers in various types of logistical as well as tactical operations. The NTDS Computer (treated in chapters 10 through 14) is used in solving tactical problems. Two general purpose computers (the Control Data 160-A and the Control Data 1604-A) are presented in this chapter as representative equipments which are used in solving logistical problems. The Navy is also making use of computers for information processing in meteorology, oceanography, intelligence and communications.

CONTROL DATA 160-A COMPUTER

The CONTROL DATA 160-A Computer (fig. 15-1) is a small (desk size), highly flexible, stored program computer. Using a high speed data transfer channel, the 160-A may communicate with other computers, allowing multi-computer processing and computer control of computers. Two 160-A computers may share input/output equipment such as magnetic tape stations to assure high computing performance at a minimum cost.

CHARACTERISTICS

The major characteristics of the 160-A are:
- Buffered input/output
- Internal and external interrupts
- An expandable magnetic core memory (8,192 to 32,768 words)
- A 350 character/second paper tape reader
- A 110 character/second paper tape punch (not shown)
- Transistor-diode logic

A 160-A system may be expanded to include the following external equipment:
- Up to 40 magnetic tape stations
- Input/output typewriters
- Punched card readers, punches, and low speed line printers
- High speed (1000 lines/minute) printers
- Plotting and digital display equipment
- Analog-to-digital and digital-to-analog equipment
- Two CONTROL DATA 160-A Computers operating independently but sharing in the same magnetic core memory

The 160-A is a single address, electronic data processor which uses the parallel method to perform data transfer. An internally stored program located in sequential addresses controls operation. The basic memory of the 160-A consists of two banks of magnetic core storage each with a capacity of 4096 12-bit binary words and a storage cycle (memory reference cycle) time of 6.4 μsec.

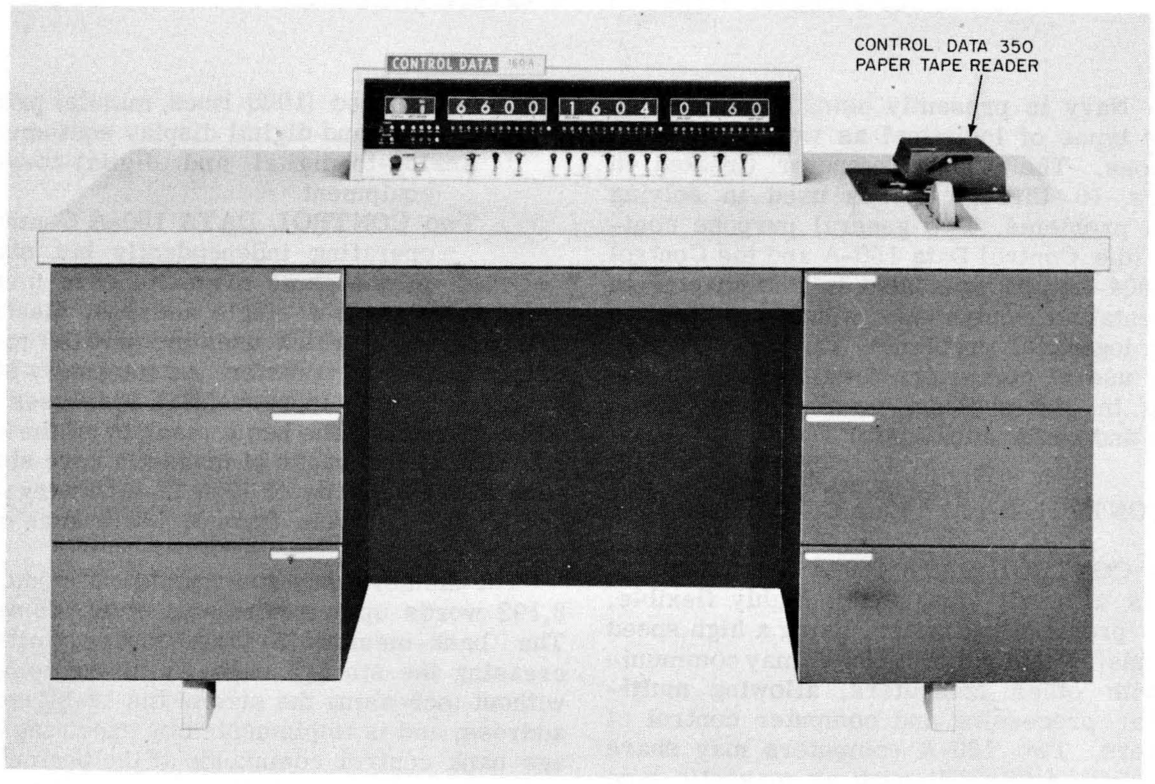
The memory may be expanded in modules of 8,192 words up to a maximum of 32,768 words. The bank-memory arrangement permits increasing the storage capacity of the computer without increasing the size of the 12-bit storage address, and is implemented by executing storage bank control commands at the beginning of a program. Bank selections may be changed at any time during the program. Instructions are executed in one to four storage cycles; the time varies between 6.4 and 25.6 μsec.

An instruction is a 12-bit word bit comprising a 6-bit function or operation code (F) and a 6-bit code extension and/or execution address (E). The average program execution time for 130 instructions is approximately 15 μsec per instruction.

A CONTROL DATA 350 Paper Tape Reader is located in the desk top of the computer to the right of the display console, and a Teletype BRPE-11 paper tape punch is housed in the left side of the computer cabinet (not shown in the diagram).

REGISTERS

The registers (shown in the block diagram of figure 15-2) are the storage units for all data



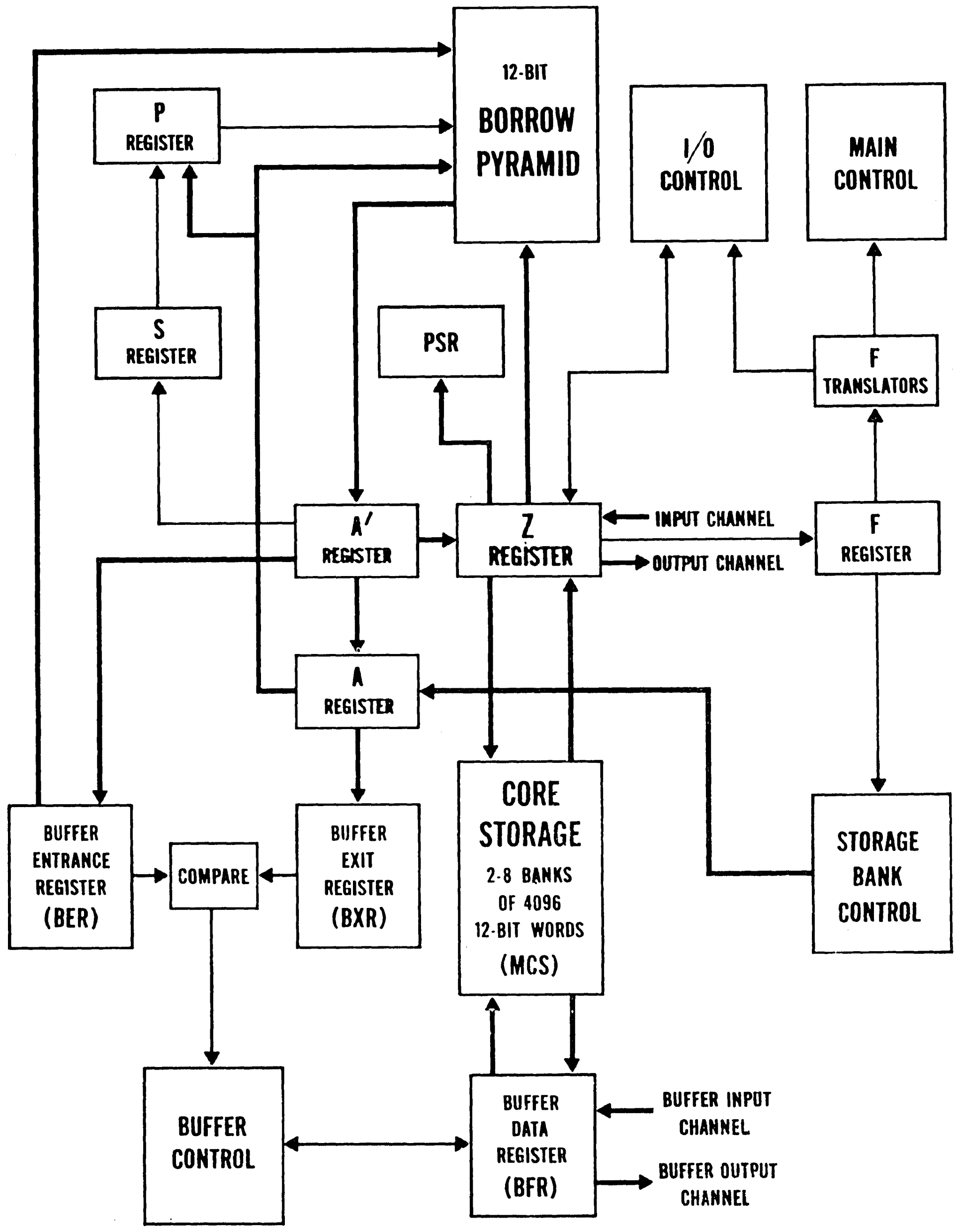
124.216X

Figure 15-1.—Control Data 160A Computer.

transmission and computation in the 160-A. Those registers available to the programmer by means of computer instructions are called addressable registers; the others are called nonaddressable. Unless otherwise specified, all registers are 12 bits long.

Addressable Registers

A Register	The A register receives the results of all arithmetic, logical, and shifting operations.
P Register	The P register contains the storage address of the current instruction. At the completion of an instruction which does not require program control to jump out of sequence, the P register is advanced by 1 or 2 to select the address of the next instruction. If control is to be transferred out of sequence the new control address is sent to the P register.



124.217X

Figure 15-2.—Block diagram of registers.

Buffer Entrance Register (BER)	During buffered input/output operations, BER holds the address to or from which information is being transferred. The contents of this register may be transferred to storage and to the A register.
Buffer Exit Register (BXR)	During buffered input/output operations, BXR holds the last word address +1 to or from which information is being transferred.

Non-Addressable Registers

Z Register	The Z register, (a transient register) holds information during transfers between storage and peripheral equipment on the normal input/output channel. The Z register also restores information read out of storage.
S Register	The S register contains the storage address currently being referenced either for an instruction or for an operand.
F Register	The F register contains the decoded instruction being executed. The upper 6 bits contain the effective function code and, in general, the lower 6 bits the effective E portion of the instruction.
Buffer Data Register (BFR)	During a buffer operation, BFR holds the word of information being transferred to or from storage.
A Register	The A register acts as an auxiliary transmission register for information moving between the other registers. This is the output register of the borrow pyramid (adder).
Punch Storage Register (PSR)	During output operations which use the paper tape punch, the 8-bit PSR releases the Z register, thereby permitting highspeed computation while the output characters are punched at a comparatively slow rate. The contents of PSR cannot be indicated on the display panel.

ARITHMETIC OPERATIONS

A 160-A computer word contains 12 binary digits. The bits within a computer word are numbered from 00 (least significant) to 11, starting on the right (see table 15-1). All arithmetic is binary, one's complement notation (described later). Any number may be represented as combinations of the two binary digits, 0 and 1. Although the computer operates in the binary system, the octal representation of a binary number is more convenient.

The 160-A word can be considered as four octal digits. Table 15-1 shows the bit position numbering, the binary representation of a 12-bit quantity, and the ocatal equivalent of that quantity.

Table 15-1.—Bit Position Numbering.

Bit-position	11	10	09	08	07	06	05	04	03	02	01	00
Binary Representation	0 0	1	1	1	1	0	0	0	1	0	1	1
Octal Representation		3			6			1			3	

In one's complement notation, positive numbers are represented by their binary equivalent; negative numbers are represented by the one's complement of the equivalent positive number. To form one's complement, complement each bit of the word..

Example:

+5 is represented as: 000 000 000 101

-5 is represented as: 111 111 111 010 (complement)

The internal arithmetic of the computer is based on subtraction. Addition is performed by subtracting the complement of the addend from the augend. In subtraction no complementing is necessary.

Example:

The computer is programmed to add +6 to +5. This operation is performed as follows:

	+5 =	000 000 000 101 augend
	+6 =	000 000 000 110 augend
Complementing +6 produces and subtraction takes place,		111 111 111 001
	+5 =	000 000 000 101
	-(+6) =	111 111 111 001
Borrow	(1)	000 000 001 100
Subtract the borrow	-	1
produces		000 000 001 011 = 13_8

During the subtraction process, the borrow from the high order end is carried around and subtracted from the low order end of the word to provide the corrected result. This "end-around-borrow" is the feature which makes the arithmetic of the 160-A modules 2^{12}-1.

The value zero can be represented in one's complement notation in either of two separate expressions:

000 000 000 000 (plus zero)
111 111 111 111 (minus zero)

Both plus and minus zero are acceptable as arithmetic operands. There are only two cases in which a zero arithmetic result will be minus zero; in all other cases, the result will be plus zero. These two cases are:

-0 + (-0) and -0 -(+0)

The 2^{11} bit position is used for sign-bit-notation. All positive numbers must have a "0" in bit position 11; all negative numbers must have a "1" in bit position 11. As shown in the illustrations above, the sign bit is extended to the most significant bit of the number, i.e., all bits to the left of the absolute value of the number are the same as the sign-bit.

STORAGE CONTROL

Magnetic core storage (not shown) in the 160-A is composed of two, four, six or eight separate banks of cores. Each bank contains 4,096 12-bit words. A minimum memory contains 8,192 words and is expandable in modules of 8,192 words to a total of 32,768 words.

Each bank is assigned a number, from 0 through 7, which is never changed. Associated with these banks is a set of four 3-bit registers called storage bank controls. These controls are relative (r), indirect (i), direct (d) and buffer (b). Each control may be set by a computer instruction to reference any one of the eight possible banks.

Each bank has its own set of 10000_8 addresses numbered from 0000 to 7777. When a bank control is set so that it references a nonexistent bank, the computer will stop with a fault indication if an instruction is executed which must reference that particular nonexistent bank.

The parenthesis, ( ), is used to identify the particular storage bank control. The letter (r, i, d or b) shown within the parenthesis specifies the register or location in which the desired contents are stored.

WORD FORMAT

As aforestated, a 160-A instruction is divided into a 6-bit function code (F) and a 6-bit execution address (E). Most instructions require only one word of storage, but certain expanded instructions are available which occupy two words of storage. The format of two-word instructions is always as shown in table 15-2; the first word contains a 6-bit function code and a 6-bit execution address. The execution address is always equal to zero. The second word contains a 12-bit address or operand (G), depending on the instruction. Words 1 and 2 must be located in sequential storage addresses of the same bank.

Table 15-2.—Format of Two-Word Instructions.

	Function Code (F)	Execution Address (E)
Word 1	6 Bits	0 0 0 0 0 0
Word 2	12 Bits	

For all instructions the function code F and in some cases the execution address E, determines which instruction in the 160-A repertoire will be executed. Because E contains only 6 bits, it is not possible for E to specify a complete storage address in all cases. Therefore, depending on F, the computer selects, for each instruction, one of nine addressing modes.

ADDRESSING MODES

The nine addressing modes used by the 160-A computer provide maximum flexibility of the 6-bit address. A brief description of the actions performed by each of the addressing modes follows.

No Address (N)	When the E portion of the instruction is used as a 6-bit operand, it performs arithmetic and logical operations with a 6-bit constant contained in the instruction. This mode eliminates the need for entering many constants into memory.
Direct Address (D)	Refers to a 12-bit operand in one of the first 64 (100 octal) storage locations, of the direct bank (d).
Indirect Address (I)	Provides for operand references and jump addresses. Instructions employing indirect addressing use E to refer to one of the first 100 octal storage locations of the direct bank (d). The contents of this address are used as the address of the operand or as the jump address.
Relative Address Forward (F)	Operand or jump address is obtained by adding E to the current contents of P to specify one of the 63 (77 octal) addresses immediately following the address of the current instruction.
Relative Address Backward (B)	Operand or jump address is obtained by subtracting E from the current contents of P to specify one of the 63 (77 octal) addresses immediately preceding the address of the current instruction.
Forward Indirect Addressing (F)	Certain instructions combine relative forward and indirect addressing. E, when added to P, produces a sum which specifies the location of the operand address or jump address. This indirect address cannot be further than 63 locations ahead of the current instruction. The operand or jump address may be located anywhere within storage. P remains unchanged.
Specific Address (S)	The effective address is always location 7777 (octal) in storage bank zero. The E portion of the instruction word is always equal to zero in the specific mode.

The two modes (C and M described below) use two sequential storage locations, the second of which is designated G. A prerequisite for these modes is that E must always be equal to zero.

Constant Address (C)	The G portion of the 24-bit instruction word contains the operand.
Memory Address (M)	The G portion of the 24-bit instruction word contains the address of the operand, in the indirect (i).

CONTROL DATA 1604 COMPUTER

The CONTROL DATA 1604-A (fig. 2-1) is a stored-program, general purpose digital computer with a large storage capacity, fast computation and transfer speeds, and special provisions for input/output communication. The

1604-A is designed to handle large-volume data processing and to solve large-scale scientific problems. The computer circuits are constructed of solid-state components.

1604-A CHARACTERISTICS

Some of the major characteristics of the computer are listed below.

Stored-program general purpose digital computer

Parallel mode of operation

48-bit word, 2 instructions per word (described later)

Single address logic (Per instruction)

Operation code	6 bits
Designator	3 bits
Base Execution Address	15 bits

Six 15-bit index registers

Indirect addressing

Magnetic core storage 32,768 48-bit words

Two independent 16,384-word banks

4.8 μsec effective memory cycle time (representative program)

6.4 μsec total memory cycle time

Input/output
- Parallel transmission of 48-bit words
- Three separate buffer input channels
- Three separate buffer output channels
- High-speed transfer channel (4.8 μsec per word)

Program interrupt

Console, includes:
- Photo-electric paper tape reader
- Paper tape punch (not shown)
- Electric typewriter
- Register contents displayed in octal

Binary arithmetic
- Parallel addition in 1.2 μsec
- Modulus $2^{48} - 1$ (one's complement)

Real-time clock

Completely solid-state
- Diode logic
- Transistor amplifiers

LOGICAL DESCRIPTION

The 1604-A performs calculations and processes data in a parallel binary mode through the step-by-step execution of individual instructions which are stored internally along with the data.

Functionally, the computer may be divided into four major sections (storage, control, arithmetic and input/output). The Storage section provides internal storage for data and instructions; the Control section coordinates and sequences all operations for executing an instruction by obtaining the instruction from storage and translating it into commands for the other sections; the Arithmetic section performs the arithmetic and logical operations required for executing instructions; and the Input/Output section provides communication between the computer and the external equipment.

The registers in the computer are identified by letters. The operational registers are those registers which have their contents displayed on the console and may be changed manually. These registers are identified by the asterisks in table 15-3. They usually hold the end result of an operation. The arithmetic properties of the registers are discussed later.

Table 15-3.—Registers of the Computer.

A*	Accumulator
Q*	Auxiliary Arithmetic
B^1 through B^6*	Index registers (six)
P*	Program Address
U^1*	Program Control
U^2	Auxiliary Program Control
R.	Address Buffer
CCR CR^1 through CR^6	Buffer Control
X	Exchange

*Can be changed manually.

STORAGE SECTION

The magnetic core storage section of the 1604-A computer provides high-speed, random access storage for 32,768 words. It consists of two independent storage units (not shown) each with a capacity of 16,384 words. These units operate together during the execution of a stored program and thus are considered as one 32,768-word storage system.

A word is 48 bits in length and is used in two ways: as two 24-bit instructions or as a 48-bit operand (data word). The location of each word in storage is identified by an assigned number or address. When a word is taken (read) from or entered (written) into storage, a reference is made to the storage address which holds the word. All odd storage addresses are located in one of the independent storage units, while all even addresses are located in the other.

The cycle time, or time for a complete storage reference, is 6.4 μsec. Since the storage cycles of the two sections overlap one another in the execution of a program, the average effective cycle time for random addresses is about 4.8 μsec.

CONTROL SECTION

The control section directs the operations required to execute instructions and to initiate the exchange of data with external equipment. It also establishes the timing relationships needed to perform the operations in the proper sequence.

The control section acquires a program word from storage, interprets it and sends the necessary commands to other sections. A program word (table 15-4) is a pair of 24-bit instructions (only one 24-bit word shown in the table) which together occupy one storage location as a 48-bit word. The higher order 24 bits are the upper instruction; the remaining 24 bits, the lower instruction.

Table 15-4.—Instruction Format (One-Half Program Word)

f (6 bits)	b (3 bits)	m, y, or k (15 bits)
Operation code	Index Designator	Base Execution Address

There are provisions for 77_8 (64_{10}) instructions in the 1604-A repertoire (see table 15-5). Two of these instructions (00 and 77) are not used. Thus, 62_{10} different instructions are capable of being performed. Each of the 62 instructions has a unique 6-bit operation code (see table 15-4) which specifies the operation or particular instruction to be performed.

The index designator (b) generally specifies one of the six index registers whose content is to be added to the execution address. This process is called "address modification." However, the index designator may also specify indirect addressing or a condition for jump and stop instructions.

The base execution address may be used in one of three ways: as an address, m, of an operand; as an operand, y; or as a shift count k.

The eight operational registers in the control section (see table 15-3) are P, U^1 and B^1 through B^6. The program address register (P) is a two's complement additive counter. It provides program continuity by generating in sequence the storage addresses which contain the individual program steps. Usually at the completion of each two instructions the count in P is advanced by one to specify the address of the next program word.

The program control register (U^1) holds a program word while the two instructions contained in it are executed. The upper instruction is executed first followed by the lower instruction.

After executing an instruction, a half exit, full exit, or jump exit is performed. A half exit allows the lower instruction of a program word to be executed. A full exit advances the count in P by one and executes the upper instruction of the new program word specified by the contents of P. A jump exit allows a new sequence of instructions to be executed; the storage location of the new instruction is specified by the execution address of the jump instruction. The execution address, in this case, is entered into P and specifies the starting location of a new sequence of program words.

The auxiliary program control register (U^2) is a 15-bit subtractive accumulator used primarily in the modification of the base execution address. The contents of the specified index register are transmitted to the Address Buffer register (R), which has provisions for counting, complementing and storing. The contents of R are then added to the contents of U^2 which holds the execution address.

Table 15-5.—1604-A Repertoire of Instructions.

00	ZRO	(not used)	40	SST	Selective Set
01	ARS	A Right Shift	41	SCL	Selective Clear
02	QRS	Q Right Shift	42	SCM	Selective Complement
03	LRS	AQ Right Shift	43	SSU	Selective Substitute
04	ENQ	Enter Q	44	LDL	Load Logical
05	ALS	A Left Shift	45	ADL	Add Logical
06	QLS	Q Left Shift	46	SBL	Subtract Logical
07	LLS	AQ Left Shift	47	STL	Store Logical
10	ENA	Enter A	50	ENI	Enter Index
11	INA	Increase A	51	INI	Increase Index
12	LDA	Load A	52	LIU	Load Index, U
13	LAC	Load A, Complement	53	LIL	Load Index, L
14	ADD	Add	54	ISK	Index Skip
15	SUB	Subtract	55	IJP	Index Jump
16	LDQ	Load Q	56	SIU	Store Index, U
17	LQC	Load Q, Complement	57	SIL	Store Index, L
20	STA	Store A	60	SAU	Substitute Address, U
21	STQ	Store Q	61	SAL	Substitute Address, L
22	AJP	A Jump	62	INT	Input Transfer
23	QJP	Q Jump	63	OUT	Output Transfer
24	MUI	Multiply Integer	64	EQS	Equality Search
25	DVI	Divide Integer	65	THS	Threshold Search
26	MUF	Multiply Fractional	66	MEQ	Masked Equality
27	DVF	Divide Fractional	67	MTH	Masked Threshold
30	FAD	Floating Add	70	RAD	Replace Add
31	FSB	Floating Subtract	71	RSB	Replace Subtract
32	FMU	Floating Multiply	72	RAO	Replace Add One
33	FDV	Floating Divide	73	RSO	Replace Subtract One
34	SCA	Scale A	74	EXF	External Function
35	SCQ	Scale AQ	75	SLJ	Selective Jump
36	SSK	Storage Skip	76	SLS	Selective Stop
37	SSH	Storage Shift	77	SEV	(not used)

Index registers B^1 through B^6 are 15-bit registers used to modify the base execution address when relative addressing is used. The index registers are also used to designate the number of words in search and transfer instructions and for other indexing operations.

ARITHMETIC SECTION

The arithmetic section of the 1604-A computer consists of two operational registers, A and Q, and one secondary or transient register, X (not displayed on the console). The arithmetic registers (except for the P-register) use the subtractive process for performing arithmetic.

The result of an arithmetic operation in A satisfies $A \leq 2^{47} - 1$ since the contents of A are always treated as a signed quantity. The 2^{47} bit position is used to designate the sign: either positive (if the bit value is 0) or negative (if the bit value is 1). When the result in A is zero, it is always represented by 000...00 (the highest order 0 represents the sign bit and the rightmost 0 represents the lowest order bit), except when 111 . . . 11 is added to 111 . . . 11 or 000 . . .00 is subtracted from 111 . . . 11. (The arithmetic section uses the subtractive process.) Thus when adding, the subtrahend is 1's complemented once, whereas for subtraction the subtrahend is 1's complemented twice and therefore appears not to have been complemented at all. The results in these cases is 111 . . . 11 (negative zero).

The A register (accumulator) is the principal arithmetic register. Some of the more important functions of A are:

1) Arithmetic operations—A register initially holds one of the operands in addition, subtraction, multiplication, and division. The result is usually held in A.

2) Shifting—The contents of the A register may be shifted to the right or left separately or in conjunction with the Q register. Right shifting is open-ended; the lowest bits are discarded and the sign extended. Left shifting is circular; the highest order bit appears in the lowest order stage after each shift; all other bits move one place to the left.

3) Control for conditional instructions—The A register holds the word which conditions jump and search instructions.

The Q register is an auxiliary arithmetic register and is generally used in conjunction with the A register. The principal functions of Q are:

1) Providing temporary storage for the contents of the A register while the A register is being used for other arithmetic operations.

2) Forming a double-length register, AQ or QA.

3) Shifting to the right or left, separately or in conjunction with A.

4) Participating with the A register in multiplication, division, and logical product operations (masking).

The X (exchange) register is used in the exchange of data between storage and the arithmetic section. X provides one of the inputs to the accumulator borrow pyramid (not shown).

INPUT/OUTPUT

The input/output section controls the flow of data to and from the computer. Data is transmitted in one of two ways: High Speed Transfer or Buffering. High speed transfer operations are controlled directly by the program and are used to transfer data between computers or between a 1604-A and high speed external equipment (e.g., a line printer). Buffering is an asynchronous

Table 15-6.—Arithmetic Properties of Registers.

Register	No. of Stages	Modulus	Complement Notation*	Arithmetic	Result
A	48	2^{48}-1	one's	subtractive	signed
Q	48	2^{48}-1	one's		signed
U^2	15	2^{15}-1	one's	subtractive	signed
P	15	2^{15}	two's	additive	unsigned
R	15	2^{15}	two's	subtractive	unsigned

transmission of data on I/O channels 1 through 6. Once a buffer operation has been initiated by the program, buffering and program operations proceed concurrently. Computation continues while buffering takes place at a rate dependent on the external equipment. Buffering and program operations share access to computer storage; buffer operations have priority.

The input buffer channels are the odd-numbered channels, 1, 3, and 5. The output channels are the even-numbered channels, 2, 4, and 6. Channel 7 is used for high-speed input/output data transfer (generally between computers) in the execution of a 62 or 63 instruction (see table 15-5).

Input and output buffer operations are controlled by a section of the control unit called buffer control. The main control of the control unit directs the interpretation of instructions, directs the execution of the instruction, and establishes the timing signals necessary to perform the instructions in the proper sequence. When an exchange of data with external equipment is requested, main control initiates the exchange and then gives control to buffer control. Main control is then free to continue executing other instructions while data exchange is taking place.

Buffer control accepts control of data exchange operations, supervises the exchange, and signals main control when the operation is completed.

A brief description of the main parts of buffer control is given below:

Auxiliary sequence (AUX)—times and initiates the commands of buffering.

High speed storage sequence (HSS)—times and controls the special operations necessary for referencing storage locations 00001-00006.

Auxiliary scanner—the scanner is used to assure that each buffer channel has equal priority in buffering information.

Control Registers (CR) 1-6—30-bit FF registers used to store the address portions of storage locations 00001-00006 (buffer control words). The upper 15 bits (CR_U) hold the current address of a buffer operation. The lower 15 bits (CR_L) hold the terminal address.

Common control register (CCR)—a 30-bit FF register used to "update" the control word after each word is buffered.

Comparator—the comparator is used to compare the upper and lower address portion of the control words to sense when the buffer operation is complete. Input to the comparator is from the CCR.

Inverter banks I^7 and I^8—used to provide more transmission paths in the buffer control section.

U^2, P^1, and CCR_U translators—these translators sense when a storage reference is made to 00001-6. The translator outputs are used as gating conditions.

CHAPTER 16

TEST EQUIPMENT

Test equipment for the computer and its peripheral equipment is much the same as for other electronic equipments. This chapter thus begins with a description of the test equipment common to all electronic equipments. The latter section will describe the test equipments that are designed to test a specific component or components of the computer or its peripheral equipment.

MULTIMETER AN/PSM-4A

Multimeter AN/PSM-4A (fig. 16-1) is a portable volt-ohm-milliammeter. It can be used to make d-c current, d-c resistance, and d-c, a-c, and output voltage measurements. The complete unit includes test probes which may be used with their prod tips, or the tips can be fitted with alligator clips or with a telephone plug to simplify all contact arrangements and connections. A high voltage probe is also included, which makes it possible to read voltages up to 5000 VDC. This probe contains a warning light to indicate the presence of high voltage.

OPERATION

Multimeter AN/PSM-4A is designed to make the following electrical measurements:

(1) Direct currents up to 10 amperes.
(2) Resistances up to 30 megohms.
(3) D-c voltages up to 5000 volts.
(4) A-c voltages up to 1000 volts (RMS)
(5) Output voltages up to 500 volts (RMS).

The following discussion and the associated block diagrams treat the general circuit within the instrument part of the multimeter as it is arranged when used to measure voltage, current, or resistance.

D-C VOLTMETER CIRCUITS

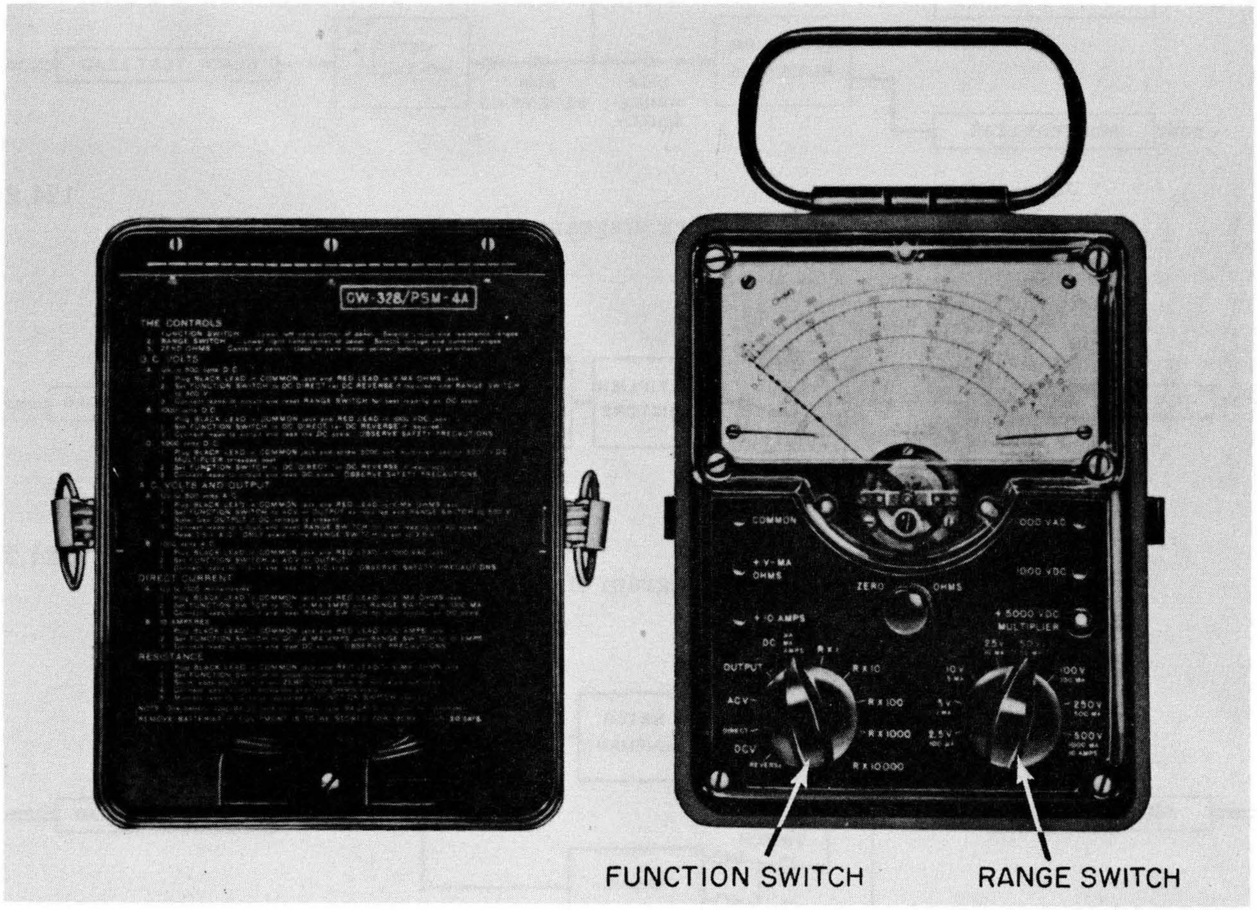
The block diagram of the circuit in Multimeter AN/PSM-4A which is used for measuring d-c voltages is shown in figure 16-2. The circuit is selected with function switch S101, in either its DIRECT or REVERSE DCV position (see figure 16-1). For voltages up through 500 volts, a range is selected with range switch S-102 (only one position shown in figure 16-2). For the 1000 volt range, the red test lead connects into the special 1000 VDC jack (fig. 16-1), and the range switch is not in the circuit. For the 5000-volt range the high voltage probe (not shown) connects the special 5000 VDC jack, and places its resistance in series with the meter movement. For any range, the total resistance in series with it will regulate the meter current to provide a proportional current to indicate the amount of voltage in the circuit.

A-C AND OUTPUT VOLTAGE CIRCUITS

The circuits which measure a-c and output voltages (fig. 16-3), are selected with the ACV and OUTPUT positions of function switch S-101. For voltages up through 500 volts, a range is selected with range switch S-102. For the 1000 volt range, the red test lead connects the special 1000 VAC jack, and the range switch, S-102, is not in the circuit. The a-c voltage impressed across the circuit between the red and black test lead sends current through the resistance of the circuit in both directions, but the rectifier allows only one direction of current flow through the meter movement. The meter is calibrated to indicate the RMS value of the a-c voltage applied to the instrument circuit.

D-C CURRENT CIRCUITS

The circuit which measures d-c currents (fig. 16-4) is selected with the d-c uA MA AMPS position of function switch S-101. For currents up to 1000 milliamperes, the range is selected with range switch S-102. For the 10 ampere range, the red test lead connects the special 10 AMPS



4.133

Figure 16-1.—Multimeter AN/PSM-4A, front view with cover open.

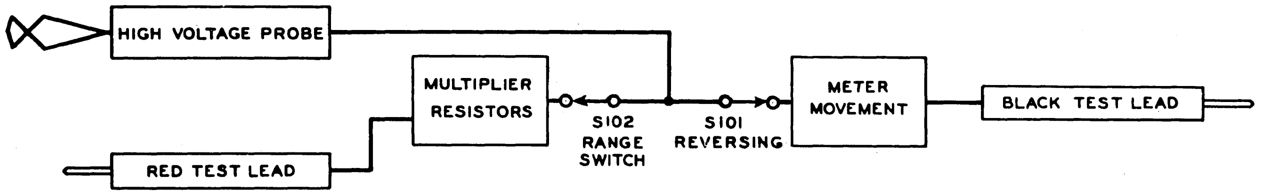
jack, and range switch S-102 is not in the circuit. Each range provides a parallel shunt resistance for the meter movement, and the circuit current divides between these two parallel paths. The proportional part which passes through the meter movement indicates the total circuit current.

OHMMETER CIRCUITS

The ohmmeter circuit (fig. 16-15) and its ranges are selected with function switch S-101. It's positions are Rx1, Rx10, Rx100, Rx1000, and Rx10000. An internal battery furnishes the power for all resistance measurements. For each range, the circuit is arranged so the meter will indicate zero, ohms, and full scale deflection whenever the red test lead and the black test lead are shorted together. When you connect a resistance between the test leads, this resistance will be in series with the instrument circuit, and less current will flow through the meter movement. The amount of reduced meter deflection indicates how much resistance is between the test leads.

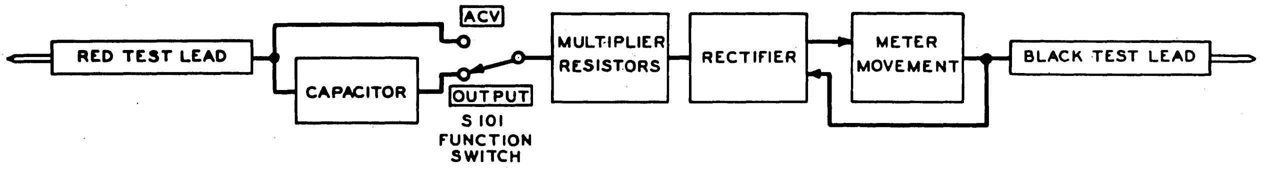
APPLICATION

Function switch S-101, (fig. 16-1) located in the lower left-hand corner of the front panel, selects the type of circuit for which the instrument is connected. There are two positions for d-c volts: DIRECT AND REVERSE. The normal position is DIRECT. When using the meter to make a d-c voltage measurement and a connection is made which causes the meter to read



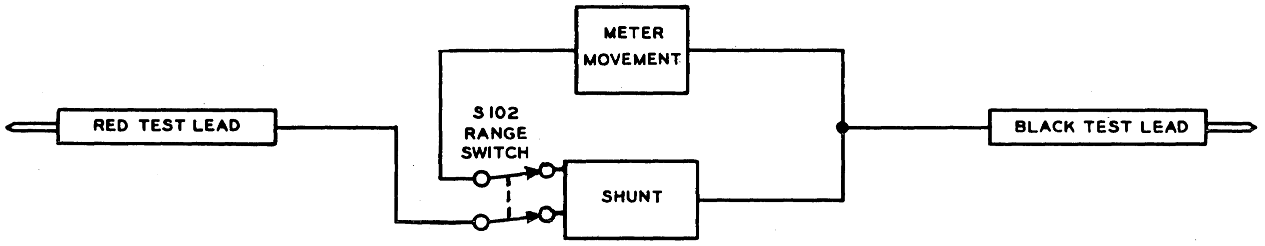
124.219

Figure 16-2.—Functional block diagram of d-c voltage circuits.



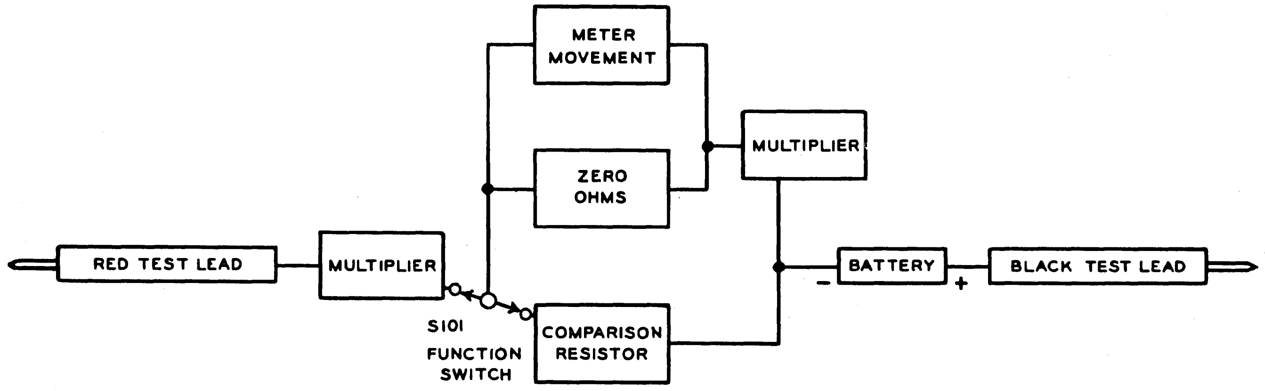
124.220

Figure 16-3.—Functional block diagram of a-c and output voltage circuits.



124.221

Figure 16-4.—Functional block diagram of d-c current circuits.



124.222

Figure 16-5.—Functional block diagram of ohmmeter circuits.

backwards (deflection of the pointer to the left), set switch S-101 to REVERSE and the pointer will be deflected up-scale. Set switch S-101 at ACV to read alternating current voltages. A rectifier within the instrument will change the a-c voltage to an equivalent d-c value and apply this to the meter. The instrument will indicate the RMS value of the applied voltage. Set switch S-101 at OUTPUT to read the a-c portion of mixed a-c and d-c voltages. Set switch S-101 at d-c uA MA AMPS to read direct current. Set switch S-101 at Rx1, Rx10, Rx100, Rx1000, or Rx10000 to read resistance. Switch S-101 also serves as a range switch for resistance measurements.

RANGE SWITCH S-102

This eight-position range switch located in the lower right corner of the front panel permits the selection of voltage and current ranges. The full scale value for each range switch position is marked on the front panel.

ZERO OHMS CONTROL

The ZERO OHMS control is located near the center of the front panel. Each time the function switch S-101 is placed in a position to read resistance, short the test leads together and rotate the ZERO OHMS control knob to make the pointer read full scale, or zero ohms. If you cannot bring the pointer to full scale, replace the battery in the rear of case.

TEST LEADS AND TEST JACKS

There are two test leads, W-101 and W-102, (fig. 16-6) which are needed for all measurements except those which require the 5000 VDC range. Test lead W-101 is red and test lead W-102 is black. Unless otherwise specified, connect black test lead W-102 in the COMMON jack, J106, and connect the red test lead W-101 in the +V MA OHMS jack, J101. For the 1000 VDC range, place red test lead W-101 in the 1000 VDC jack, J-103. For the 1000 VAC range, place red test lead W-101 in the 1000 VAC jack, J-104. Use the red test lead to contact the positive side of the source for d-c measurements and the black test lead to contact the negative side. For the 5000 VDC range, use black test lead, W-102 in the COMMON jack, J-106, and use the high voltage test lead, W-103, screwed on over recessed post J-102, +5000 VDC MULTIPLIER. For the 10 ampere range, place red test lead W-101 in the special +10 amps jack, J-105.

ACCESSORIES E-101, E-102, AND E-103

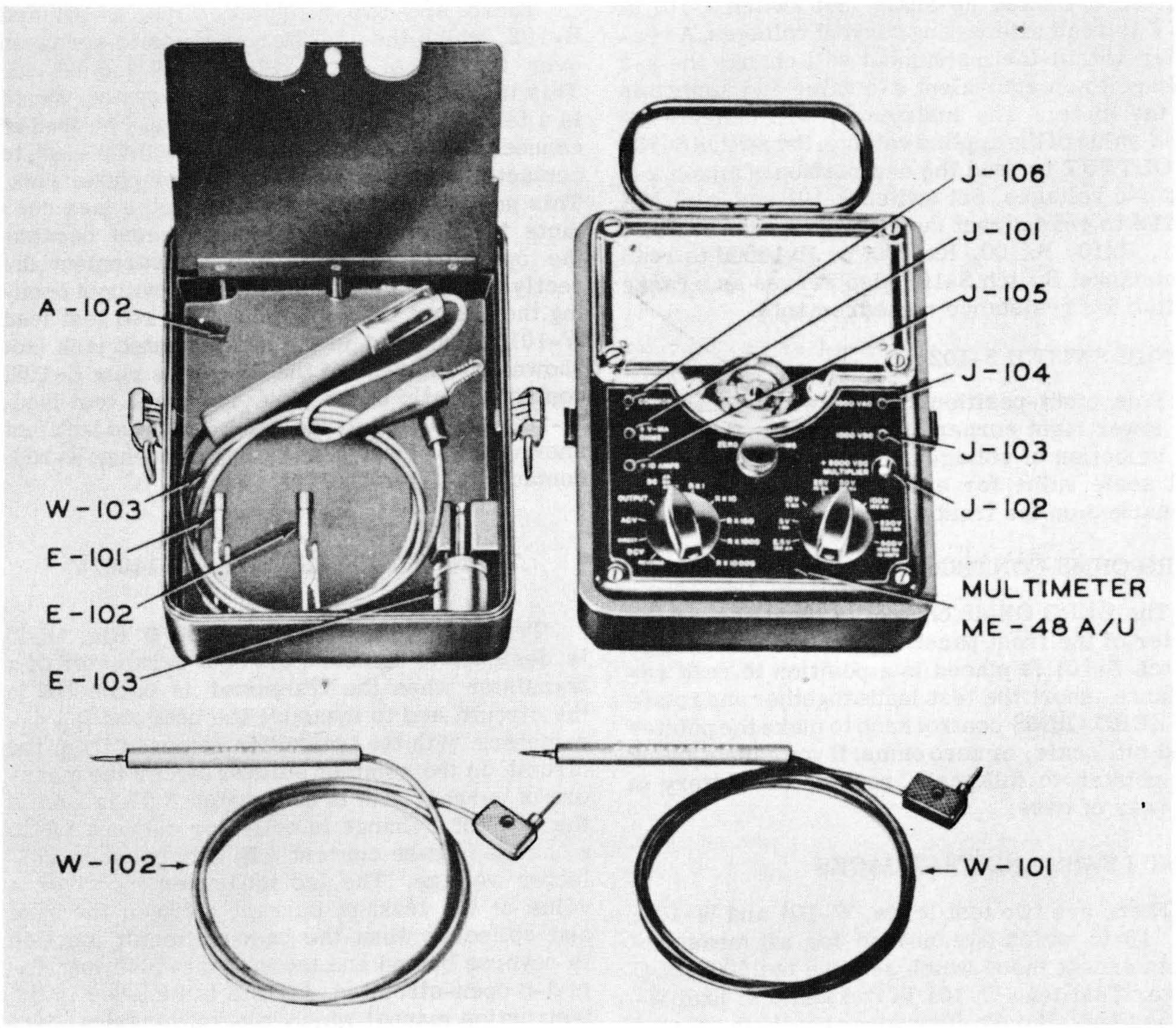
There are two alligator clips, E-101 and E-102, which the operator may use to screw on over the end of test leads W-101 and W-102. This is for the convenience of the operator. There is a telephone plug, E-103, which may be used to connect both the test leads, W-101 and W-102, to contacts within a two-contact telephone jack. This permits easier connection to the jack contacts for any electrical measurement because the operator can make the measurement directly through an equipment panel without opening the case of the equipment. The red test lead W-101, connected in the red insulated jack (not shown) on the rear of the telephone plug E-103, contacts the tip of the plug. The black test lead, W-102, connected in the black insulated jack (not shown) on the rear of the telephone plug, E-103, contacts the sleeve of the plug.

TRANSISTOR TEST SET, TS-1100/U

Transistor Test Set, TS-1100/U (fig. 16-7) is designed to measure the beta parameter of a transistor when the transistor is connected in the circuit, and to measure the beta and I_{co} parameters with the transistor removed from the circuit. In the common emitter circuit the measure of current gain is designated β (Beta) and is the ratio of a change in collector current ΔI_c to a change in base current ΔI_b with constant collector voltage. The Ico measurement gives a value of the leakage current between the base and collector when the base-collector junction is reverse biased and the emitter-base junction is d-c open-circuited. A table is included in the instruction manual which gives a numerical listing of the transistors capable of being tested.

DESCRIPTION

The Test Set (TS-1100/U) as shown in the block diagram of figure 16-8 consists essentially of a reference oscillator, tuned amplifier, and a variable bias supply. The transistor to be tested is shown in the dashed circle. The oscillator is used to generate a test signal, the tuned amplifier provides a means of measuring the second harmonic component of the current in the output circuit of the transistor test, and the bias circuit furnishes the appropriate voltages for testing the transistor.



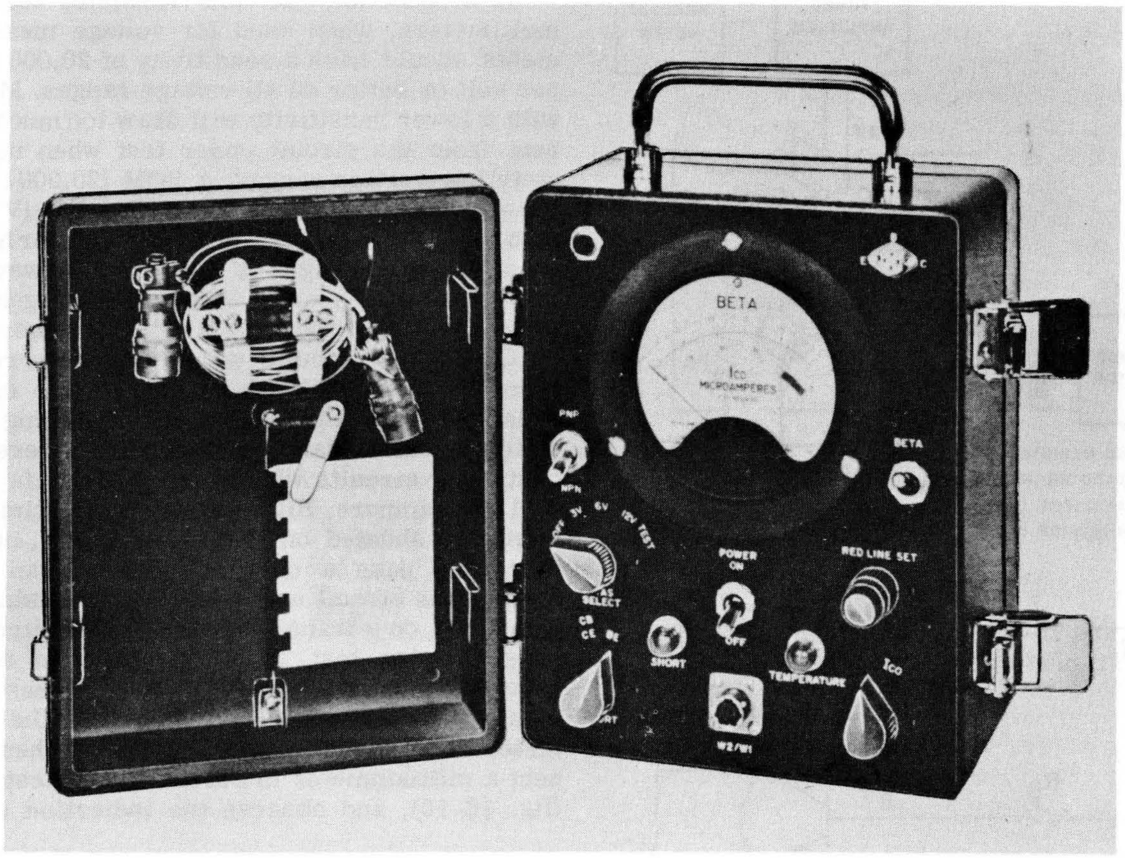
4.133

Figure 16-6.—Controls, jacks, leads, and accessories.

MEASUREMENTS IN TRANSISTOR CIRCUITS

The many types of transistor base connection arrangements require that the leads be properly identified to secure correct hookup to the tester. The TS-1100/U test socket arrangement is compatible with some transistor types, but not all types. If the leads are long enough, it is generally possible to effect proper hookup by bending them. If the leads are too short, it will be necessary to use the test cable and alligator clips provided with the tester. In all cases, however, the transistor leads should be identified and then matched up with the tester connections.

The great advantage of the TS-1100/U lies not only in its accuracy and simplicity, but also in its use of a-c as the testing current. This eliminates interference from any d-c currents and voltages that may be present and permits measurement of the gain of a transistor in-circuit, thus making it unnecessary to unsolder or disconnect the transistor for this test. This is particularly advantageous where the transistor is mounted on a module printed wiring board.



72.37

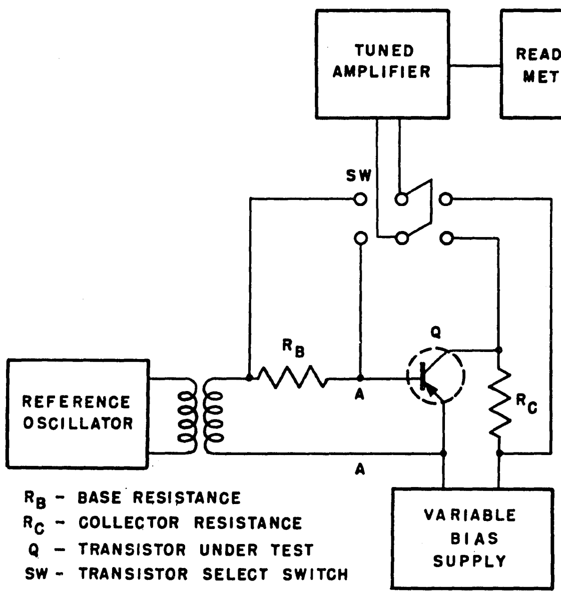
Figure 16-7.—Transistor Test Set TS-1100/u.

The a-c testing current is provided by means of a built-in oscillator and amplifier. To avoid possible interference from other frequencies, the amplifier is sharply peaked for a single frequency, 2250 cycles per second. The test current is then rectified and read on a microammeter which has a full-scale deflection of 50 microamperes. The test is also provided with a variable bias supply. The circuit arrangement for measuring beta for a given transistor is shown in figure 16-8. When the proper adjustments are made according to the instructions in the instruction book, the value of beta will be indicated directly on the meter.

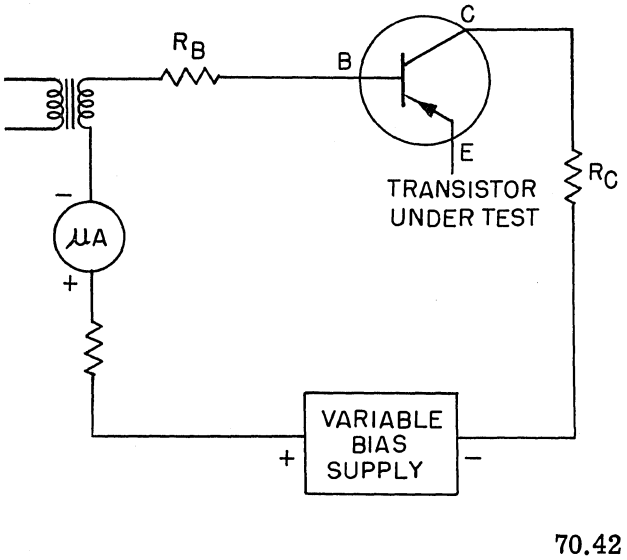
TS-1100/U is also designed to measure collector leakage current, I_{CO}, which is read directly on the meter. The circuit arrangement is shown in figure 16-9. The technical manual for the test set provides information concerning the collector bias to be applied for a particular transistor and the maximum permissible collector leakage current.

However, since this current is d-c, the reading may be affected by other d-c potentials in the circuit. For this reason it is necessary to remove the transistor from the circuit in order to obtain an accurate measurement of collector leakage current.

The test set is also equipped to indicate a short between any two of the three elements of the transistor under test. With the transistor in-circuit, it will also indicate a short if the circuitry between any two of the transistor elements has a resistance of 500 ohms or less. To determine whether the short is in the transistor itself or in the associated circuit, it is necessary to remove the transistor from the circuit.



124.223
Figure 16-8.—Measuring beta, using Transistor Test Set TS-100/u.

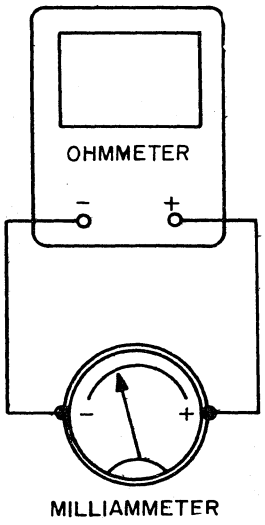


70.42
Figure 16-9.—Measuring collector leakage current, Ico, using Transistor Set TS-100/u.

The test set has the following additional features: a switch marked PNP-NPN, which selects the proper bias polarity for the type of transistor under test; a temperature alarm indicator lamp, which will light when the ambient temperature surrounding the equipment exceeds 50°C; and a switch marked TEST, which checks the test set battery output.

In conjunction with the transistor test set, multimeters, when used for voltage measurements should have a sensitivity of 20,000 ohms per volt or better on all voltage ranges. Meters with a lower sensitivity will draw too much current from the circuit under test when used in their low-voltage ranges. A VOM (20,000 ohms-per-volt) or an electronic voltmeter (VTVM) with an input resistance of 11 megohms or higher on all voltage ranges is preferred. However, a VTVM should be used with an isolating transformer between the VTVM and the power line.

Ohmmeter circuits which pass a current of more than 1 milliampere through the circuit under test cannot be used safely in testing transistor circuits. Many electronic voltmeters have ohmmeter circuits which exceed this safe value of 1 milliampere. High-sensitivity multimeters often are shunted on ohmmeter ranges, so that they also pass a current of more than 1 ma through the circuit under test. Before using any ohmmeter on a transistor circuit, the current it passes under test should be checked on all ranges. Do not use any range which passes more than 1 ma. To check the current, adjust the ohmmeter for resistance measurements; then connect a milliammeter in series with the test leads (fig. 16-10), and observe the indication on the



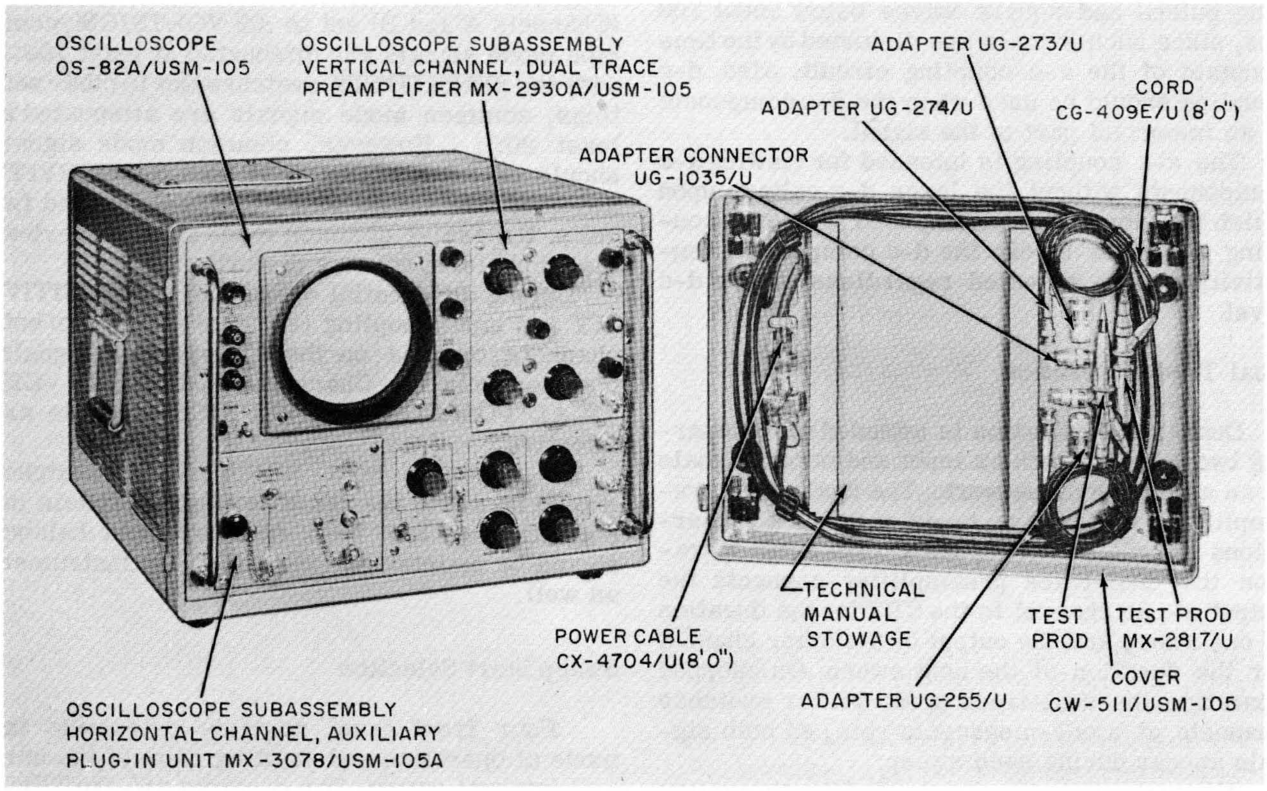
1.289
Figure 16-10.—Measuring current passed by ohmmeter

milliammeter. The meter used should have a low resistance such as contained in Multimeter TS-352A/U or equivalent.

OSCILLOSCOPE AN/USM-105A

Oscilloscope AN/USM-105A (fig. 16-11) is a general purpose, high-speed laboratory type oscilloscope designed for shipboard use. It produces a graphical display of simple and complex voltage variations which contain frequency components ranging from zero to 14 megacycles. To simplify operation and the interpretation of the display, the instrument provides calibrated vertical sensitivities, triggered internal sweeps, calibrated sweep times, calibrated expanded sweeps, beam finder, and calibrator.

Oscilloscope AN/USM-105A consists of a major unit and two plug-in units. The major unit, Oscilloscope OS-82A/USM-105, contains the power supplies, horizontal amplifier, sweep generator, main vertical amplifier, cathode ray tube (CRT), calibrator, and the controls associated with these circuits. Oscilloscope Subassembly, Vertical Channel, Dual Trace Preamplifier MX-2930A/USM-105 (hereafter referred to as dual trace preamplifier) is a plug-in preamplifier to the main vertical amplifier. The dual trace preamplifier contains two separate voltage channels each with its own controls. An electronic switch, controlled from the front panel, connects one channel or the other to the main vertical amplifier and thereby determines the vertical presentation on the CRT. To produce a dual trace, the electronic switch alternates channels, either on alternate sweeps, or continuously at a one-megacycle rate. Oscilloscope Subassembly, Horizontal Channel, Auxiliary Plug-in Unit MX-3078/USM-105A (hereafter referred to as auxiliary plug-in) permits single-sweep operation and external intensity modulation.


OSCILLOSCOPE OS-82A/USM-105
OSCILLOSCOPE SUBASSEMBLY VERTICAL CHANNEL, DUAL TRACE PREAMPLIFIER MX-2930A/USM-105
ADAPTER UG-273/U
CORD CG-409E/U(8'0")
ADAPTER UG-274/U
ADAPTER CONNECTOR UG-1035/U
TECHNICAL MANUAL STOWAGE
POWER CABLE CX-4704/U(8'0")
TEST PROD
TEST PROD MX-2817/U
COVER CW-511/USM-105
ADAPTER UG-255/U
OSCILLOSCOPE SUBASSEMBLY HORIZONTAL CHANNEL, AUXILIARY PLUG-IN UNIT MX-3078/USM-105A

1.86.5

Figure 16-11.—Oscilloscope, AN/USM-105A.

The basic principles of oscilloscopes are treated in the Basic Electronics training course NavPers 10087 (revised). It is therefore assumed that the reader is familiar with these principles.

FUNCTIONAL OPERATIONS

Personnel should become familiar with the controls and operating features of the AN/USM-105A before proceeding with tests and measurements. The front panel controls and connectors are illustrated and described in figure 16-12. Some of the operating features are described below.

D-c or A-c Coupled Input

The vertical and horizontal amplifiers of the AN/USM-105A are d-c coupled. However, internal coupling capacitors (not shown) can be switched into these circuits in series with the INPUT connectors to provide a-c coupling as well.

The d-c coupling should be used for viewing long pulses and square waves below about 100 cps, since such signals are distorted by the time constant of the a-c coupling circuit. Also, d-c coupling should be used when the d-c component is an important part of the signal.

The a-c coupling is intended for viewing a-c components without the large d-c voltage upon which they may be superimposed. Since the coupling capacitor blocks the d-c component, sensitivity can be selected regardless of the d-c level.

Dual Trace Operation

Dual trace operation is intended for comparing two signals, such as input and output signals of an amplifier or network. The dual-trace preamplifier provides two types of dual trace operations alternate and chopped. On alternate operation the dual-trace preamplifier connects the output of one channel to the CRT for the duration of one sweep and the output of the other channel for the duration of the next sweep. On chopped operation the dual-trace preamplifier switches channels at a one-megacycle rate, so both signals appear during each sweep.

Alternate operation is intended for comparing signals which require high sweep speeds. However, for valid time comparisons, the sweeps must be triggered by an external trigger signal which is synchronized with both vertical signals. In many cases, one of the vertical signals can be used as the trigger signal as well.

Chopped operation is intended for comparing signals which require sweep speeds below about five microseconds/centimeter, sweep speeds which are low compared to the 1-megacycle switching rate. This type of operation permits precise time comparisons as long as the presentation is stable on the CRT because both signals are displayed during the same sweep. In general, external triggering is required. If internal triggering is used, the one-megacycle switching signal may trigger the sweep and cause an unstable presentation.

Differential Operation

Differential operation permits observation of the difference between two signal voltages (one on the channel A input and the other on channel B) and is useful for observing signals which are balanced with respect to ground. Hum and pickup signals common to the two main signals are attenuated. With both SENSITIVITY switches (Channels A and B) set to .02 VOLTS/CM, common mode signals are attenuated at least 100:1; with the SENSITIVITY switches set to other settings, common mode signals are attenuated at least 20:1. However, common mode signals should not, for example, if both SENSITIVITY switches are set to .02 VOLTS/CM, exceed two volts. Excessive common mode signals overload the amplifiers, causing distortion.

During differential operation, the SENSITIVITY and input coupling (a-c d-c) switches of both channels operate on their respective signals; however only the Channel A POLARITY, VERTICAL POSITION, and VERNIER controls are effective.

In general, both SENSITIVITY switches should be set to the same setting to maintain input balance. For best results, input balance should be maintained external to the instrument as well.

Sweep Start Selection

Four front-panel controls determine the mode of operation and starting point of the normal internal sweep. The controls are the TRIGGER SOURCE switch, the TRIGGER LEVER control, the TRIGGER SLOPE switch, and the SWEEP MODE control.

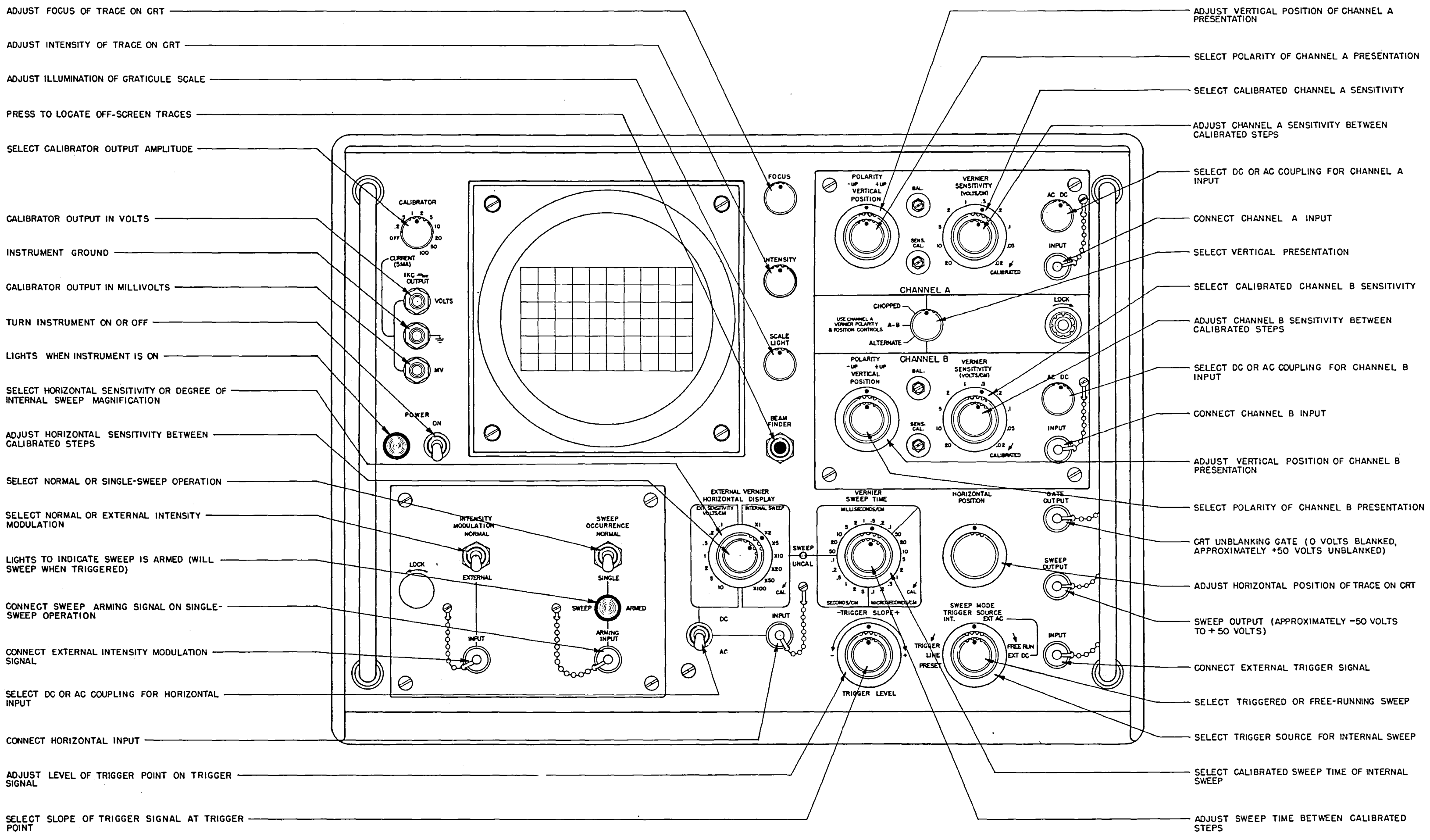

ADJUST FOCUS OF TRACE ON CRT
ADJUST INTENSITY OF TRACE ON CRT
ADJUST ILLUMINATION OF GRATICULE SCALE
PRESS TO LOCATE OFF-SCREEN TRACES
SELECT CALIBRATOR OUTPUT AMPLITUDE
CALIBRATOR OUTPUT IN VOLTS
INSTRUMENT GROUND
CALIBRATOR OUTPUT IN MILLIVOLTS
TURN INSTRUMENT ON OR OFF
LIGHTS WHEN INSTRUMENT IS ON
SELECT HORIZONTAL SENSITIVITY OR DEGREE OF INTERNAL SWEEP MAGNIFICATION
ADJUST HORIZONTAL SENSITIVITY BETWEEN CALIBRATED STEPS
SELECT NORMAL OR SINGLE-SWEEP OPERATION
SELECT NORMAL OR EXTERNAL INTENSITY MODULATION
LIGHTS TO INDICATE SWEEP IS ARMED (WILL SWEEP WHEN TRIGGERED)
CONNECT SWEEP ARMING SIGNAL ON SINGLE-SWEEP OPERATION
CONNECT EXTERNAL INTENSITY MODULATION SIGNAL
SELECT DC OR AC COUPLING FOR HORIZONTAL INPUT
CONNECT HORIZONTAL INPUT
ADJUST LEVEL OF TRIGGER POINT ON TRIGGER SIGNAL
SELECT SLOPE OF TRIGGER SIGNAL AT TRIGGER POINT
ADJUST VERTICAL POSITION OF CHANNEL A PRESENTATION
SELECT POLARITY OF CHANNEL A PRESENTATION
SELECT CALIBRATED CHANNEL A SENSITIVITY
ADJUST CHANNEL A SENSITIVITY BETWEEN CALIBRATED STEPS
SELECT DC OR AC COUPLING FOR CHANNEL A INPUT
CONNECT CHANNEL A INPUT
SELECT VERTICAL PRESENTATION
SELECT CALIBRATED CHANNEL B SENSITIVITY
ADJUST CHANNEL B SENSITIVITY BETWEEN CALIBRATED STEPS
SELECT DC OR AC COUPLING FOR CHANNEL B INPUT
CONNECT CHANNEL B INPUT
ADJUST VERTICAL POSITION OF CHANNEL B PRESENTATION
SELECT POLARITY OF CHANNEL B PRESENTATION
CRT UNBLANKING GATE (0 VOLTS BLANKED, APPROXIMATELY +50 VOLTS UNBLANKED)
ADJUST HORIZONTAL POSITION OF TRACE ON CRT
SWEEP OUTPUT (APPROXIMATELY -50 VOLTS TO +50 VOLTS)
CONNECT EXTERNAL TRIGGER SIGNAL
SELECT TRIGGERED OR FREE-RUNNING SWEEP
SELECT TRIGGER SOURCE FOR INTERNAL SWEEP
SELECT CALIBRATED SWEEP TIME OF INTERNAL SWEEP
ADJUST SWEEP TIME BETWEEN CALIBRATED STEPS
CALIBRATOR
FOCUS
INTENSITY
SCALE LIGHT
BEAM FINDER
POWER
ON
VOLTS
MV
CHANNEL A
CHANNEL B
LOCK
CHOPPED
A-B
ALTERNATE
POLARITY
VERTICAL POSITION
VERNIER SENSITIVITY (VOLTS/CM)
BAL.
SENS. CAL.
CALIBRATED
AC DC
INPUT
EXTERNAL VERNIER HORIZONTAL DISPLAY
VERNIER SWEEP TIME
SWEEP UNCAL
HORIZONTAL POSITION
GATE OUTPUT
SWEEP OUTPUT
INTENSITY MODULATION
NORMAL
EXTERNAL
SWEEP OCCURRENCE
SINGLE
SWEEP
ARMED
ARMING INPUT
DC
AC
-TRIGGER SLOPE+
TRIGGER LEVEL
SWEEP MODE TRIGGER SOURCE
INT.
EXT AC
EXT DC
FREE RUN
TRIGGER
LINE
PRESET

Figure 16-12.—Front panel controls and connectors.

1.86.5

The TRIGGER SOURCE switch selects the source of the trigger signal: the power line (LINE), the signal applied to the dual-trace preamplifier (INT), or an external signal applied to the trigger source INPUT (EXT AC or EXT DC).

The TRIGGER LEVEL control adjusts the voltage level on the trigger signal at which the sweep starts. The control provides continuous adjustment of the trigger level from about -30 volts to about +30 volts on external trigger signals and over a range equivalent to about six centimeters of vertical deflection on the CRT on external trigger signals.

The TRIGGER SLOPE switch determines whether the sweep starts on the positive-slope or negative-slope portion of the trigger signal.

The SWEEP MODE control determines whether the sweep requires a trigger or runs free. The control is continuously adjustable and has a switched position PRESET, at its counterclockwise extreme. The switched position is the best overall setting for the control when the trigger signal is below about 10 megacycles. This position is internally set for optimum triggered operation. For trigger signals above 10 megacycles, free-run operation may be better. In this case the trigger signal synchronizes the sweep with the signal being viewed. For very high frequency trigger signals, a fine adjustment of the SWEEP MODE and/or TRIGGER LEVEL controls may be necessary to stablize the presentation on the CRT.

Single Sweep Operation

Single-sweep operation is intended for viewing transients and other one-shot signals and also periodic (irregular) signals which can not be viewed using normal triggered-sweep operation. In single-sweep operation, the sweep must be re-triggered before it can generate another. The sweep generator can be triggered automatically with an external signal or manually by rotating the SWEEP MODE control just out of PRESET.

Beam Finder

The BEAM FINDER is a pushbutton switch which, when pressed, limits maximum beam deflection to the edge of the CRT. Thus the BEAM FINDER helps locate traces which are normally deflected off the screen. When the switch is pressed, the beam is confined to the face of the CRT, brightened, and also defocused to prevent accidental burning of the CRT face. The trace can be centered with the horizontal and vertical position controls while the BEAM FINDER is pressed; the trace will then remain on screen when the switch is released.

Calibrator

An internal calibrator provides a 1-kc voltage square wave calibrated in volts and millivolts peak-to-peak and a 1-kc current square wave 5 millamperes peak-to-peak. The calibrator output can be used to check vertical and horizontal sensitivities and to match the test prods or other external coupling devices to a particular input.

Test Prod

The test prods are used to decrease circuit loading. The one megohm input impedance of the vertical and horizontal circuits plus the shunt capacity of a cable connecting the oscilloscope to the test circuit may degrade the operation of the circuit under test. The test prod increases input impedance to 10 megohms shunted by 10 picofarads. The test prod also introduces a 10:1 voltage division which must be considered when translating waveform deflection on the CRT into volts.

The test prod has an adjustable compensating capacitance so the prod can be matched exactly to a particular input of the oscilloscope. The procedure for matching a test prod to the Channel A input of the dual-trace preamplifier is given below. The procedure is similar for matching a test prod to any other input.

(1) Connect test prod cable to Channel A INPUT.

(2) Select CHANNEL A and set Channel A SENSITIVITY switch to .02 VOLTS/CM.

(3) Set SWEEP TIME switch to .5 MILLISECONDS/CM, HORIZONTAL DISPLAY switch to X1, TRIGGER SOURCE switch to INT, and SWEEP MODE control to PRESET.

(4) Set CALIBRATOR switch to 1 and touch test prod to VOLTS terminal of calibrator output. A square wave five centimeters high should appear on the CRT.

(5) Loosen knurled locknut just behind rear flange on test prod body.

(6) Hold test prod behind locknut and rotate rear flange to give flat-topped square wave.

(7) Tighten knurled locknut without disturbing adjustment. This completes adjustment.

OPERATING INSTRUCTIONS

The following procedures give step-by-step operating instructions for Oscilloscope AN/USM-105A. The first procedure gives complete instructions for single-channel operation. The remaining procedures give instructions peculiar only to the modes of operation with which they are concerned.

Before making any test or measurements, allow the instrument about five minutes warmup. Rotate INTENSITY control fully counterclockwise before turning instrument on to prevent accidental burning of the CRT face during warmup.

Use the test prods and other accessories furnished with the oscilloscope as necessary.

Single Trace Operation-Internal Sweep

(1) Connect vertical signal to INPUT of Channel A.

(2) Set Vertical Presentation switch to CHANNEL A.

(3) Set SENSITIVITY switch for Channel A as desired. (Set VERNIER to CALIBRATED for calibrated sensitivity.)

(4) Set input coupling for a-c or d-c coupling as desired.

(5) Set POLARITY switch to +UP or -UP as desired.

(6) Set TRIGGER SOURCE as desired. If external trigger is used, connect it to trigger source INPUT.

(7) Set INTENSITY MODULATION and SWEEP OCCURRENCE switches to NORMAL.

(8) Set HORIZONTAL DISPLAY switch to INTERNAL SWEEP X1.

(9) Set SWEEP MODE to PRESET.

(10) Set TRIGGER SLOPE switch for triggering on positive or negative slope of trigger signal as desired.

(11) Set TRIGGER LEVEL control to 0.

(12) Set SWEEP TIME switch for desired sweep time (set VERNIER control to CAL for calibrated sweep time).

(13) Set INSENSITY control as desired.

(14) Adjust VERTICAL POSITION and HORIZONTAL POSITION controls as desired.

(15) If trace does not appear on screen, press BEAM FINDER switch and readjust position controls to center trace.

(16) Adjust TRIGGER LEVEL control to start trace at desired level of trigger signal. It may be necessary to switch SWEEP MODE control from PRESET and select a better setting for the particular trigger signal being used.

Note

To use Channel B for single trace operation follow the above procedure except to subsitute Channel B controls and terminals for the Channel A controls and terminals called out in the procedure.

Dual Trace Operation

(1) Connect one signal to INPUT connector of Channel A and set Channel A controls as desired.

(2) Connect second signal to INPUT of Channel B and set Channel B controls as desired.

(3) Set Vertical Presentation switch to CHOPPED for display of both signals on same sweep, to ALTERNATE for display of signals on alternate sweeps.

Note

For best results, use external triggering.

Differential Operation

(1) Connect one signal to INPUT of Channel A.

(2) Set SENSITIVITY switch of Channel A as desired.

(3) Connect second signal to INPUT of Channel B.

(4) For best results, set SENSITIVITY of Channel B to same setting as SENSITIVITY of Channel A.

(5) Set POLARITY switch of Channel A to +UP.

(6) Set Vertical Presentation switch to A-B.

(7) If vertical adjustment is necessary, use VERTICAL POSITION control of Channel A.

Internal Sweep Magnification

(1) Set SWEEP TIME switch as desired (set VERNIER control to CAL for calibrated sweep time).

(2) Set HORIZONTAL DISPLAY switch to INTERNAL SWEEP X1 (unmagnified sweep position).

(3) Adjust HORIZONTAL POSITION control to place portion of trace to be magnified under vertical center line of graticule (lined scale on the screen of the CRT).

(4) Set HORIZONTAL DISPLAY to desired magnification.

(5) Readjust INTENSITY as necessary.

(6) If selected sweep time with magnification is less than minimum calibrated sweep time (0.02 microseconds/cm), SWEEP UNCAL indicator will light indicating that sweep time is no longer calibrated.

External Horizontal Input

(1) Connect horizontal signal to horizontal INPUT connector.
(2) Set horizontal input coupling switch to a-c or d-c as desired.
(3) Set HORIZONTAL DISPLAY to desired external sensitivity.
(4) Adjust HORIZONTAL POSITION control as desired.

Intensity Modulation

(1) Set INTENSITY MODULATION switch to EXTERNAL.
(2) Connect modulation signal to external intensity modulation INPUT connector. Positive signal of 20 volts peak will blank trace from normal intensity; negative signal will brighten trace.

Single Sweep Operation

(1) Set SWEEP OCCURRENCE switch to NORMAL.
(2) Set SWEEP TIME switch as desired (set VERNIER control to CAL for calibrated sweep time).
(3) Set TRIGGER SOURCE switch according to trigger signal used.
(4) Set SWEEP MODE as desired.
(5) Set TRIGGER SLOPE as desired.
(6) Adjust TRIGGER LEVEL as desired.
(7) Set SWEEP OCCURRENCE to SINGLE.
(8) To arm sweep circuit, switch SWEEP MODE just out of PRESET and back to PRESET, or apply pulse 1 to 4 microseconds long and +15 to +25 volts peak to ARMING INPUT connector.
(9) SWEEP ARMED indicator will light. After sweep, indicator will extinguish, and sweep circuit will remain disabled until rearmed.

Connecting Signal Directly to CRT Deflection Plates

CAUTION

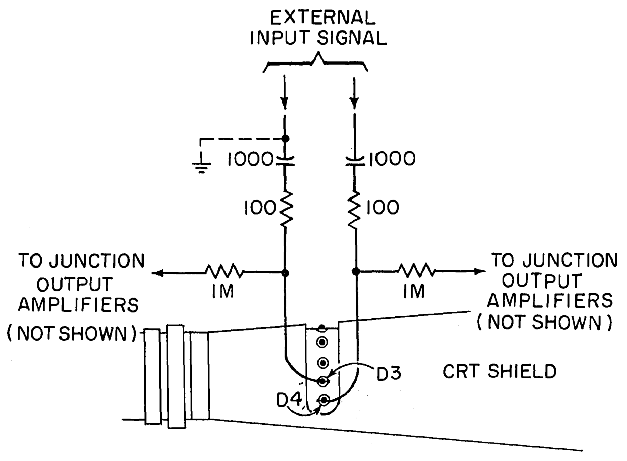
Do not contact CRT deflection plate terminals with instrument turned on. These terminals are normally operated about +200 volts.

(1) Turn oscilloscope off and remove access plate on top of cabinet.
(2) Remove leads from vertical deflection plate terminals D3 and D4. See figure 16-13.
(3) Connect components as shown in figure 16-13. Use capacitors with good high-frequency response. Front-panel VERTICAL POSITION control remains effective.
(4) For single-ended input, ground common signal lead as shown by dashed ground lead in figure 16-13. For balanced input leave both signal leads ungrounded.
(5) Turn oscilloscope on. Use external signal to trigger sweep.

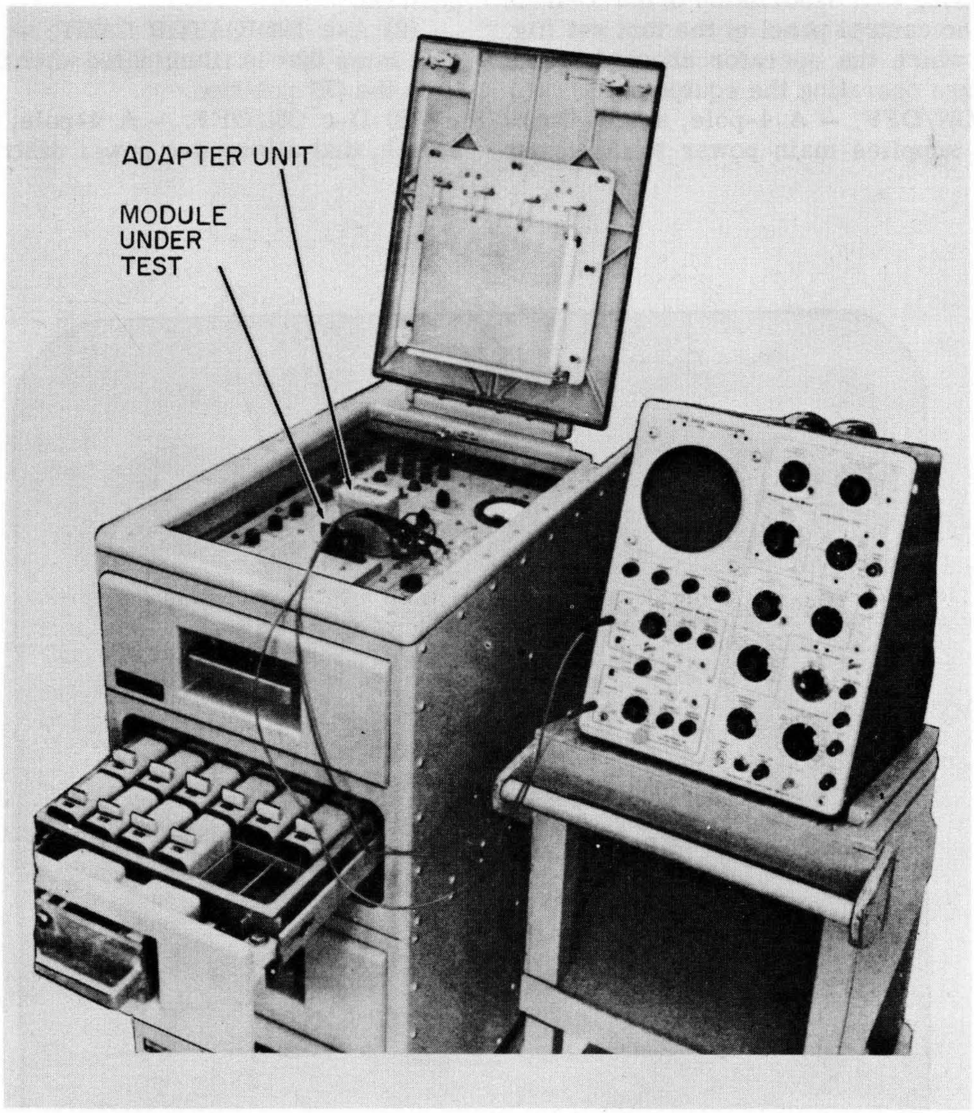
TEST SET, ELECTRONIC CIRCUIT PLUG-IN UNIT AN/USM-142(v)

The Test Set, Electronic Circuit Plug-In Unit, AN/USM-142(v) shown in figure 16-14 is designed to give an adequate functional test to a majority of the modules (cards) used in the NTDS Computer and its peripheral equipment. The test set provides operating conditions similar to those found in the computer and its peripheral equipment.

The test set generates test signals of the desired waveform, pulse repetition rate, and amplitude. Switches and potentiometers on the control panel enable the operator to set up the various conditions necessary for providing the specific module under test with an operational check.


EXTERNAL INPUT SIGNAL
1000
1000
100
100
TO JUNCTION OUTPUT AMPLIFIERS (NOT SHOWN)
IM
IM
TO JUNCTION OUTPUT AMPLIFIERS (NOT SHOWN)
D3
CRT SHIELD
D4

124.224
Figure 16-13.—Direct connection to deflection plates of cathode-ray tube.



124.225

Figure 16-14.—Electronic Circuit Plug-in Unit Test Set AN/USM-142(v).

Input signals, output loads, and d-c voltages are supplied to the module under test by means of a prewired, 104-pin, adapter unit. The adapter unit also connects the output signal or signals of the module under test to an oscilloscope, where comparison is made between the waveform on the oscilloscope and the expected waveform depicted in the individual test procedure for a specific module. This comparison indicates the relative operational value of the suspected module.

The top of the test set cabinet has a hinged cover that provides access to the operator's control panel. All controls necessary for testing a module can be operated through this opening. The hinged cover also contains the Test Procedure Manual that contains instructions for setting up tests for suspected modules.

The test set provides facilities for giving each module under test a functional check of its complete circuits.

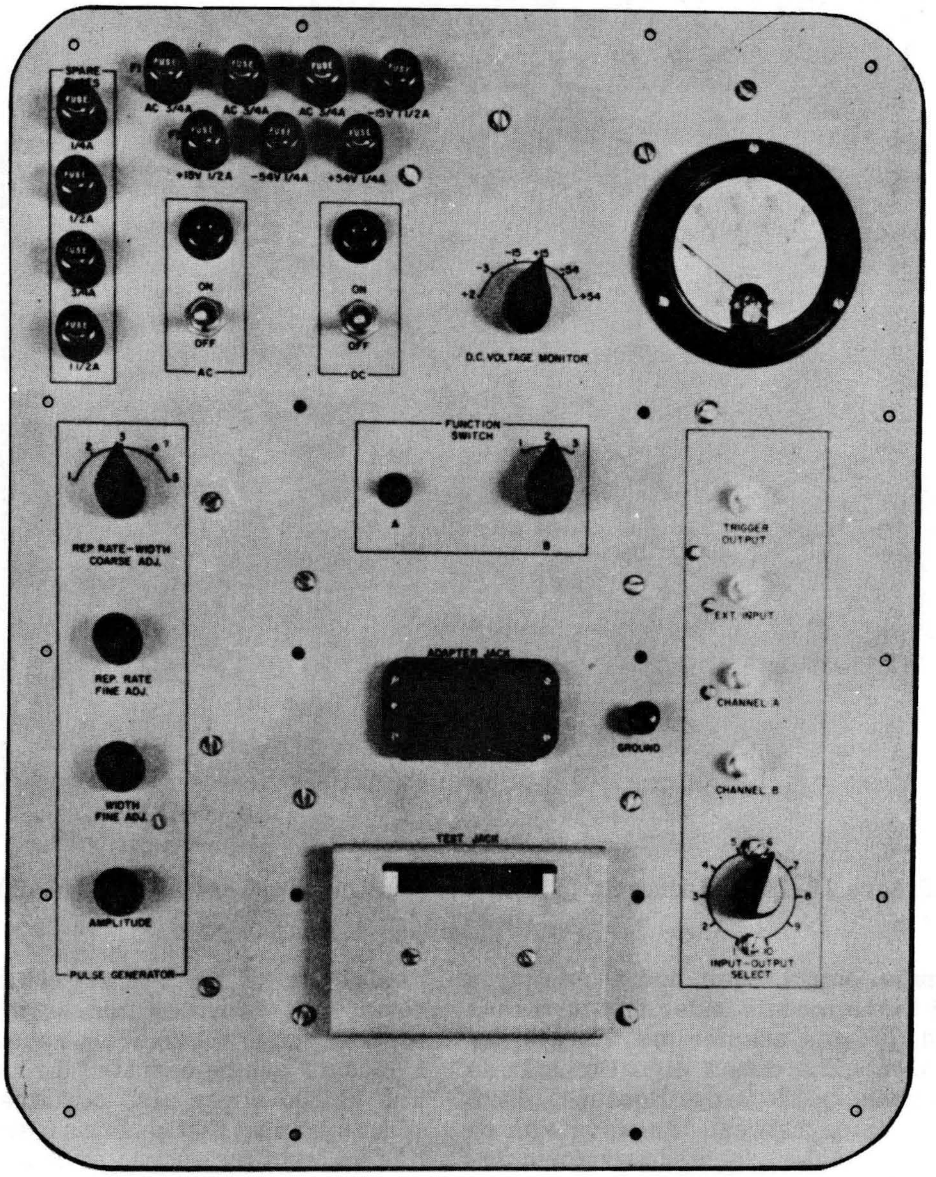
DESCRIPTION OF CONTROLS

The following is a description of the controls located on the control panel of the test set (fig. 16-15) with which the operator should become familiar before operating the equipment.

(1) A-c ON/OFF. — A 4-pole, single-throw switch, that supplies main power to the power supply transformer, the cooling fan, the a-c indicator lamp, and a relay when in the ON position.

(2) A-c INDICATOR LAMP. — An incandescent lamp that is illuminated when the a-c switch is in the ON position.

(3) D-c ON/OFF. — A 4-pole, single-throw switch, that allows d-c power distribution to the



124.226

Figure 16-15.—Operator's control panel.

meter, pulse generator, and adapter jack when in the ON position.

(4) D-c INDICATOR LAMP. — An incandescent lamp that is illuminated when the d-c switch is in the ON position.

(5) D-c VOLTAGE MONITOR.—A 6-position, rotary-wafer switch that changes the range of the meter for measuring the +2, -3, +15, -15, +54, and -54 d-c volt outputs of the power supply.

(6) METER. — A 3-range d-c milliammeter calibrated in d-c volts, used to indicate the value of the d-c voltage outputs of the power supply, and used in conjunction with the DC VOLTAGE MONITOR switch.

(7) REP. RATE-WIDTH COARSE ADJ. — A 5-position, rotary-wafer switch that simultaneously controls the rep. rate and width of the output of the pulse generator.

(8) REP. RATE FINE ADJ. — A potentiometer that fine adjusts the rep. rate of the pulse generator.

(9) WIDTH FINE ADJ. — A potentiometer that fine adjusts the width of the pulses generated by the pulse generator.

(10) AMPLITUDE. — A potentiometer that adjusts the magnitude of the output pulses generated by the pulse generator.

(11) FUNCTION SWITCH A. — A momentary contact pushbutton switch that can be programmed (enabled) in by the adapters for certain tests.

(12) FUNCTION SWITCH B. — A 3-position, rotary-wafer switch that can be programmed in by the adapters for certain tests.

(13) INPUT-OUTPUT SELECT. — A 10-position, rotary-wafer switch that is programmed by the adapter to make input and output connections to the module under test.

(14) TRIGGER OUTPUT. — A coaxial connector that supplies a trigger output from the test set.

(15) CHANNEL A. — A coaxial connector that provides an output to CHANNEL A of an oscilloscope.

(16) CHANNEL B. — A coaxial connector that provides an output to CHANNEL B of an oscilloscope.

(17) EXT. INPUT. — A coaxial connector that provides an external input terminal to the test set.

(18) ADAPTER JACK. — A 104-pin jack that holds the adapters during operation of the equipment.

(19) TEST JACK. — A 15-pin jack that holds the module to be tested during operation of the equipment.

OPERATING INSTRUCTIONS

The sequence of instructions listed in table 16-1 must be completed before a suspected module is tested.

TEST ADAPTER

The test adapter (fig. 16-16) is a plug-in unit that programs the test set for testing a specific module. It provides a means for connecting the output signals of the pulse generator to the module under test, connecting the oscilloscope to the module under test, and providing loads to the module similar to those encountered in computer operation.

Manipulation of the various controls on the test set control panel and oscilloscope control panel, as described in table 16-2, will complete the programming for a specific module. A comparison between the waveforms as seen on the oscilloscope with those illustrated in the lower right section of table 16-2 should be made to determine the module's operational condition.

The AN/USM-142 is not presently being furnished with NTDS equipment and is being removed from old installations. A later and more versatile equipment, the TS-1780/SYA-4(v) display module test set, is being supplied as a replacement for the AN/USM-142.

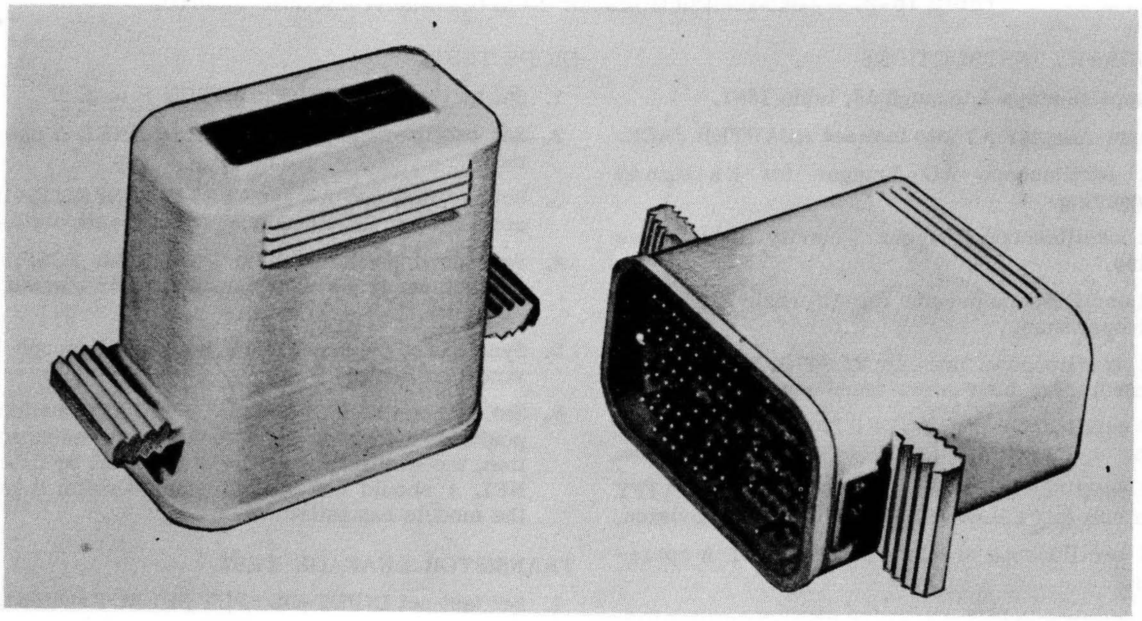
CONVERTER TESTER, TS-1538/USQ-20(v)

The converter tester (fig. 16-17) provides the operator a means of varying the a-c input power to an equipment undergoing a marginal power operating test. A self-contained a-c voltmeter allows monitoring the three-phase, 115-v line-to-line voltage. A marginal operating environment can thus be simulated for the purpose of observing equipment operation under adverse power conditions. As a result, early detection of circuit, or component deterioration is possible. The Converter Tester is intended for use with the equipments specified in table 16-3.

In use, the Converter Tester is connected between the primary power source and the equipment's power supply circuit supplying d-c power to the circuits undergoing the marginal operating test. Figure 16-18 illustrates the functional relationship between the Converter Tester and the CV-1123/USQ-20(v) which is discussed in chapter 14. The Converter Test is set to adjust the input a-c power over a range plus or minus ten percent of normal.

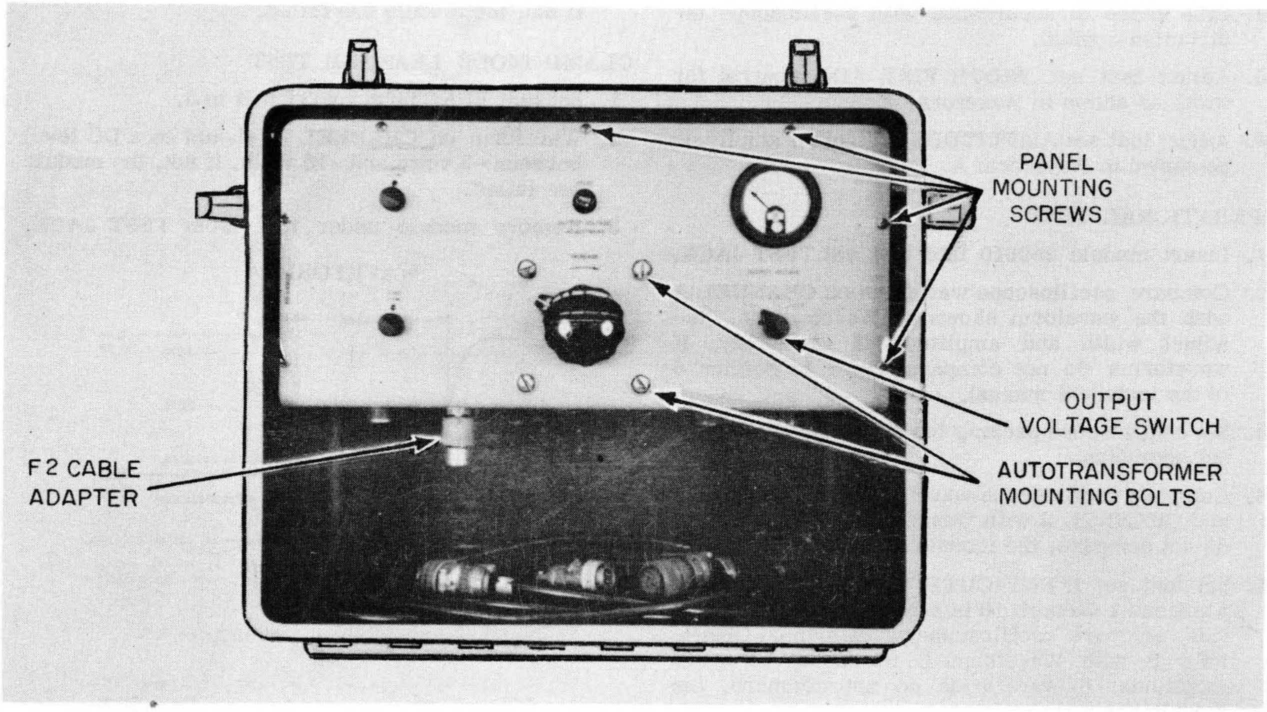
Table 16-1. — Preliminary Operating Instructions.

Step	Action	Remarks
1	Remove module from TEST JACK	Precaution
2	Remove test adapter from ADAPTER JACK	Precaution
3	Connect test set power cord to 115 VAC, 3Ø, 400-cps supply source	Main power to test set
4	Connect oscilloscope power cord to 115 VAC, single-phase, 60-cps supply source	Main power to oscilloscope
5	Connect probe from oscilloscope CHANNEL A to test set CHANNEL A	10:1 attenuated probe
6	Connect probe from oscilloscope CHANNEL B to test set CHANNEL B	(10:1 attenuated probe)
7	Connect probe ground leads to GROUND connector on test set	Provide reference level
8	Connect a cable from oscilloscope TRIGGER INPUT to test set TRIGGER OUTPUT	(Do not use attenuated probe)
9	Set POWER switch on oscilloscope to the ON position	Oscilloscope warming up
10	Set AC switch on test set to ON	Test set warming up
11	Set DC switch on test set to ON	Test set warming up
12	Set test set DC VOLTAGE MONITOR switch to +2	Meter will indicate 0 volts
13	Set test set DC VOLTAGE MONITOR switch to -3	Meter will indicate 3 volts on the 5 volts scale
14	Set test set DC VOLTAGE MONITOR switch to +15	Meter will indicate 15 volts on the 25 volts scale
15	Set test set DC VOLTAGE MONITOR switch to -15	Meter will indicate 15 volts on the 25 volts scale
16	Set test set DC VOLTAGE MONITOR switch to +54	Meter will indicate 54 volts on the 100 volts scale
17	Set test set DC VOLTAGE MONITOR switch to -54	Meter will indicate 54 volts on the 100 volts scale
18	Locate test procedure for module to be tested and follow the procedure indicated	



124.227

Figure 16-16.—Adapter units.



124.228

Figure 16-17.—Converter tester, front view.

Table 16-2. —Test Instructions for Representative Module 250010.

PRELIMINARY INSTRUCTIONS

1. Complete steps 1 through 18, table 16-1.
2. Insert Adapter A1 into test set ADAPTER JACK.
3. Set oscilloscope AC Trigger for External triggering.
4. Set oscilloscope Trigger Polarity for negative pulse.
5. Set oscilloscope preamp for Alternate Dual channel operation.
6. Set oscilloscope preamp VERTICAL POSITION controls for convenient trace reference levels.
7. Set oscilloscope preamp VERTICAL SENSITIVITY control for CHANNEL A to 0.1 volt/division.
8. Set oscilloscope preamp VERTICAL SENSITIVITY control for CHANNEL B to 0.1 volt/division.
9. Set oscilloscope preamp for CHANNEL A operation.
10. Set test set INPUT-OUTPUT SELECT switch to 1.
11. Set test set FUNCTION SWITCH B to 1.
12. Set test set REP. RATE-WIDTH COARSE ADJ. switch to 5.
13. Set test set REP. RATE FINE ADJ. control full right.
14. Sync pulse in accordance with oscilloscope instruction manual.
15. Adjust test set WIDTH FINE ADJ. control for width as shown in Waveform A.
16. Adjust test set AMPLITUDE control for amplitude as shown in Waveform A.

OPERATIONAL TEST

1. Insert module 250010 into test set TEST JACK.
2. Compare oscilloscope waveform on CHANNEL A with the waveform shown at Waveform A. Readjust width and amplitude if necessary. If waveforms do not compare, refer to Section 6 of the technical manual.
3. Set oscilloscope preamp for Alternate Dual channel operation.
4. Compare oscilloscope waveforms on CHANNEL A and CHANNEL B with Waveforms A and B. If they do not compare, the module has failed.
5. Set test set INPUT-OUTPUT SELECT switch to positions 1 through 10 in sequence. At each position, compare oscilloscope waveform on CHANNEL B with Waveform B for delay time and amplitude. If waveforms do not compare, the module has failed.

DIODE TEST

1. Set test set FUNCTION SWITCH B to 2.
2. Set oscilloscope preamp for CHANNEL A operation.
3. Set oscilloscope preamp VERTICAL SENSITIVITY control for CHANNEL A to 0.5 volt/division.
4. Set oscilloscope preamp VERTICAL POSITION control so trace is convenient for measuring +15 volts.
5. Sync pulse in accordance with oscilloscope instruction manual.
6. Set test set INPUT-OUTPUT SELECT switch to positions 1 through 10 in sequence. At each position, the oscilloscope DC voltage level on CHANNEL A should not be less than +3 volts. If less, the module has failed.

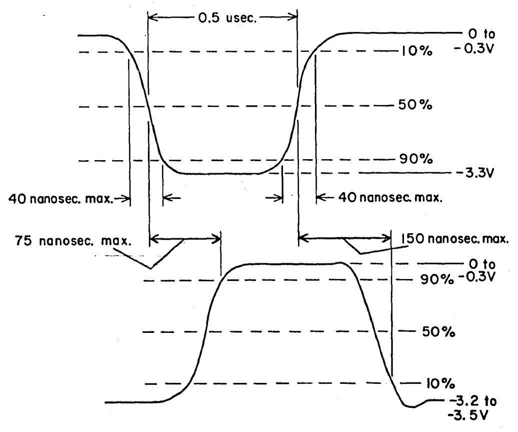
TRANSISTOR LEAKAGE TEST

1. Set test set INPUT-OUTPUT SELECT switch to 1.
2. Set oscilloscope preamp for CHANNEL B operation.
3. Set oscilloscope preamp VERTICAL SENSITIVITY control for CHANNEL B to 0.5 volts/division.
4. Oscilloscope waveform on CHANNEL B should be DC level between -3 volts and -15 volts. If not, the module has failed.

CLAMP DIODE LEAKAGE TEST

1. Set test FUNCTION SWITCH B to 3.
2. Waveform on CHANNEL B should be a DC level between -3 volts and -15 volts. If not, the module has failed.
3. Remove module under test from TEST JACK.

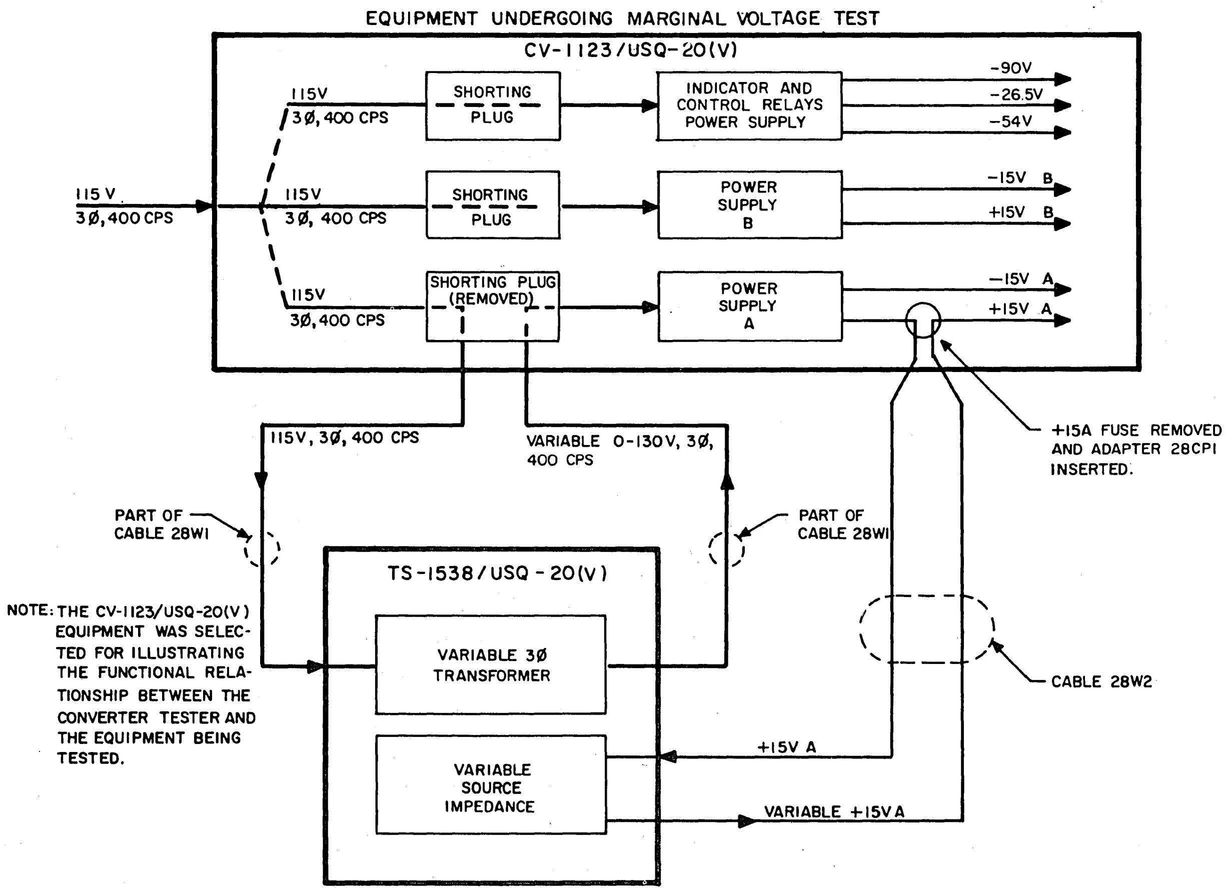
WAVEFORM A



WAVEFORM B

Table 16-3. —Equipments Tested With TS-1538/USQ-20(V).

NAME	DESIGNATION	INSTRUCTION MANUAL
Digital Data Converter (Keyset Central)	CV-1123/USQ-20(V)	NAVSHIPS 94093
Digital Data Introducer (Keyset)	MX-3195(V)/USQ-20(V)	NAVSHIPS 94097
Signal Data Recorder-Reproducer (Magnetic Tape System)	RD-243/USQ-20(V)	NAVSHIPS 94091
Teletypewriter Set, Modified With Adapter (TTY, UGC-6)	AN/UGC-6	NAVSHIPS 94084
Teletypewriter Set, Modified With Adapter (TTY, UGC-13)	AN/UGC-13	NAVSHIPS 94104
Digital to Analog Converter Group (IDAC)	AN/SYA-3	NAVWEPS OP-3160
Digital Data Converter (Video Processor)	CV-760/SS	NAVSHIPS 94099
Data Line Terminal (TEL)	MX-3502/USQ-20(V)	NAVSHIPS 94092
Introducer Tester	TS-1539/USQ-20(V)	NAVSHIPS 94098(A)


EQUIPMENT UNDERGOING MARGINAL VOLTAGE TEST
CV-1123/USQ-20(V)
115V 3Ø, 400 CPS
SHORTING PLUG
INDICATOR AND CONTROL RELAYS POWER SUPPLY
-90V
-26.5V
-54V
115 V 3Ø,400 CPS
115V 3Ø, 400 CPS
SHORTING PLUG
POWER SUPPLY B
-15V B
+15V B
115V 3Ø,400 CPS
SHORTING PLUG (REMOVED)
POWER SUPPLY A
-15V A
+15V A
+15A FUSE REMOVED AND ADAPTER 28CPI INSERTED.
115V, 3Ø, 400 CPS
VARIABLE 0-130V, 3Ø, 400 CPS
PART OF CABLE 28WI
PART OF CABLE 28WI
TS-1538/USQ-20(V)
VARIABLE 3Ø TRANSFORMER
VARIABLE SOURCE IMPEDANCE
+15V A
VARIABLE +15V A
CABLE 28W2
NOTE: THE CV-1123/USQ-20(V) EQUIPMENT WAS SELECTED FOR ILLUSTRATING THE FUNCTIONAL RELATIONSHIP BETWEEN THE CONVERTER TESTER AND THE EQUIPMENT BEING TESTED.

124.229

Figure 16-18.—Functional relationship between the converter tester and the equipment being tested.

CHAPTER 17

MAINTENANCE INFORMATION

To meet reliability requirements and provide efficient trouble-shooting procedures special maintenance techniques have been developed for large computers. Maintenance program tests provide loop circulation of a simulated data pattern. The type of testing will determine whether the test signal becomes distorted in the circulation process and whether the control circuits in the loop allow passage of the entire succession of signals. Marginal checking circuits are included in some computers to detect aging of the circuit parts before a failure occurs. In addition to these special techniques normal signal-tracing methods are used to locate the specific circuits in which malfunctions exist. Basic measurements are used to detect faulty parts.

MAINTENANCE PROGRAMS

Computers can be given an overall check by means of maintenance programs. A maintenance program provides a thorough and rapid method for you to detect failure in a special portion of a computer. This type of overall maintenance check is very flexible and efficient. These programs use the same type tape, memory, computing, and drum circuits as the operational programs. A program can be changed when the computer or auxiliary components are changed, and the program can be constantly improved. No extra test equipment is required since the computer circuits are used to perform the test. Testing by means of maintenance programs also results in the computer circuits being used in a more normal manner than during signal-tracing procedures. When a program has been checked and accepted as a good maintenance tool, it is not subject to deterioration. In contrast, test equipment may be checked and accepted only to become unreliable shortly after being placed in actual use.

Maintenance programs are divided into three main classes: reliability, diagnostic, and utility programs. Maintenance programs that are used to detect the existance of errors are called reliability programs. Reliability programs should be arranged to check as many computer circuits as possible. Maintenance programs that are used to locate the circuits in which computer malfunctions originate are called diagnostic programs. An effective diagnostic program should locate the source of trouble as closely as possible. Actually, in many cases reliability programs have diagnostic features, and diagnostic programs have reliability features. For convenience, a program is called either a reliability or diagnostic program depending on its intended emphasis. In general, programs that check rather than diagnose are shorter and simpler.

BASIC PROGRAMS

A program is a series of instructions which control the operations of a computer. Each instruction is used to cause some action which is a part of the overall task that the computer must perform. Therefore, an instruction may be considered to be the basic building block of a computer program.

An efficient program makes full use of the instructions which are available to accomplish the task in the shortest possible time and uses the least number of instructions. In most cases, on criterion, either time or the number of instructions, has to be chosen over the other, and the program is developed along this line. If time is important, you should try to write a maintenance program which used instructions of short duration but may use quite a few memory locations for storage. On the other hand, if time is relatively unimportant, but only a few locations are available, you must choose instructions which do a number of things or cause the computer program to run through the same program more than once.

To write a satisfactory maintenance program, it is necessary to have a thorough knowledge of the instructions that can be used. This includes execution time, the overall purpose of the instruction, when the instruction may be used, and the state of the computer after the instruction has been carried out. In addition, you should know whether the instruction can be indexed and what internal conditions must be satisfied before it can be executed.

Halt Instruction

The halt instruction causes the computer to stop executing instructions under program control. However, any operation which is in progress at the time the halt instruction is decoded will be completed first. For example, if information is being read into the memory from a deck of punched cards, all the cards will be read before the computer halts, even if the halt instruction was issued just after the reading operation began. The address portion of the halt instruction is not used; therefore, indexing is not possible. When the computer is halted by instruction, the program counter retains the address of the instruction immediately following, so that restarting the computer will cause this next instruction to be executed.

Clear and Add Instruction

The clear and add instruction is used to enter a quantity into the accumulators from the memory section without changing the sign or magnitude of the words. This instruction is normally used when it is desired to begin a type of addition problem. The accumulators are first cleared, and then the information selected from the memory section by the address portion of the clear and add instruction is transferred to the computer registers. Addition of the data in the registers and the data in the accumulators is started; however, since the accumulators are cleared to +0, this addition has the overall effect of transferring the word from the memory section into the accumulators unchanged. The memory area used is unchanged, and the registers are cleared to +0 after execution of the clear and add instruction.

Add Instruction

This instruction is similar to the clear and add instruction except that it does not provide for clearing the accumulators before the addition process begins. Thus, the add instruction will generate the sum of the data contained in the specified memory address and the data that is in the accumulators. This sum is placed in the accumulators, and the registers are cleared to +0. It should be noted that the add instruction can cause an overflow if the numbers added together are sufficiently large. If this happens, the result in the accumulator is meaningless.

Full Store Instruction

The full store instruction is used to transfer words from the accumulators into the memory area specified by the address portion of the instruction. Thus the results of any operation performed by the arithmetic section are placed in the memory section for future use. The contents of the specified register are first cleared, and then the contents of the accumulators are transferred to the memory section buffers.

RELIABILITY PROGRAMS

Reliability programs are used in both preventive and corrective maintenance tests to detect circuit failures rapidly and to discover failures that may occur only under particular operating conditions. Examples of troubles that are not evident at all times are failures that appear at specific repetition rates or for certain combinations of bits. In order to detect such failures, it is necessary to use reliability programs which check logical operation, paths of information flow, timing, ability of the computer to perform all functions, execution of instructions, etc.

TYPES

Reliability programs check either the logical functioning of an entire computer section or the logical functioning of individual circuit groups in a section. Whichever method is used, it is assumed that associated circuits which are not directly checked by the program are in satisfactory operation condition. Thus these programs can be considered to fall into two categories, first order and second order. First-order reliability programs check the operation of assemblies or circuit groups, such as registers, counters. In most cases, first-order programs are merely a combination of several second-order programs.

Interpretation

A reliability program provides a good-or-bad indication regarding the ability of the tested computer section or circuit to perform its operating functions. For example, consider a reliability program that checks the switching time for relays within a specific section of a computer. As long as the switching time of the relays is within normal limits, the reliability program will indicate satisfactory operation. When the switching time is excessive however, there is an indication that maintenance is required. If the program runs successfully, there are no failures within the checked area. In the event of a failure indication, the failure may be in the area being checked or in another area that has been assumed free of trouble. Diagnostic maintenance programs should then be used to locate the source of trouble.

DIAGNOSTIC PROGRAMS

To be efficient, maintenance programs for diagnostic applications must enable you to narrow the area of a failure down to the smallest possible number of circuits. This can be accomplished by employing increasing-area, decreasing-area, overlapping area and large-area checks. The most effective method will depend on the particular type of computer being tested.

Increasing-Area Check

A maintenance program using the increasing-area check initially tests a small number of circuits. If a check indicates that all tested circuits are operating properly, successive checks are run in which progressively greater numbers of circuits are added. By this method, circuits which are found to be operating correctly are used to check other circuits. This process is continued until all circuits that can be checked by a maintenance program have been tested.

Decreasing-Area Check

When this method is used to find a trouble, a large number of circuits are initially checked by the maintenance program. If trouble is detected in a large area, additional checks are made of successively smaller portions of the equipment until the stages affected by the failure are not included in the test area. You should then be able to determine which stages are defective. If the check of a large area reveals no error, the remaining large areas of the equipment are checked until the trouble is detected. In many cases, trouble can be located more rapidly by this procedure than by the increasing-area method.

Overlapping-Area Check

Another efficient method of locating trouble to within a small section of the equipment is the overlapping-area check. The routines of this type of maintenance program overlap each other. Thus, a failure is located at the overlapping portions of the routines which indicate the presence of trouble.

Large-Area Check

You may not be able to program an effective maintenance test for some small sections of a computer. A maintenance program can then be used only to detect the general area in which the malfunction occurs. When the general area is located, conventional troubleshooting will be necessary to find the circuit in which a failure has occurred.

UTILITY PROGRAMS

Utility programs are used as aids for both operation and maintenance programing procedures. This type of program is used to print out information from magnetic cores, magnetic drums, or other storage devices within the computer memory section. It is also used to transfer maintenance programs from punched cards or magnetic tape into the computer memory section. Utility tracing programs provide a printed record of the contents of various computer registers to enable you to follow maintenance program operations.

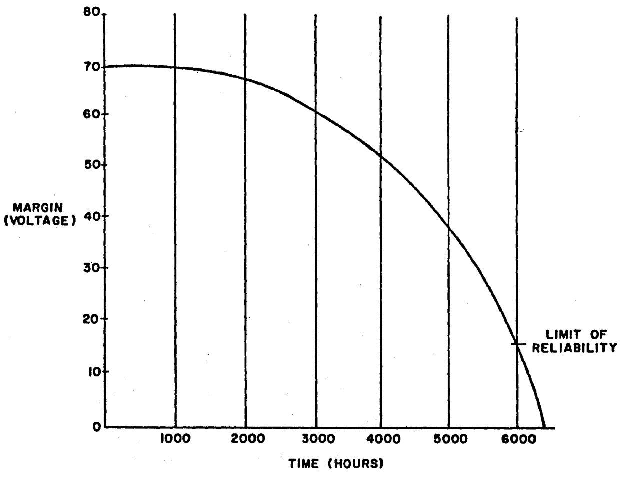
MARGINAL CHECKING

Marginal checking is a preventive-maintenance technique that is used for some Navy and commercial computer equipment to detect the decrease in reliability of circuit parts due to aging. Aging circuit parts almost invariably change in value, current-handling capabilities, or voltage limitations. Generally, the changes brought about by aging are gradual, and you will not notice any variation in the normal operation of the equipment. For maximum equipment reliability, parts that are beginning to deteriorate must be detected and replaced before a failure occurs.

Marginal checking is usually controlled by a maintenance program. The program directs the computer to perform the normal computer operations of addition, subtraction, etc, while the program varies certain circuit parameters about their normal values. In this way, the computer is made to perform normal functions under adverse operating conditions.

To accomplish marginal checking, certain operating conditions are changed from their normal values. Since circuit part values normally change with age, the variations that can be introduced before a failure occurs become less as the parts age. The amount of variation, from the normal value, that can be introduced before equipment failure occurs is called the margin of reliability of the circuit or group of circuits being tested. If the margin is regularly checked and its gradual decrease noted, as shown in figure 17-1, the time of circuit failure can be anticipated. Three methods of changing operating conditions for marginal checking are as follows:

1. Variation of d-c supply voltages.
2. Variation of circuit values.
3. Variation of electron-tube filament voltages.



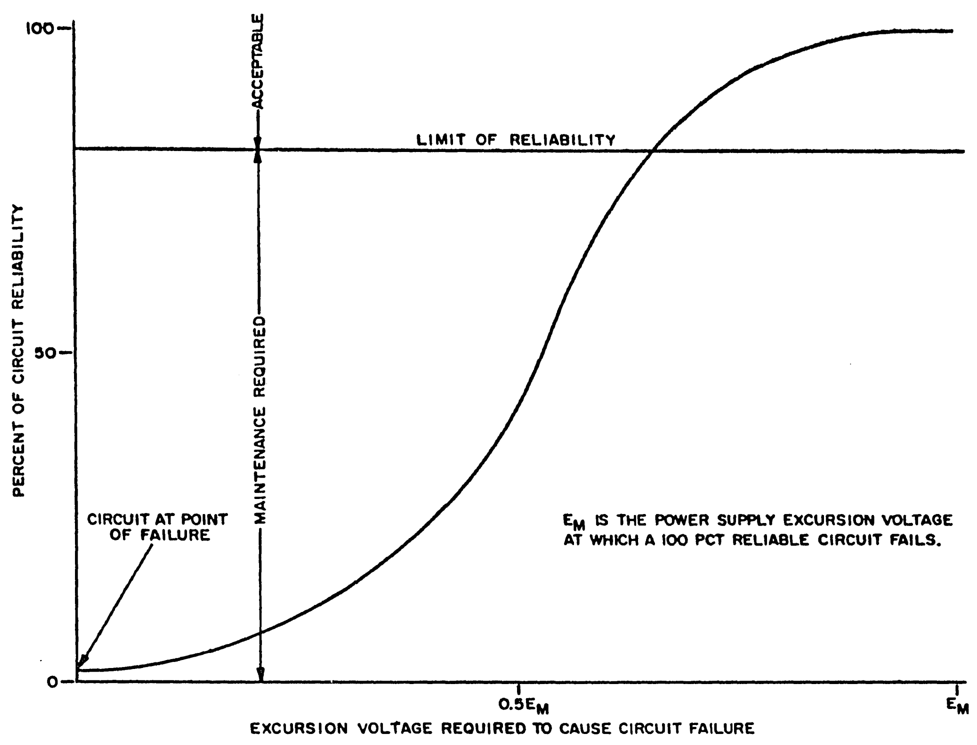
124.230
Figure 17-1.—Typical circuit part, life curve.

D-C SUPPLY VOLTAGE VARIATION

The most versatile method of marginal checking is by variation (excursion) of the d-c supply voltage for one or more circuits. Causing an excursion of a circuit's d-c supply voltage will simulate the changes that normally result from the aging of circuit parts. Gradually increasing the excursion of the supply voltage to a circuit will eventually result in a circuit failure regardless of the circuit's age. Figure 17-2 shows the relationship between circuit reliability and the excursion voltage required to cause a circuit failure. The magnitude of the voltage excursion necessary to cause a failure is called the margin of the voltage on the circuit. This margin becomes smaller as the circuit ages. When the circuit fails at the normal operation voltage, the margin is zero.

As long as the possibility of circuit failure is low the circuit is considered satisfactory. For the example shown in figure 17-2, the circuit reliability of 80 percent is acceptable. When the voltage excursion necessary to cause a circuit failure decreases so that the circuit reliability is below the 80-percent value, maintenance must be performed to replace parts or an entire plug-in assembly. The level at which the circuit reliability is acceptable must be determined for each circuit or circuit group that is tested by marginal-checking methods.

A life curve for a circuit can be drawn, as shown in figure 17-1; you can then estimate the time for a circuit to age to the point where failure is expected. Maintenance will be performed on the equipment when the marginal check indicated that a failure will probably occur before the next marginal check is due. Such a life curve is useful for maintenance application when the margin limit is approached gradually during each successive testing period. Certain circuits will not have useful life curves because they age gradually but fail suddenly. These circuits may operate for long periods with practically no change in margin, and then the margin drops sharply in a short period of time. Since no appreciable decrease in margin takes place before the circuit fails, it is difficult to determine when a faulty part should be replaced and marginal checks are not effective. The circuits for which useful life curves cannot be established, however, represent only a small percentage of general computer circuits. Most circuits have life curves that enable you to



124.231

Figure 17-2.—Circuit reliability versus excursion Voltage required to cause circuit failure.

determine the time at which a circuit part must be replaced to prevent failure during computer operating time.

CIRCUIT PART VALUE VARIATIION

Failure of circuit parts other than electron tubes can be anticipated by periodically simulating the aging of the parts. Aging can be simulated by changing register, capacitor, or inductor values of a circuit and noting the effects of such changes. This can be done when adjustable electrical parts are included in the circuits to be tested. By periodically measuring the amount of change necessary to cause circuit failure, and by determining how much normal component aging will produce such a change, it is possible to establish the time at which a part must be replaced to prevent circuit failure before the next scheduled test.

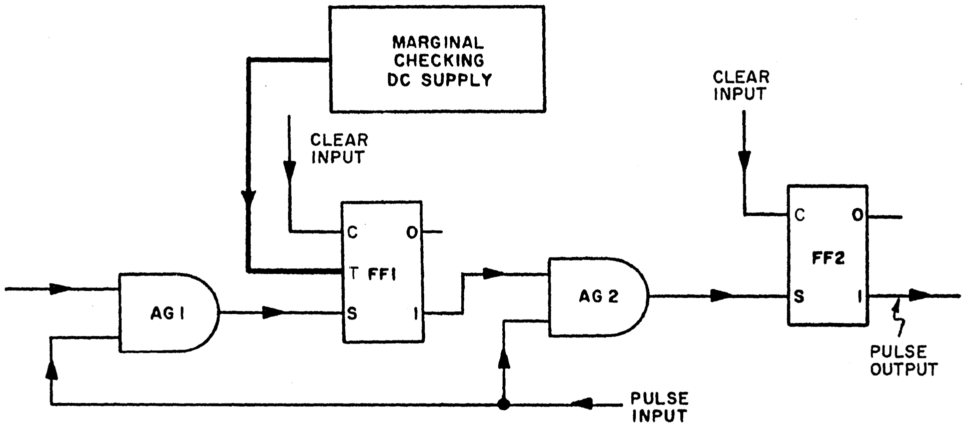
FILAMENT VOLTAGE VARIATION

Reducing the filament voltage of electron tubes is a method used to simulate a condition of low cathode emission which causes circuit failure. Anticipation of this condition enables you to replace a deteriorating tube before failure occurs.

For marginal checking applications, low cathode emission can be simulated by periodically decreasing the filament voltage from its normal operating value to a value at which useful emission ceases. Each time a marginal check is made, the difference between the normal filament voltage and the filament voltage at which the equipment fails is noted. Over a period of time this difference, or margin, will become smaller. Eventually, a point is reached where the margin is so small that you can expect emission failure before the next periodic check. The weak tube should then be replaced to prevent its failure during normal operating time. This method of marginal checking is seldom used because it is limited to the prevention of computer failures caused by tube deterioration.

PROCEDURE

Figure 17-3 represents a circuit that can be selected by a maintenance program for



124.232
Figure 17-3.—Typical circuit selected for marginal checking, logic diagram.

marginal checking. The operation of this circuit is such that successive pulses place a "1" in FF1, transfer it to FF2, and then clear both flip-flop stages.

During the marginal checking operation, an excursion is applied to the supply voltage line of FF1. Assume that, at some voltage value, the computer senses that a "1" was not transferred to FF2. The excursion is stopped and the margin noted. To determine which stage has failed, you might first make voltage and resistance checks of the circuit parts of FF1. If this check indicates the FF1 is functioning correctly, it is possible that the output pulse from gate AG1 is so small that a slight change in the supply voltage applied to FF1 causes the circuit to fail. It is also possible that gate AG2 has aged to the point where any decrease in the signal output level of FF1 will result in failure of the pulse to be transferred to FF2.

MARGINAL CHECKING UNITS

Marginal checking units for variable supply voltage applications include a voltage-regulator unit, relays to connect the test voltage to circuits, and control circuits necessary to initiate marginal checking. When high currents must be furnished by the regulator, an amplidyne may be used to provide a variable d-c supply voltage. This device provides a well-regulated output voltage that can be accurately controlled over a wide positive-to-negative range.

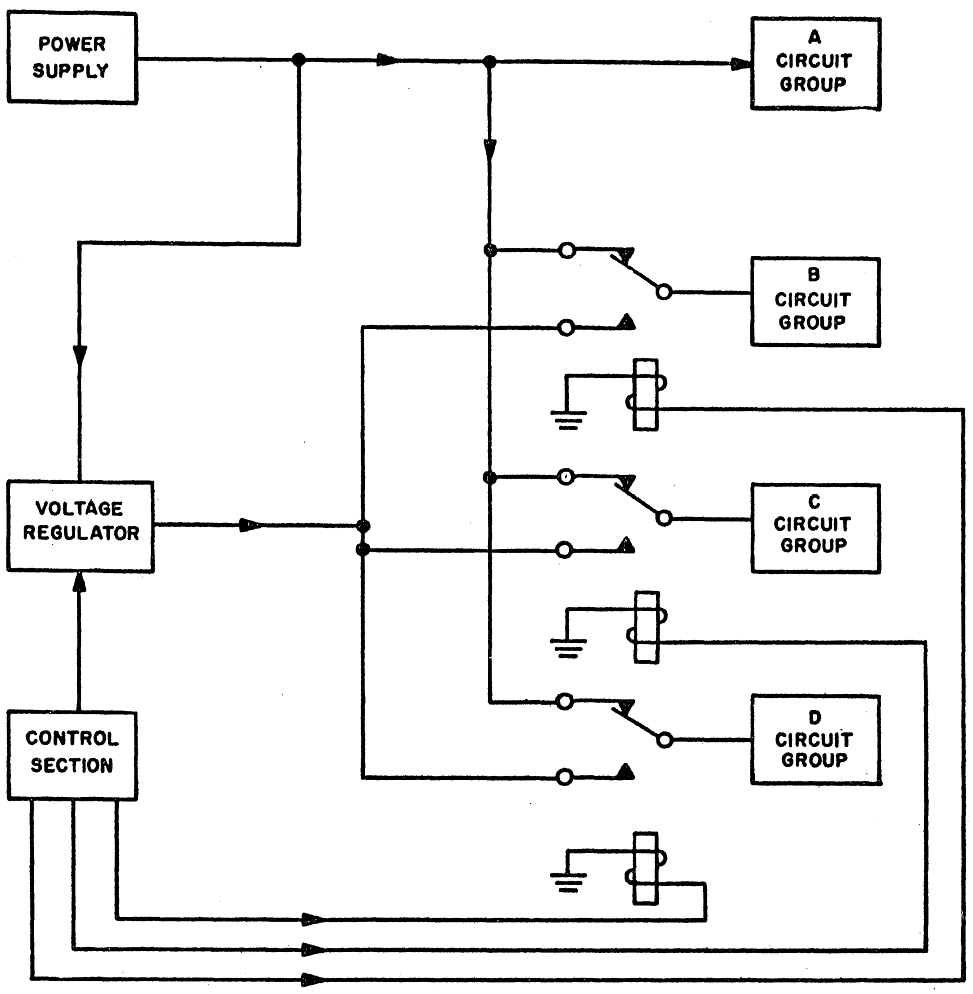
A simplified block diagram of marginal checking units is shown in figure 17-4. The relays, which are controlled by maintenance programs, can connect any one of three circuit groups to the marginal-checking voltage regulator. Circuit group A represents circuits that cannot be effectively tested by this method; therefore, it is not connected to a relay in the marginal checking circuit.

COMPUTER DIAGRAMS

Logic diagrams are usually included with computer maintenance instructions to show the paths of data flow through the computer circuits. The use of logic diagrams will often simplify overall circuit tracing and thus help you to locate trouble in equipment that has many circuits. When you have determined that a trouble is in a specific circuit or group of circuits, conventional schematic diagrams should be used as an aid to locate faulty parts.

LOGICAL SYMBOLS

Symbols used in logic diagrams are geometric figures that represent the various types of computer circuits. Common electrical symbols are used to represent circuits and devices such as amplifiers, rotating machinery, and test points. Symbols peculiar to computer applications are used for circuits and devices such as magnetic drums, flip-flop stages, and gating circuits. Basic symbols that are used in technical manuals are shown in figure 17-5, and



124.233

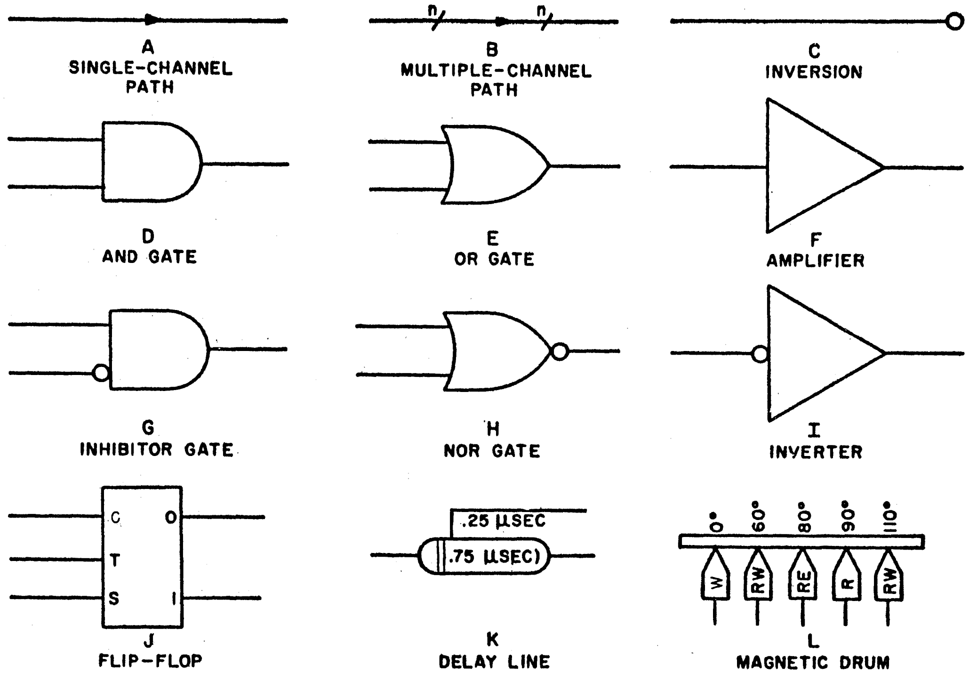
Figure 17-4.—Marginal checking units, simplified block diagram.

common reference designations for these symbols are listed in table 17-1. Variations of these symbols are often used in commercial and older Air Force technical manuals, and abbreviations are usually added to the symbols. Therefore, many computer technical manuals that include logic diagrams with nonstandard symbols contain tables of symbols and abbreviations.

Each symbol represents a function or, in special cases, a combination of functions. The symbols are interconnected by lines that indicate information or control signal paths but not actual wire connections. As a rule, there is no correlation between the logical functions and the individual modules of a computer. One module may be used to perform several logical

Table 17-1.

Designation	Name
AG	And Gate
CT	Counter
FF	Flip-Flop
OS	One Shot
OR	Or Gate
RG	Register
ST	Schmitt Trigger
SR	Shift Register



124.234

Figure 17-5.—Logical symbols for computer diagrams.

functions, or a logical function may require several modules.

Summarization logic diagrams are sometimes included in a technical manual to further simplify explanations of equipment operation. Such a diagram may show groups of circuits that perform a related function, such as an accumulator, storage register, or full adder, as one block. All signal inputs and outputs of the block are identified, and the function may be marked on the block.

Signal Paths

Single-and multiple-channel paths that connect logical symbols are shown in parts A and B of figure 17-5. A multiple-channel path is distinguished by crossties, and the number of channels is indicated by a circled number. The direction of signal flow is indicated by arrowheads superimposed on the channel-path line. You should keep in mind that these lines indicate the direction of data flow, not voltage or current.

Basic Circuits

And-gate, or-gate, and amplifier symbols are shown in parts D, E, and F of figure 17-5. These symbols can be modified by the addition of a small circle, as shown in part C of the figure, to indicate signal inversion. The inversion symbol can be placed adjacent to the inputs or outputs of symbols to represent circuits such as inhibitor gates, nor gates, and inverters, as shown in parts G, H, and I of the figure.

The symbol for a flip-flop multivibrator is shown in part J of figure 17-5. This circuit is a bistable multivibrator that has three possible inputs: set S, reset C, and trigger T; and two possible outputs: "1" and "0". The flip-flop circuit assumes the "1" state when an effective signal appears at the S input; it assumes the "0" state when an effective signal appears at the C input. When an effective signal appears at the T input, the flip-flop output state is reversed. If signals appear simultaneously at more than one input, you can not be certain which will trigger the circuit and the output has no meaning.

Common Devices

The symbol in part K of figure 17-5 represents a passive delay device such as an electromagnetic or ultrasonic line. The total duration of the delay is written on the symbol. If the delay device is tapped, as indicated in the figure, the duration of the delay from the input to the tap is written within parentheses adjacent to the tap output. These delay lines are used for both timing and memory applications.

Part L of figure 17-5, shows the symbol for a magnetic drum assembly. The functions of the heads are indicated by the following abbreviations: reading only, R; writing only, W; erasing only, E; reading and writing, RW; and reading and erasing, RE. In some diagrams the angular relationship of the heads is marked in degrees, as shown in the figure.

Summarization Symbols

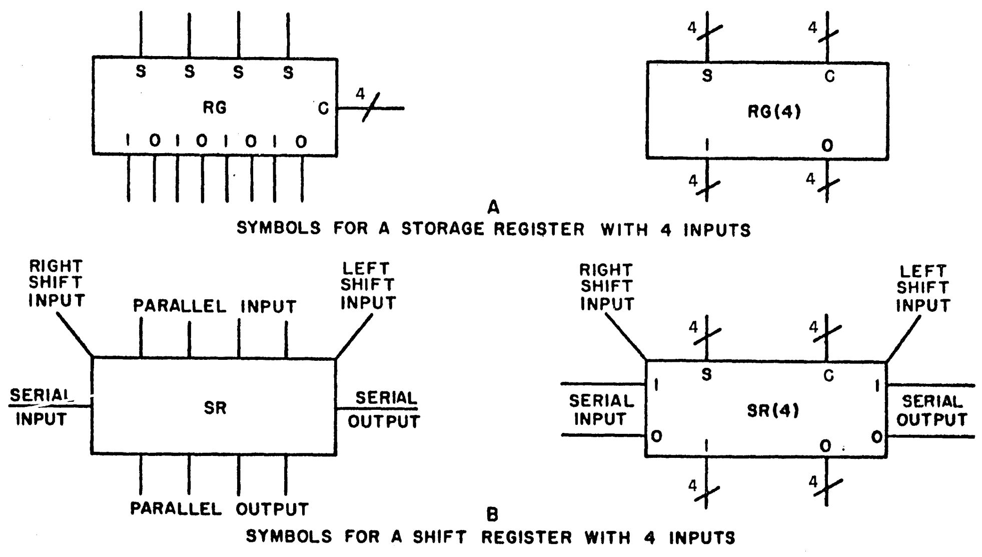
Figure 17-6 illustrates summarization symbols that are used for binary registers. Part A of the figure shows a storage register consisting of a group of four flip-flop circuits connected in parallel. The symbol on the left side shows the individual input and output paths. In the symbol on the right side, the four S inputs, the four "1" outputs, and the four "0" outputs are represented by multiple-channel lines.

Part B of figure 17-6 shows a four-stage binary register; the contents of this register can be shifted, one stage at a time, to the right or left when triggered by a shift input signal. The symbol on the left side shows four individual parallel input and output paths. The symbol on the right side combines the parallel S inputs, C inputs, "1" output, and "0" output into multiple-channel lines. Figure 17-7 is a more detailed logic diagram of a four-stage flip-flop shift register. This diagram also represents different functions by a single symbol since gate circuits may be included as part of a flip-flop symbol.

STYLIZED PULSE WAVEFORMS

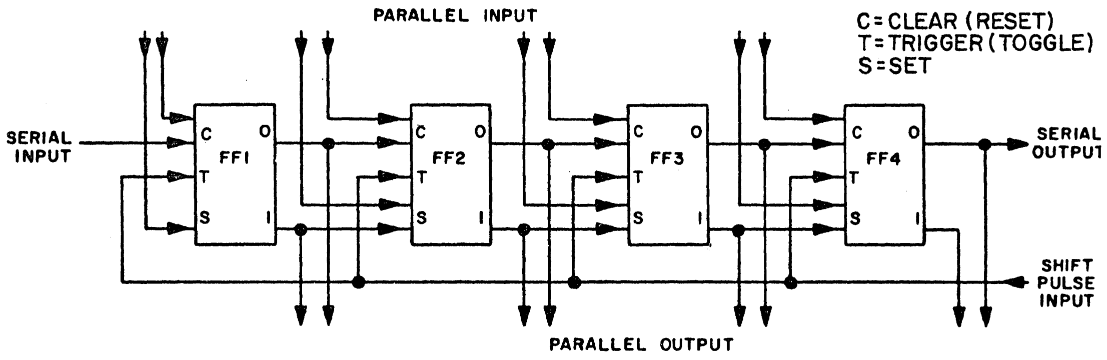
Pulse waveforms are used on logic diagrams to indicate the nature and timing of the signals. Waveform symbols are used to represent the signal level, time of occurrence, beginning and ending times, and rise and fall times, as shown in figure 17-8. Examples of positive-and-negative-going pulses are shown, and representative voltage and time values are used. A pulse may be represented as a single or double line, and, in the case of information-bearing pulse trains, not all pulses are necessarily present.

Figure 17-9 shows a representative pulse waveform with level and time designations that are used in computer manauls. The letters Vd represents the maximum voltage amplitude of the pulse. The rise time, tr, is the time required for the beginning of the pulse to increase from 0.1 x Vd to 0.9 x Vd. The pulse duration, td, is the time that the pulse is equal to or greater than 0.9 x Vd. The fall time, tf, is the time for the end of the pulse to decrease from 0.9 x Vd to 0.1 x Vd.



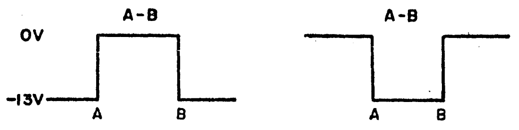
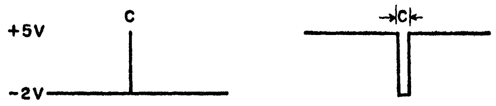
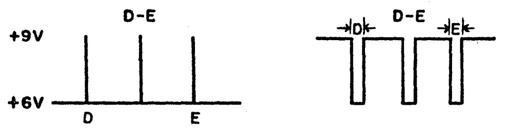
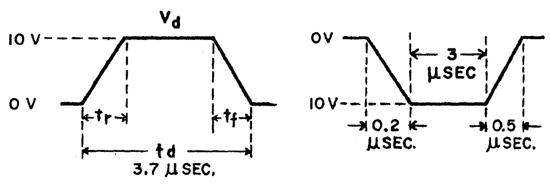
124.235

Figure 17-6.—Summarization-type logical symbols.



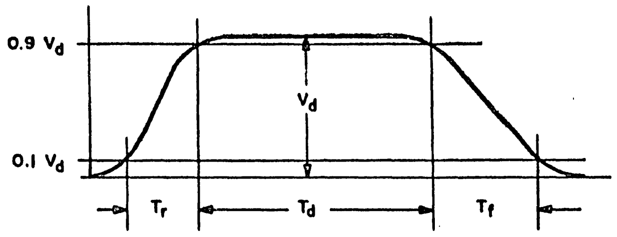
124.236

Figure 17-7.—Flip-flop shift register, detailed logic diagram.

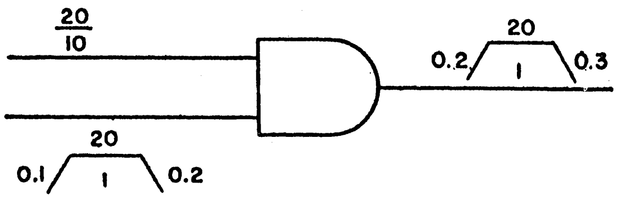
WAVEFORM NOTATIONS	DESCRIPTION
	PULSE STARTS AT TIME A AND ENDS AT TIME B
	PULSE OCCURRING AT PULSE TIME C
	PULSE TRAIN WITH FIRST PULSE OCCURRING AT PULSE TIME D AND LAST PULSE AT PULSE TIME E
	PULSE WITH VOLTAGE LEVEL V_d, DURATION t_d, RISE TIME t_r, AND FALL TIME t_f
$\frac{V_d}{t_d}$ $\frac{10}{3}$	PULSE NUMERICAL RELATIONSHIP USED WITHOUT A WAVEFORM SYMBOL

124.237

Figure 17-8.—Logic diagram pulse waveform notations.



124.238
Figure 17-9.—Representative pulse waveform.



124.239
Figure 17-10.—Waveform designations for gating circuit.

Figure 17-10 shows the waveforms and notations that may be used with an and gate circuit. Two pulse signals with amplitudes of 20 volts and durations of 1 and 10 microseconds are applied to the gate. The resulting output is a pulse signal having an amplitude of 20 volts and a duration of 1 microsecond. The output pulse rise time is 0.2 microsecond, and the fall time is 0.3 microsecond.

LOGICAL EQUATIONS

Equations listed on logic diagrams provide additional information regarding the signal and circuit. Such logical equations may indicate the type of gate circuits used and the conditions that must be present to produce an output signal from a particular gate. Figure 17-11 shows a schematic diagram, the related logic diagram, and the logical equation of a diode netword with and-gate and or-gate circuits. The equation is read, E is equal to a binary 1 if A and B are 1 or C and D are 1. The equation consists of two and functions that form an overall or function. Only one of the functions must equal a binary 1 for E to equal 1.

Other relationships that can be expressed in equation form include signal designations with switch numbers and positions marked above them as follows:

$$S_1 - N \quad S_1 - 1 \quad S_1 - 0$$

$$a_1 = A + \text{GND} + (-13v)$$

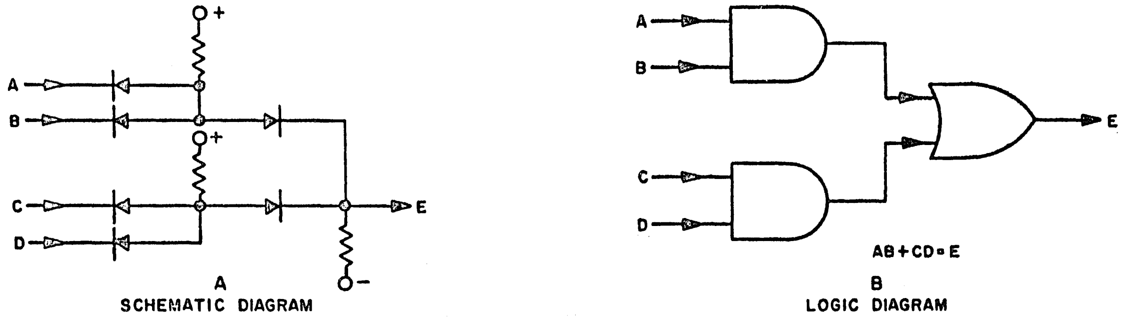
This equation means: signal a_1 is the same as A (-13 volts or 0 volts) if switch S_1 is in the N position, or signal a_1 is at ground potential (0 volts) if S_1 is in the 1 position, or signal a_1 is at -13 volts if S_1 is in the 0 position.

Flip-flop inputs can also be expressed as a logical equation. A flip-flop stage with and-gate circuits in the input might have the following equation:

$$FF_1\colon 1 = f_1 f_2 f_3$$

$$0 = f_4 f_1 f_5$$

That is, the 1 -side inputs to flip-flop stage FF_1 are f_1 and f_2 and f_3, and the 0-side inputs are f_4 and f_1 and f_5.



124.240
Figure 17-11.—Diode gating network diagrams.

ALIGNMENT

Circuit adjustments may be necessary when input or output units are either connected or disconnected from a computer. These adjustments may vary the gain of amplifier stages, the frequency of a timing circuit, or the voltage furnished by a power supply. The need for other adjustments will be indicated by preventive-maintenance checks. Power-supply outputs must be adjusted to furnish the correct current to magnetic cores and to correct voltage to amplifier and oscillator stages. The speed of magnetic drum and tape devices must be adjusted to provide computer signals having the correct pulse width and timing.

CORE MEMORIES

Core memory circuits may require realignment because of the following reasons:

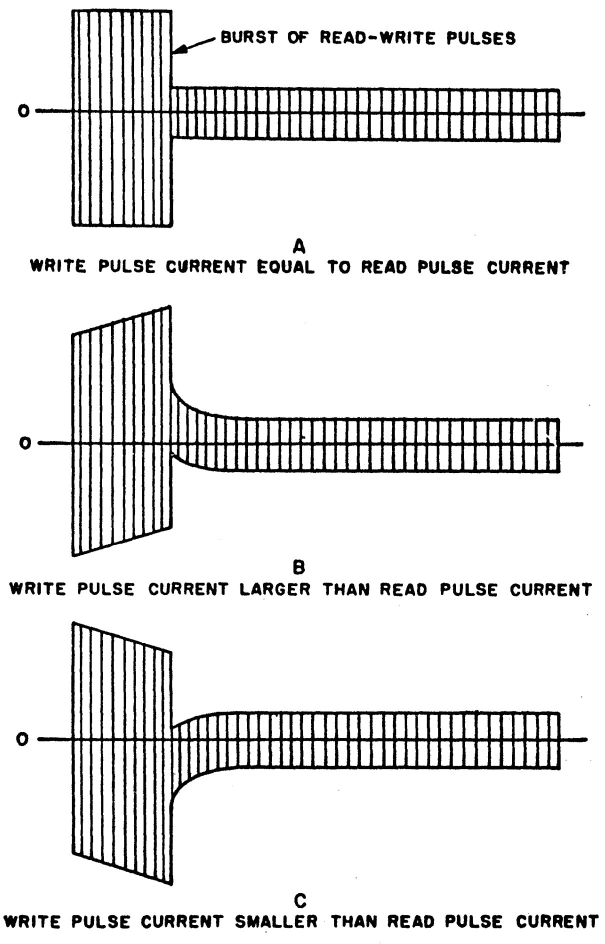
1. Changes in operating currents due to the aging of parts.
2. Replacement of plug-in units, such as digit plane drivers, memory gate generators, or timing stages.
3. Accidental changing of the original potentiometer settings.

The procedures for adjusting a core memory require checks of the memory timing cycle, the digit-plane-driver current, the read current, and the write current.

The memory timing cycle can be measured with an oscilloscope calibrated to measure time. The time between the clear-memory, start-memory, set-read, clear-read, set-inhibit, set-write, clear-write, and clear-inhibit pulses must be checked. If a discrepancy exists in the timing of any of these pulses an adjustment must be made. In some memory units this is done by changing the taps on the memory clock delay line.

An oscilloscope calibrated to indicate voltage can be used to determine the digit-plane-driver, read, and write currents. Maintenance programs can be used to inject the proper computer signals into the memory-coil arrays. You can then calculate the current through each drive line after measuring the voltage drop across the terminating resistor for each set of windings. Potentiometers located in the memory units are used to set the current to the correct amplitude. Recheck and, if necessary, readjust the read and write currents after the first adjustments are completed. This readjustment may be necessary because of interaction between the two circuits.

After the read and write currents have been adjusted, you can check the balance by means of an oscilloscope and, if necessary, make a fine adjustment. Connect the oscilloscope probe to a read-write drive line. Run a maintenance program to apply synchronized bursts of read and write pulses to the drive line. If the read and write currents are equal, the oscilloscope display should be similar to that shown in part A of figure 17-12. If any unbalance is present, the oscilloscope display will resemble the display shown in part B or C of the figure. Either the read or write current potentiometer should be adjusted slightly to provide a balance.


BURST OF READ-WRITE PULSES
0
A
WRITE PULSE CURRENT EQUAL TO READ PULSE CURRENT
0
B
WRITE PULSE CURRENT LARGER THAN READ PULSE CURRENT
0
C
WRITE PULSE CURRENT SMALLER THAN READ PULSE CURRENT

124.241
Figure 17-12.—Magnetic core read and write current balance check waveforms.

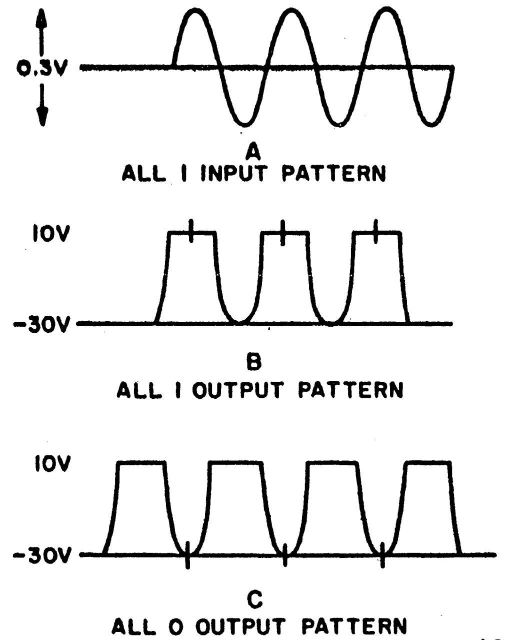
MAGNETIC DRUMS

Magnetic-drum read and write head adjustments are always required in the initial stages of installation and after replacement. When the read and write functions are accomplished by two drum heads, the heads must be adjusted with respect to both signal amplitude and timing. Since these two adjustments are interacting, they must be made concurrently. The interaction of these adjustments results from the construction and location of the heads, head bars, and rotor drum. When writing and reading are accomplished by a single drum head, the timing will always be correct and only the amplitude adjustment must be made.

The use of two oscilloscopes is recommended for drum head adjustments. One oscilloscope is connected to the input of the drum read amplifier to monitor the head amplitude, and the other is connected to the output of the drum read amplifier to provide a check of timing. Typical waveforms that may be obtained are shown in figure 17-13.

Timing Adjustment

For a coarse adjustment, loosen the head-retaining screws and move the head the desired amount. Then tighten the screws until



124.242

Figure 17-13.—Typical drum read amplifier test waveforms.

they are snug enough to retain this position. Observe the timing pattern on the oscilloscope, and use the retaining screws to rock the head for a fine adjustment. One screw should be tightened and the other loosened to tilt the head to the position which provides accurate timing. Movement of the screws in either direction should be slight. Excessive tightening of a screw may cause the head to be cocked an abnormal amount and result in a distorted input signal to the read amplifier.

Signal Amplitude Adjustment

The output signal amplitude of a read head is adjusted by varying the air gap between the drum surface and the read head core. On some magnetic drums, the read heads are adjusted to provide a signal output which is 75 percent of the amplitude that is measured when the head is in contact with the drum surface. To make this adjustment, proceed as follows:

1. Connect an oscilloscope to the output of the read head.

2. Loosen the lock nut on the amplitude screw just enough to permit a snug fit.

3. Each time an adjustment is made, tap the amplitude adjustment screw lightly to overcome static friction in the associated mechanism.

4. Turn the amplitude adjustment screw until the head core just makes contact with the drum surface. The head core should remain in this position only long enough to measure the head peak-to-peak voltage output with the oscilloscope; otherwise the head may be damaged.

5. Turn the amplitude adjustment screw until the amplitude of the head output waveform is reduced to 75 percent of the value measured in step d. For example, assuming that the contact amplitude is 300 millivolts, you should adjust the head to provide an output equal to 75 percent of 300 millivolts, which is 255 millivolts.

6. The acceptable limits for the output of a drum head is usually listed in the equipment technical manual. For the example above, the final amplitude setting must fall within the limits of 125 to 300 millivolts; that is, if the 75-percent value is higher than 300 millivolts, it must be reduced to this maximum value. If the resultant is lower than 125 millivolts, the head is probably defective and may have to be replaced.

7. If noise spikes appear at the output of the drum head at the 75 percent setting of the

amplitude adjustment, they should be minimized by lowering the amplitude below this value. However, the amplitude must not be decreased below the lower limit described in step f.

8. When the desired amplitude is obtained, tighten the lock nut, taking care not to disturb the adjustment.

Timing and Amplitude Relationship

A definite timing and amplitude relationship exists between corresponding read and write heads, and this relationship must be taken into account when adjustments are deemed necessary. If a timing error is detected in a read head and you cannot move the head far enough to obtain correct timing, the following procedure is recommended:

1. Move the corresponding write head in a direction to correct the timing error. Write a new test signal with the head in the new position.
2. Adjust the write-head amplitude. This is accomplished by removing the write-head plug from the write-head-amplifier cable connector and applying the output of the write head to the vertical input of an oscilloscope. Then using the write head as a receiver, adjust the gap between the write-head core and the drum surface to provide an amplitude equal to 75 percent of the contact amplitude. This must be within the limits specified in the equipment technical manual. The limit for the write head may be narrower than for the receive head; for this example assume the limits to be 175 and 200 millivolts.
3. Reconnect the write head to the write-head amplifier, and write a new test pattern. Readjust the read head for correct timing and amplitude. If the minimum limit remains unattainable, replace the head.

PREVENTIVE MAINTENANCE

Preventive maintenance for computer equipment is basically the same as for other electrical equipment; it consists of inspection, cleaning, preservation, lubrication, and performance checks. However, a unique feature of computer preventive maintenance is that you can use reliability programs to quickly determine whether a computer is operating properly. In some computers marginal checks can be controlled by the reliability program, or performed manually to detect deterioration of computer parts before a failure occurs.

Because of the extensive amount of wiring in large computers, you should carefully inspect wiring and cabling. Check the wires for loose or broken lacing, and frayed insulation. Check the cables and connectors for improper placement that might subject them to strains or kinks. Inspect the connectors and wires for burned or charred parts, dirt, cracks, and breaks.

RELAYS AND SWITCHES

Inspect relays to see that they are securely mounted, that the contacts are not pitted, that the springs have sufficient tension, that the armature does not stick, and that there are no signs of overheating and corrosion. You can check the action of the armature by operating the relay manually. Relay contacts can be examined with the aid of a flashlight and a mirror. Avoid bending the springs, and do not open sealed relays.

Inspect switches for loose mountings and connections. Examine the contacts for dirt, pitting, and corrosion. Test the action of the switches and see that they operate without binding. In gang and wafer switches, see that the movable blade makes good contact with the stationary member. Make sure that the stationary contact leaves spread as the movable blade slides into them. Some switches have contacts that are impossible to reach without damaging the switch assembly. Check these switches for defective mechanical action and looseness of mountings and connections.

Cleaning

Clean the exteriors of relays and switches carefully by blowing away the dust with approximately 5 psi of air pressure. If the connections are dirty, clean them with trichloroethylene. If it is necessary to remove covers from relays or switches, make sure that no dirt, lint, or other undesirable material is present that might get into the contacts. You can remove dust and lint from a switch or relay with a soft-bristled brush.

There are various types of contacts in the relays used in a computer. For cleaning purposes they are divided into hard-surfaced contacts, made of palladium or platinum and soft-surfaced contacts, which are either silver or silverplated.

Clear hard-surfaced relay contacts, when necessary, by using a flat blade in a burnishing tool. Clean the blade by wiping it with a lint-free cloth moistened with trichloroethylene, and then move the blade two or three times between the relay contacts to brighten them. Open contacts can be pressed gently together with the fingers or an orange stick to apply pressure against the blade of the burnisher. Closed contacts will usually apply enough pressure against the blade of the burnisher. Therefore, you can open the relay armature manually, insert the blade of the burnishing tool between the contacts to be cleaned, release the relay armature, and burnish.

NOTE: Avoid excessive burnishing. When too much of the contact metal is removed, the contact movement is altered and readjustment is necessary. Separate contacts carefully, when necessary, and use care never to bend the springs.

If burnishing does not correct the contact trouble, deposit a few drops of trichloroethylene, with a toothpick, on the contacts. Before the solvent has had a chance to dry, add a few more drops to flush away dirt loosened by the first application. Allow the solvent to dry on the contacts, and then burnish the contacts as described above to remove any remaining residue. Always burnish the contacts after cleaning with trichloroethylene.

Clean solid-silver contacts by inserting a strip of hard-surfaced bond paper between the contacts lightly together. Repeat with fresh strips of paper until the contact surfaces are clean. If this does not adequately clean the contacts, apply trichloroethylene as described above, and again polish them with a paper strip.

Silver-plated relay contacts are ordinarily not cleaned. However, if such maintenance is necessary, they are cleaned in the same manner as for solid-silver contacts. Extreme care must be taken not to wear away the thin silver plate.

NOTE: Never use newspaper or any soft paper, emery cloth, or highly abrasive material such as coarse sandpaper to clean relay or switch contacts.

MAGNETIC DRUM UNITS

To eliminate noise which may be generated by the rotating drum, you must periodically inspect the static grounding brush that is mounted at the end of the drum assembly. If the brush wears to less than half its original length, you should replace it. After a brush has been replaced, check to make sure that there is low resistance between the end plate of the rotor and the drum unit. This check will assure you that the brush is seated properly in the holder and is making good electrical contact.

The drum rotor, drive motor, bolt, and pulleys are all subject to mechanical wear during normal operation; therefore, they must be regularly inspected and maintained. Check the drum drive motor pulley for tightness. An indication of a loose pulley is an accumulation of belt rubber on the pulley and pulley guard. If the pulley is allowed to remain loose, the belt will be chafed and its life shortened.

TAPE DRIVE UNITS

Many digital computers include tape handling equipment. This equipment must be cleaned, lubricated, and periodically checked to ensure proper operation. Daily maintenance includes cleaning the head assembly at the beginning of each day's operation and running a short reliability maintenance program, if the computer has this feature.

Weekly maintenance may include the cleaning of items such as the friction drive clutch and the air filters. You should check the machine's forward and reverse transfer time, the high-speed rewind, creeping of the tape reel, and the tape break circuitry. You may also be able to check the ability of the equipment to reproduce test pulses, to measure the tape speed, and to measure the moving coil and erase coil currents. A more comprehensive reliability maintenance program than is used for the daily check can then be run to determine whether error-free operation can be achieved for a period of 15 minutes.

MAGNETIC TAPE

Dust and dirt can reduce the intensity of the reading and recording pulses by increasing the gap between the tape and the head. Therefore you should take care of tape in the following manner:

1. Keep tape in a dustproof container whenever it is not in use on a tape unit.
2. While the tape is on the machine, keep the container closed and store it where it is not exposed to dust or dirt.

3. Store tapes in some type of cabinet that is elevated from the floor and away from sources of paper or card dust. This should minimize the transfer of dust from the outside of the containers to the tape reel during loading and unloading operations.

4. Do not use the top of tape units as a working area. Placing materials on top of the units may expose the tape to heat and dust from the blowers in the unit.

5. When identifying tape reels, use a material that can be removed without leaving a residue. Adhesive stickers, that can be easily applied and removed, are satisfactory. Never alter a label identification by means of an eraser, since particles from the eraser may come in contact with the tape.

6. Inspect tape containers periodically and remove any dust by washing with a regular household detergent.

7. When necessary to clean tape, gently wipe it with a clean, lint-free cloth moistened with an approved cleaning fluid, such as trichloroethylene.

Recorded information normally comes very close to the edge of a tape. Therefore, for proper operation, the edges of the tape must be free from nicks and kinks. To accomplish this, observe the following precautions:

1. When you remove a tape reel from a recorder, handle the reel near the hub whenever possible. If there is resistance in removing the reel, press it from the rear with your hands as near to the hub as possible. Under no circumstances should the reel be racked by grasping the outer edge.

2. Avoid throwing or dropping reels, and do not make contact with the exposed edge of the tape. Dropping a reel can easily damage both the reel and the tape. The use of a reel and tape after the reel has been dropped is usually unsatisfactory. Therefore, never throw or mishandle reels, even while they are in their containers.

3. When mounting reels on the recorder, push them firmly against the stop on the mounting hub to ensure good alignment.

4. When placing the tape on the takeup reel, carefully align the tape to prevent damaging the edge on the first few turns.

5. If a tape break occurs, wind the resulting pieces onto two smaller reels. Splicing is not recommended unless it is necessary to make a temporary splice to recover information.

Magnetic tape is sensitive to changes in humidity and temperature. Recommendations for tape storage are as follows:

1. If possible, store the tape in the computer room where it is to be used. Location of tape storage near the tape drives reduces both handling and variations in atmospheric conditions.

2. The storage area should be kept at a temperature of 65° F to 80° F and a relative humidity of 40 to 60 percent.

3. If the tape must be removed from the computer room, you can hermetically seal it in a plastic bag to reduce the effect of temperature and humidity changes on the tape's physical dimensions. If the tape is not hermetically sealed, before it is reused it should be allowed to remain in the computer room for a length of time equal to the time it was out of the computer room. If the tape is out of the computer room for a period longer than 24 hours, it should be conditioned to the computer room for 24 hours before being reused.

4. For long-term storage, enclose the reel and container in a hermetically sealed plastic bag. Store in an area of constant temperature. Either cold or hot temperature can harm tape. A temperature between 40° F and 120° F is satisfactory.

TROUBLESHOOTING

Computer circuits such as multivibrators, clock oscillators, and gates can be tested in the same manner as similar circuits in radar, radio, and multiplex carrier equipment. Test equipment such as oscilloscopes, voltmeters, ohmmeters, frequency meters, tube testers, and transistor testers can be used to determine which parts are actually defective and, if possible, what has caused the failure. However, large computers have such enormous numbers of circuits that efficient trouble shooting to locate defective circuits can be accomplished only with the aid of maintenance programs, built-in test equipment, and special test sets designed to be used with specific computers.

In computers which feature maintenance programs, reliability checks may be used to determine the area of a computer in which a failure has occurred. You can then run diagnostic programs to localize the trouble to a group of circuits or plug-in units. For example, if trouble appears to be in a plug-in matrix assembly, the suspected matrix assembly can be replaced by a unit known to be operating correctly. If this results in proper computer operation, you may then be able to test the defective matrix assembly with the aid of a matrix test to locate the defective parts. If a special matrix test set is not furnished for the computer, the assembly must be tested with general-purpose test equipment to find the defective parts.

Test equipment such as test-pattern generators, trouble indicators, and memory units are furnished with some computers as built-in equipment. Test-pattern generators can be used for signal-tracing applications or to provide a simulated signal for maintenance purposes. Visual and audible alarms are provided to indicate errors in calculations or the existence of conditions that might cause damage to the computer. Neon indicators are often connected to flip-flop stages to indicate the status of registers. By observing the indicators and alarms, you can often decide what action should be taken to correct a malfunction.

MEMORY UNITS

Trouble within a memory unit can usually be located by means of diagnostic maintenance programs. An example of a troubleshooting procedure is as follows:

1. Apply power to the computer and prepare the computer for test mode operation.
2. Check the supply voltages to the memory units; if they are not set properly, adjust them to their correct values.
3. Determine which memory unit is faulty by analyzing the computer error indicating, program error, or error printout.
4. Run a diagnostic maintenance program. When an error printout occurs, check the computer condition indicators to determine in which routine the failure occurred.
5. Set the maintenance memory unit to repeat the routine which produced the first error. Run this routine to make sure that the error will occur when the routine is run separately. If the test routine does not produce the error when run separately, set the maintenance program to the previous routing so that program operation will switch to the first error printout.
6. If the trouble is intermittent, you can use the marginal-check version of the program to apply prescribed margins to the memory unit. If the original memory failure is detected under marginal-check conditions, verify the failure margin by applying a manually controlled voltage excursion to the tested marginal unit while running the routine.
7. Check the memory unit for the remaining errors that were detected by the maintenance program. This is accomplished in the same manner as in steps 5 and 6.
8. Analyze the error printouts to determine whether the error is a bit failure or a selection failure. If the error is a bit failure, determine which bit is causing the failure, and check the associated sense amplifier or digit plane driver. If the error is a selection failure, further analyze the printout to determine which selection line or lines are causing the failure, and check the associated driver stage.
9. Refer to the equipment technical manual troubleshooting guides to determine which circuits should be tested using an oscilloscope. Calibrate an oscilloscope to make the necessary voltage and timing measurements. Then rerun the maintenance routine and observe the action of the suspected memory circuits.

MAGNETIC DRUM UNITS

The general procedure for troubleshooting the magnetic drum units of a computer is to verify the existence of a failure and determine its source and cause. Maintenance programs are the chief means of determining the particular areas in which malfunctions have occurred. However, in equipment where maintenance programs are not effective or not available, manual testing must be performed at the computer maintenance console or directly at the magnetic drum units. Marginal-checking procedures are applicable for troubleshooting intermittent malfunctions.

Maintenance Programs

If the computer has this feature, you can run a diagnostic program whenever a drum unit is not functioning correctly. A drum unit maintenance program can be used to check the fields, status controls, disconnect counter, field switch, and information circuitry. To make a test, you place the drum unit in the computer-test mode and initiate the maintenance program which is stored on cards or magnetic tape. The program first clears the drum fields, then writes a test pattern. The computer reads and checks information transfer. Errors are indicated if the program is completed or if incorrect information is printed out. Maintenance tables are normally furnished with the equipment technical manuals as aids to determine the source of trouble from the results of a maintenance check.

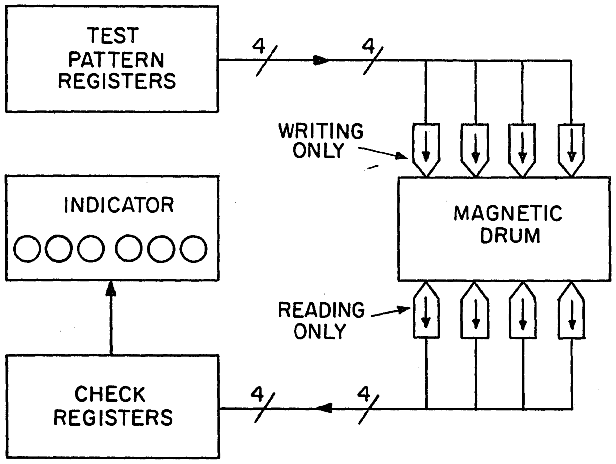
Write-Read Check

The bits of information that are recorded on a magnetic drum by a test pattern may be displayed by lamps on a check register in the drum unit. These indicators are used in conjunction with tables in the equipment technical manual to help you analyze the cause and location of a malfunction. Such tables are arranged according to the types of bit errors which may be encountered; i.e., missing bits common to two or more fields on one drum, a missing bit in one field, or two adjacent bits in error in one field. When one of these conditions is present, the corresponding table should be used to locate the trouble within a defective circuit or device. An arrangement of computer circuits for testing a magnetic drum is shown in figure 17-14.

Several test patterns should be used individually or in combination to eliminate the possibility that an error may go undetected or be misinterpreted. Examples of such test patterns are as follows:

1. Straight "0" bit pattern.
2. Straight "1" bit pattern.
3. "0" and "1" bits alternating in the first register and complementing in the following register; i.e., first register, 010101, second register, 101010, etc.
4. All "1" bits in the first register, all "0" bits in the second register, etc.

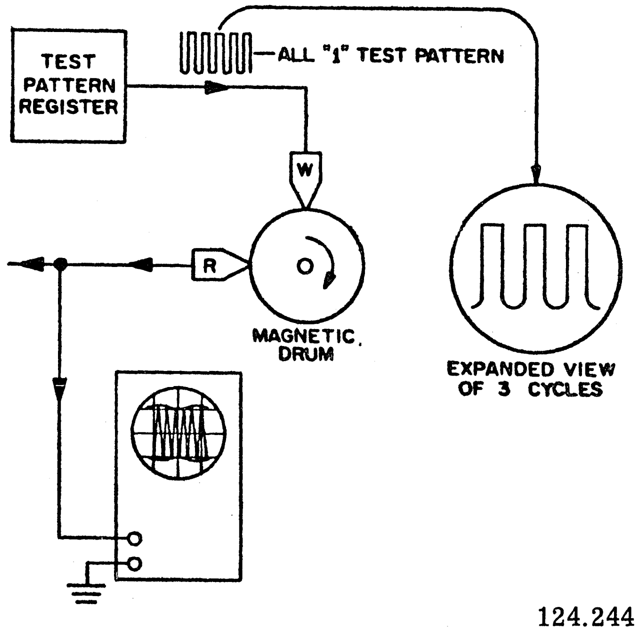
A test pattern applies a "1" or "0" bit to each write head for the purpose of receiving identical information from the corresponding read head. The use of one test pattern may not prove sufficient, in all cases, to thoroughly identify a trouble. You may then require one pattern or a combination of two or more patterns to locate the trouble which causes bit errors. For example, a straight "0" bit pattern, which is applied to the drum write heads, may appear as all "0" bits on the check register, and yet one of the read heads may be defective and produce only "0" bits. The use of a straight "1" bit pattern, in this case, will reveal the particular read head which is at fault. You must then make further checks to determine whether the failure is caused by a defective head, a misadjusted head, or the read amplifier stages that are connected to the head. If the read head and corresponding amplifier stages are operating correctly, you should test the corresponding write head and its associated circuitry.



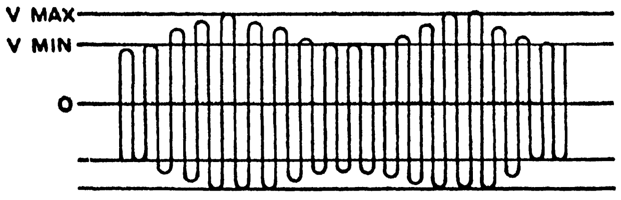
124.243
Figure 17-14.—Magnetic drum write-read test, block diagram.

Runout Test

Excessive bearing wear, unbalance, and ovalness of a drum rotor can be checked by means of a runout test. To perform this test, connect an oscilloscope to a read head as shown in figure 17-15. Operate the drum motor at its normal speed, and apply an all "1" bit test pattern to the write head. The read head waveform should resemble the waveform shown in figure 17-16. For the drum to be considered



124.244
Figure 17-15.—Magnetic drum runout test, block diagram.

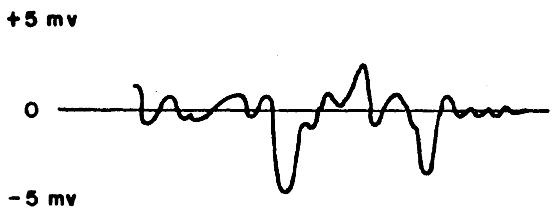


124.245
Figure 17-16.—Runout test waveform.

satisfactory, any change in the amplitude of the test pattern bits, Vmax minus Vmin, must be less than 20 percent of the maximum bit amplitude. This measurement should be made from read heads located at each end of the drum, and also from a read head at the approximate center of the drum.

Noise Test

You can determine whether noise is being generated by a magnetic drum by using an oscilloscope in the same manner as in the runout test. For this test, disconnect the test-pattern generator from the write head, and erase all signals from the drum. Then start the drum motor and observe the display on the oscilloscope. The oscilloscope display may resemble the waveform shown in figure 17-17. Any voltage that is present is a result of residual electrical noise or spurious magnetization at the drum rotor. This test must be performed for each recording track on the drum rotor. You will have to refer to the equipment technical manual to determine the noise spike amplitude that can be tolerated and the number of tracks that can be noisy before the drum rotor must be replaced. For the example shown in figure 17-17, 5 millivolts is considered to be the maximum noise pulse amplitude that can be permitted.



124.246
Figure 17-17.—Noise test waveform.

Crosstalk Test

A crosstalk test determines whether each channel processes data free of interference from other channels. To perform this test, erase all information from the track to be checked, and connect the read-head output to the oscilloscope. Then operate the drum and apply test patterns to all channels except the channel being tested. Use a calibrated oscilloscope to measure the output noise level due to crosstalk. This test must also be performed for each track on the drum. The noise level and number of noisy channels that can be tolerated will be listed in the equipment technical manual.

MAGNETIC TAPE UNITS

The troubleshooting of computer magnetic tape units can also be accomplished by means of diagnostic maintenance programs. Such a program may use the control units, memory units, line printer, and marginal check equipment to test the tape units. Such a program may be composed of three test routines, as follows:

1. Routine 1 uses four different test patterns to check the timing of the write, read, backspace, and rewind instructions, as well as their operation. This routine also checks the write and read end-of-file circuits, the read zero word instruction, the disconnect circuits for the conditions either read more or read less than the number of words contained in a record, and all information transfer paths involved.
2. Routine 2 uses a random number pattern to check the timing of the read, write, and rewind instructions and their operation. This test is accomplished by checking the rewind operation, the stepping of the tape clock, the tape speed, and the effects of a number of short random-number records.
3. Routine 3 is used to check the file-protect-on circuitry.

The routines to be run will depend on the error indication. These three maintenance program routines can be run in five different modes, as follows:

1. Mode 1 uses routine 1 only.
2. Mode 2 used routine 2 only.
3. Mode 3 automatically applies excursions for prescribed margins or margins to failure, using either routine 1 or routine 2 depending on which marginal-check word is being used.
4. Mode 4 selects the tape drives available and uses routines 1 and 2 on each tape drive

automatically. This mode is normally used for preventive maintenance and not for trouble shooting.

5. Mode 5 uses routine 3 only.

When these routines are run, all errors will be indicated by a printout, which will include the type of error, the trouble location, and possible plug-in units that caused the failure. In modes 1, 2, and 5, errors will also be indicated by a programmed halt, with the computer accumulator designating when the error occurred. In mode 3, the marginal check words will also be printed, designating the selected line and excursion that caused the failure. In mode 4, the program will print out errors, but will halt only after all tape drives have been tested.

REPAIR PROCEDURES

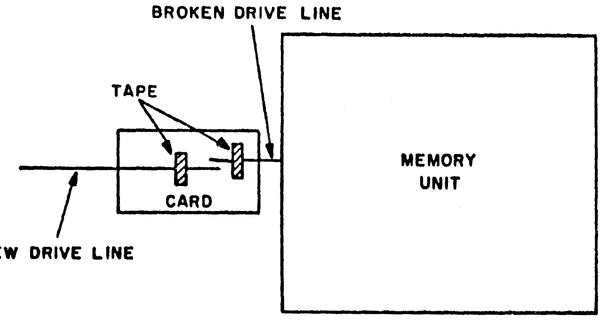
When the cause of a computer failure has been found by troubleshooting procedures, you will normally perform corrective action, such as adjustment, alignment, tuning, cleaning, and replacement of parts. The replacement of computer parts involves the same basic procedures as are used for radio, radar, and carrier equipment. When replacing parts with many wire connections, such as switches, relays, connectors, and transformers, each wire should be tagged and identified. You can then remove the wires from the defective part and make the replacement without making an error in the wiring. Many computer circuits are contained in modules, which may be either plug-in units or soldered to the equipment. The extent of repair for module units will depend on what is considered to be economically practicable. The units may be repaired in field maintenance shops or sent to depots. Appropriate soldering techniques must be used when repairing computers that use semiconductors, miniature circuit parts, and printed circuits to avoid overheating the parts. The voltages used in ohmmeter measurements of semiconductor circuits must be limited to safe values.

MEMORY UNIT WIRE REPLACEMENT

On large magnetic core memory units, where many planes are used, replacing a broken wire or an entire plane must be done with extreme care to avoid creating new troubles. If one of the X or Y drive lines within a memory plane breaks, the plane will normally have to be removed from the memory unit to be repaired.

If an X or Y drive line breaks close to the terminal on a memory plane, you may be able to replace the wire without having to remove the plane. A recommended procedure is as follows:

1. Disconnect the power to the memory unit, and remove necessary panels from the cabinet or rack.
2. Double-check the suspected wire by making a continuity check with an ohmmeter.
3. Cut a new wire of the correct diameter and length, and tin the ends.
4. Remove the jumper wires, on both sides of the plane, that are connected to the same terminals as the broken drive line. Then remove the portion of the broken end of the drive wire that is wound around the terminal.
5. Have another person hold the broken end of the drive line so that it cannot be pulled into the plane. Carefully unsolder and unwind the wire on the terminal at the unbroken end of the drive line.
6. As well as possible, straighten out the portion of the wire that has been unwound from the terminal at the unbroken end of the drive line. Place the wire end-to-end with the new wire. Overlap the ends approximately 1/16 inch, and tape them to a support such as a computer cord, as shown in figure 17-18. Solder the wires together to make a temporary mechanical bond, and remove the tape.
7. Smooth the solder joint with a sharpening stone until there are no ends or sharp bulges that will catch as the wire is pulled through the cores.
8. Gently pull the broken end of the wire out of the plane. If the solder joint binds within the plane, try gently jockeying the wire back and



124.247
Figure 17-18.—Method of connecting replacement wire to broken drive line.

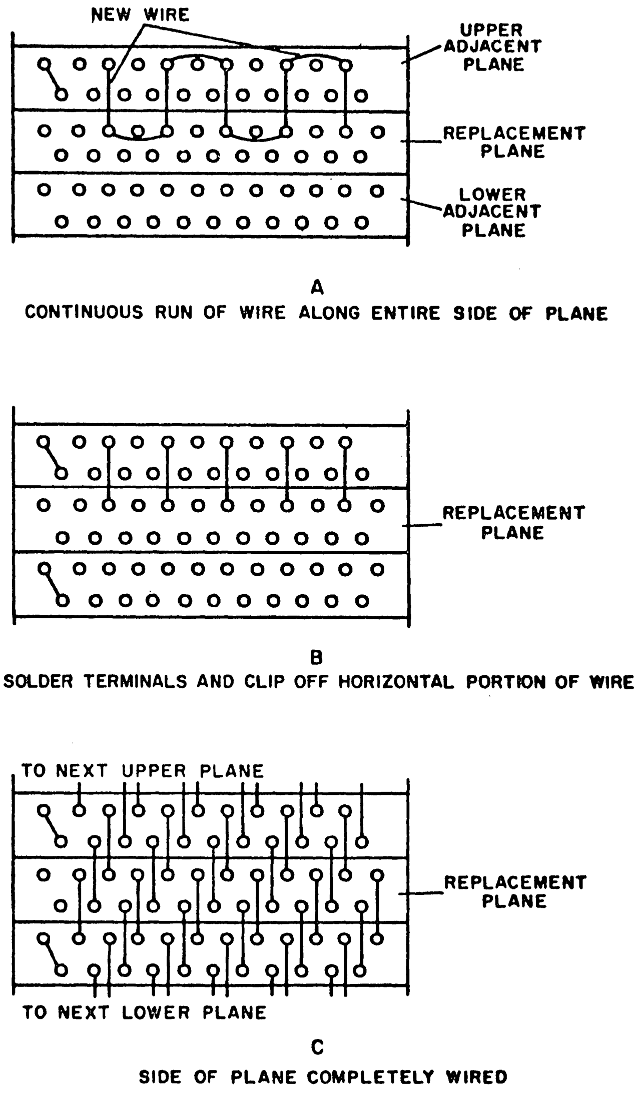
forth. Excessive pull on the wire can cause it to break in the plane, damage a core, or damage another winding; in such cases, the entire plane must be removed. If jockeying does not help, pull the wire back and smooth the solder joint for another try.

9. After a new wire is pulled into place, clip off the old wire at the solder joint. Solder the new wire and the jumper wires, that were previously removed, to the proper terminals.

MEMORY PLANE REPLACEMENT

If a memory plane is found to have a defective core or a wire having a break that cannot be reached from the outside, it must be removed and either repaired or replaced. Since this is a time-consuming job which, if not done carefully, can create other troubles, you should make every possible check to prove that the suspected plane is really defective before removing it. Make continuity checks of suspected wires, and make certain that the trouble is not caused by a poor solder connection at one of the terminals. When you are certain that a plane must be removed, proceed as follows:

1. Disconnect the power to the memory unit and remove the necessary panels from the cabinet or rack.
2. Shield the assemblies below the planes to be worked on by attaching a cheesecloth with masking tape.
3. Use a grease pencil to mark the defective plane on all four sides so that you will not make a mistake and remove any wrong wires.
4. Cut all jumper wires to adjacent planes by cutting close to the defective plane terminals.
5. Remove the cut jumper wires from adjacent planes by grasping the wire with needle-nose pliers and unsoldering at the terminal. Wipe the terminals clean of surplus solder.
6. Remove any retaining screws and air seal material from the defective plane.
7. Make a note of the proper position for the defective plane so that you will be able to insert a replacement plane in the same position. Then slide the defective plane out of the memory unit.
8. Insert a replacement plane, replace the retaining screws, and replace the air seal material that was removed during step 6.
9. Replace all jumper wires that were removed in steps 4 and 5. Figure 17-19 shows an efficient method of doing a good wiring job. Choose two horizontal rows of terminals that must be connected together with jumper wires.



124.248
Figure 17-19.—Replacement of plane jumper wires.

Connect these terminals with a single wire, as shown in part A of the figure. Keep the wire taut, and wrap it around each terminal. Then solder the wire to the terminals to be used, and cut out the horizontal portion of the wire. The plane wiring will then be similar to that shown in part B of figure. This procedure should then be repeated for the other sets of horizontal terminals that must be connected by jumper wires. Part C of the figure shows the jumper wires reconnected to one side of the replacement plane.

10. When all four sides of the plane are completely rewired, remove the cloth shield and check for wire scraps and solder splashes that may have fallen past the cloth. Then replace the cabinet panels and check the operation of the memory unit.

TECHNICAL PUBLICATIONS

Electronic technical publications include various handbooks, bulletins, and manuals published and distributed by the Bureau of Ships, and manufacturers' technical manuals. The Requisitioning Guide and Index of Forms and Publications, NavSandA-2002 furnishes a complete list of BuShips technical publications along with instructions for ordering copies.

HANDBOOKS AND BULLETINS

Handbooks present information about a particular field of work, or about a particular type of equipment, in a practical form. One such publication is the Handbook of Test Methods and Practices, NavShips 91828A (or the latest revision); others are the Handbook of Naval Shore Station Electronics Criteria, NavShips 92675, and the Electronic Test Equipment Handbook, NavShips 900-155.

One bulletin of great importance is the Electronics Information Bulletin (EIB), NavShips 900-022A, published biweekly for naval electronics activities. A complete file of these bulletins should be maintained.

This bulletin lists field changes and corrections that must be made in instruction books and other publications that are used in the maintenance of electronic equipment. It also lists electronics publications that become available, and gives valuable suggestions, from case histories, for servicing electronic equipment.

Approximately once a year, the Bureau of Ships determines which EIBs become obsolete because their items are published elsewhere in permanent form (in EIMB and IB publications). For example, the initial 200 copies of EIB, were named Repair Information Bulletin (RIB) and these are canceled (as summarized in EIB 476) along with EIBs from serial numbers 1 through 380. Until further notice, serial numbers above 380 shall be considered authoritative and directive in nature for announcing field changes that are active.

Although not called a bulletin as such, the Electronics Installation and Maintenance Book (EIMB), NavShips 900-000, includes essentially the summarized (permanent) information formerly contained in various service bulletins, such as the RMB, CEMB, EIB, and SB. For convenience, the EIMB comprises several volumes, covering general information of an electronics nature in each major electronics field. The present name of EIMB was changed from EMB.

At present the EIMB is being completely revised. This revision will be carried out in several steps, and includes incorporating into the EIMB the following NavShips publications; Reporting Electronic Equipment Installation, 900-135(B); Handbook of Test Methods and Practices Manual, 91828(A); Installation Practices Manual, 900-171; Antenna Details, 900-121(A); Basic Communication Systems Interconnection Wiring Plans, 900-176, Volume 1; Shipboard Radar and IFF System Design Plans, 900-176, Volume 2; and Shipboard Electronics Equipment Installation Plans, 900-153(A).

The Handbook of Electronic Circuits, NavShips 900-000.102, currently under preparation will include most conventional electronic circuits (oscillators, detectors, amplifiers, etc.) and their theory of operation. The new equipment technical manuals will not duplicate these explanations but will simply refer to them. In this way it is planned to reduce the amount of material in the theory portions of technical manuals associated with electronic equipments.

INSTALLATION AND MAINTENANCE MANUALS

Installation and maintenance manuals contain information concerning the installation, operation and maintenance of specific electronic equipments. This information is included in instruction books or technical manuals issued by the manufacturers, and various BuShips publications.

MANUFACTURERS TECHNICAL MANUALS

Manufacturers technical manuals are prepared according to military specification MIL-M-15071 E, of 15 April 1962. This specification establishes four types of manuals and provides specific instructions for preparing each type. The types of manuals are: type 1, electrical and mechanical; type 2, electronic and special equipment; type 2a, experimental equipment; and type 3, systems. You will be concerned with the type 2 manual. The material in this manual is arranged in six sections as follows:

Section 1. General information—This section provides a functional description of the equipment and includes the capabilities, limitations, and relationship of the units.

Section 2. Installation—This section includes such information as primary power data, initial adjustment, inspection procedures, unpacking and handling, and installation requirements.

Section 3. Operation—This section includes routine and emergency operating instructions, safety precautions, and operating limits.

Section 4. Troubleshooting—Included in this section are all diagrams and information required to troubleshoot the equipment.

Section 5. Maintenance—This section provides information and instructions necessary for maintaining and repairing the equipment. All preventive maintenance procedures and test inspections are included in this section if they are not included in a separate maintenance standards book.

Section 6. Parts list—This section includes a list of manufacturers, and data concerning maintenance parts for the equipment.

In addition to the material in sections 1 through 6, the manual contains front matter and an index.

SHIP'S INFORMATION BOOK

The Ship's Information Book, (a replacement for the currently used General Information Book) provides a source of technical information relative to shipboard arrangements and systems. The book is prepared by the shipbuilder of each combatant ship and for many of the larger auxiliary ships. A Ship's Information Book comprises six volumes which are entitled:

Volume 1—Hull and Mechanical
Volume 2—Piping, Ventilation, Heating, and Air-Conditioning Systems
Volume 3—Power and Lighting Systems
Volume 4—Electronic Systems
Volume 5—Interior Communication and Fire Control Systems
Volume 6—Instrumentation

The technician will find the following three volumes extremely useful. Volume 3-Power and Lighting Systems will locate fuse panels and circuit breakers in electrical power lines feeding the various compartments and electronic equipment. Volume 4-Electronic Systems will describe all electronic equipment and explain external connection between electronic systems. Volume 5—Interior Communication and Fire Control Systems will describe types of communication available from various electronic spaces; also interconnections from fire control systems and electronic systems.

BUREAU OF SHIPS TECHNICAL MANUAL, (NAVSHIPS 250-000)

The Bureau of Ships Technical Manual is an authoritative technical publication issued for the information and guidance of naval personnel, afloat and ashore, responsible for or engaged in the operation, maintenance, and repair of machinery, apparatus, and equipment under cognizance of the Bureau of Ships.

Chapter 9670 of the manual is titled Electronics, and is required reading for electronics personnel. Additional chapters of the technical manual and various other publications are listed in chapter 9670. These are informative and of value to electronics personnel.

THE BUREAU OF SHIPS ALLOWANCE LIST

The Bureau of Ships Allowance List is furnished to inform the ship of the machinery, equipment, equipage, special tools, repair parts, and other material authorized and required to be on board, as specified by the Bureau of Ships, for operation of the ship. Ship's copies of the allowance list must be kept complete and up-to-date at all times. Instructions for accomplishing this requirement are contained in the Introduction section of each ship's allowance list.

BUREAU OF SHIPS JOURNAL NAVSHIPS 250-200

The Bureau of Ships Journal is a monthly unclassified periodical containing current and informative technical articles on the work and problems of the Bureau of Ships and its field activities. Subject matter contained in this publication is not to be construed as official directives; but rather, the carefully considered opinion of personnel in BuShips and activities under its management control.

Articles in the Bureau of Ships Journal are published as information, and for possible adoption at the discretion of personnel in the field.

NTDS LIBRARY NAVSHIPS-94976

The NTDS library consists of five volumes now in print and two volumes in process of being printed.

Vol. 1	Introduction to NTDS
Vol. 2	NTDS Equipment Configurations
Vol. 3	NTDS Equipment Handbook
Vol. 4	NTDS Display Handbook
Vol. 5	NTDS Index and Bibleography
Vol. 6	Data Processing
Vol. 7	Communications as to NTDS

The titles of volumes six and seven are not firm so may differ slightly when published. Volumes one through four are CONFIDENTIAL and will be in custody of the classified publication officer. Volumes five through seven are UNCLASSIFIED. The information available in this library will give the technician an overall picture of the system and break it's use down to input, processing and output. A film version of volume one through four will be available at a later date.

CORRECTIONS TO TECHNICAL PUBLICATIONS

It is important that all technical publications be kept up to date. You will be concerned with making corrections to the electronics technical publications. The corrections are distributed by BuShips as change sheets, as required.

These change sheets are detailed regarding the kind of entries (temporary or permanent) and their purposes. Many changes involve field changes covering specific serial numbers; when that condition applies, the entry is made only upon completion of the job. Be certain that holders of equipment accompanied by technical manuals do not make this correction in the manual until accomplishment of the field change.

Make pen-and-ink corrections in all entries and in all locations as specified. Make them clear and legible. Remember, such entries are permanent; therefore they must be readable.

Following the initial entry, review the change sheet instructions to be certain every requirement is fulfilled.

CHAPTER 18

MAINTENANCE PROCEDURES

JOINT ELECTRONICS TYPE DESIGNATION SYSTEM (AN SYSTEM)

The Electronic Type Designator System (AN System) for electronic equipment was adopted in 1943. The type designation includes the assigning of a short combination of letters and numbers to an equipment in a specific sequence wherein the equipment can be easily identified. The system is designed to:

1. Be logical in principle so that the nomenclature type numbers and/or letters will be readily understood, and the operation of the armed forces supply services will be facilitated.
2. Be flexible and sufficiently broad in scope to cover present types of equipment and new types and uses of equipment that will be developed in the future.
3. Provide adequate identification on name plates with or without the name part of the nomenclature.
4. Provide a ready means of identifying equipment in correspondence and other types of communication.

A discussion of all of the letters and numbers which are used in assigning equipment designators is beyond the scope of this text. However, a knowledge of the complete system is a DS requirement. Complete information on the AN nomenclature system can be found in the Navy Stock List.

The AN/ denotes a SET of electronic components. Thus, the Unit Computer AN/USQ-20 (discussed earlier in this course) is a combination of equipments which together represent a Computer SET. The CP-642A/USQ-20 is a component of the AN/USQ-20. The following tables are shown to illustrate the meaning of the letters and numbers used to identify these equipments.

Components are identified by means of indicating letters (table 18-2) which tell, in this case, the type of component it is; i.e., the CP which indicates "computer." The number 642 which follows, identifies the particular component. The letter "A" after 642 denotes the first modification. Finally, the USQ-20, which follows the slant, specifies the set for which the component is a part or with which it is used.

Table 18-1.—Designation For Unit Computer AN/USQ-20.

Unit Computer	AN/	U	S	Q	20
Item name as prescribed	Assigned in Joint Electronic Type Designation System (Formerly Joint Army-Navy System)	General Utility	Special Types	Special or combination of purposes	Model Number

Table 18-2.—Designation For CP-642A/USQ-20.

CP	642	A	USQ 20
Computer—a mechanical and/or electronic mathematic calculating device	Identifying number	First modification	Set identification for which the component is used.

CABLES

The various electrical and electronic systems aboard ship depend on power supplied by the ship's service generators. This power is distributed to the equipment by a system of cables. Power cables are normally installed by shipyard forces, but the DS must know the location and characteristics of the cabling, which supplies his data processing equipment.

He should become familiar with the current-carrying capacity of cables; their insulation strength; and their ability to withstand heat, cold, dryness, bending, crushing, vibration, twisting, and shock. Several types of cables are used in the applications under discussion, with design characteristics suited to their location and purpose.

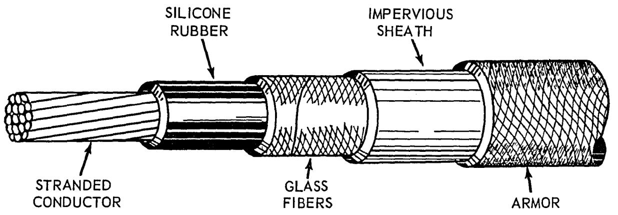
Type SGA (Shipboard, General use, Armored) cables are designed to have a minimum diameter and weight consistent with service requirements in fixed wireways on Naval ships.

Type SSGA cable (fig. 18-1) consists of stranded copper conductors (in this case, only a single conductor-indicated by the "S" before SGA) insulated with silicone rubber and glass fibers around which is placed an impervious sheath. The sheath is covered with braided metal armor, and then a coat of paint is applied.

The SGA cables are designated as follows: (1) SSGA, single conductor; (2) DSGA, twin (double) conductor; (3) TSGA, three conductor; and (4) FSGA, four conductor.

Many applications aboard ship require cables than can be bent without damaging the conductor insulation or the protective covering. For such applications, flexible cables are used.

Flexible cables have synthetic rubber or synthetic resin insulation and a flexible sheath that is resistant to water, oil, heat, and flame. However, these cables are not as heat and flame resistant as armored (SGA) cables. Flexible cables for general use are designated by the letters, HOF-for example, DHOF, THOF and FHOF. Flexible cables for limited use are designated by the letters COP-for example, DCOP, TCOP, and FCOP.



1.33

Figure 18-1.—SSGA cable.

Other types of cables used in electronics work are:

1. DRHLA-Double conductor, radio, high-tension, lead armored.
2. FHFTA-Four conductor, heat and flame resistant, thin-walled armor.
3. MCSP-Multiple conductor, shielded, pressure resistant (submarine applications).
4. TTRSA-Twisted-pair telephone, radio shielded, armored (characteristic impedance approximately 76 ohms).

DESIGNATION OF CONDUCTOR SIZE

Generally, when the size of the individual conductors contained in the cable is indicated in the cable designation, the numeral (or numerals) following the letter designation indicate the approximate cross-sectional area of the individual conductors in thousands of circular mils to the nearest thousands. For example, TSGA-60 is a 3-conductor armored cable for general shipboard use, with each conductor having a cross-sectional area of 60,090 circular mils. However, when the numerals immediately following the letter designation indicate the number of conductors comprising the cable, the size of the individual conductors may be indicated by additional numerals enclosed in parenthesis. For example, MDGA-19 (7) is a 19-conductor electrical power cable for shipboard nonflexing service, with each conductor having a cross-sectional area of 6,512 circular mils.

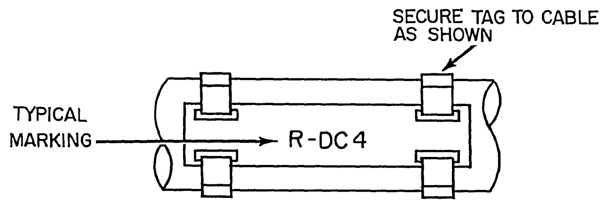
MULTIPLE-CONDUCTOR DESIGNATIONS

Multiple-conductor cable types and class designations are followed by a number that indicates the number of conductors. For example, MSCA-30 is a multiple conductor shipboard control armored cable containing 30 conductors.

For telephone cable, the number indicates twisted pairs. For example, TTHFWA-25 means that the cable contains 25 twisted pairs; TTRSA-4 means that the cable contains 4 pairs individually shielded.

TAGGING CABLES

Ship's cables are identified by metal tags (fig. 18-2) that give information about the cable. Permanently installed ships' cables are tagged



12.74
Figure 18-2.—Cable tag.

as close as practicable to each point of connection, on both sides of decks, bulkheads, and other barriers. Cables located within a single compartment in such a manner that they can be readily traced are not tagged.

Power and lighting cables are tagged as specified in article 9600-270 of the Bureau of Ships Technical Manual. Cables between units of electronic equipment (indicated by the R as the first letter in the nomenclature) are tagged with electronics designations (table 18-3). For example, in figure 18-2, the R indicates electronics, DC indicates Radar and sonar data converters, and 4 (not shown in the table) indicates cable number 4 of the radar and sonar data converters. Electronics Information Bulletin #578 contains additional information concerning electronics designations.

LOCATION AND LENGTH OF CABLE RUNS

When possible, cables are run along different well separated paths to reduce the probability of battle damage to several cables simultaneously. Wherever possible, high-temperature locations are avoided. Pulse cables are run separately, when possible, to reduce coupling and interference.

Because attenuation (power loss) in a line increases with its length, cables are kept as short as practicable, consistent with avoiding high-temperature locations, sharp bends, and strain on the cable.

If the equipment is shock mounted, enough slack in the cable is allowed to permit unrestricted motion of the equipment. The cable may be wrapped with friction tape for a distance of three or four inches from a point under the last cable clamp in the direction of the equipment. This eases the bending of the cable at the point and reduces the possibility of cable deformation because of constant vibration.

Table 18-3.—Electronics Circuit or System Designations.

Circuit or system designation	Circuit or system title
R-A	Meteorological equipments
R-B	Beacons
R-C	Countermeasures
R-D	Data
R-DA	Data-Analog
R-DAD	Analog to digital converters
R-DC	Radar and Sonar data converters
R-DD	Data-digital
R-DDA	Digital to Analog converters
R-E	Radar
R-F	Fire control radar
R-G	Telemetering Systems or Remote Control Electronic Guidance
R-H	CW passive tracking equipments
R-I	IFF equipments
R-K	Timing functions
R-L	Automatic vectoring
R-M	Missile support circuits
R-N	Infrared equipment
R-P	Special purpose circuits
R-R	Radio communication
R-S	Sonal
R-T	Television

When cables are connected to equipment that slides out for maintenance, extra slack is provided.

TROUBLESHOOTING TECHNIQUES FOR MINIATURE COMPONENTS AND DEVICES

The use of transistors, crystal diodes, and other semiconductors in Naval electronic equipments is constantly increasing. Because of its versatility, the transistor is used in amplification, modulation and demodulation, and other electronic circuitry applications. Its miniature dimensions make the transistor particularly suitable for use in unitized and modular constructed equipment. For the same reason—miniaturization and compactness—troubleshooting in equipment containing transistors is made more difficult.

The successful installation, repair and maintenance of electronic equipment, especially equipment using transistors, has raised many questions concerning proper servicing procedures and troubleshooting practices that previously have been used in electron-tube circuitry.

Like electron tubes, transistors come in various shapes and sizes and often are classed in special categories according to their use and application. The characteristics of each of these devices are usually presented in SPECIFICATION SHEETS or they may be included in tube or transistor manuals.

It should be noted that the primary difference between the operation of a transistor and an electron tube is that the electron tube is a voltage-operated device and the transistor is a current-operated device.

The comparison of a given transistor and electron tube shows that there is great similarity between the functions of a transistor and those of an electron tube. Therefore, any knowledge picked up by the technician in his work on electron tube equipment will be useful in the servicing of transistorized circuits. However, there are great differences between a transistor and an electron tube from the stand point of servicing. For instance, the reliance placed on the senses of sight, touch, and smell in the visual inspection of electron tube circuits is not feasible in the case of transistor circuits. Many transistors develop so very little heat that nothing can be learned by feeling them. High-frequency transistors hardly get warm. Usually, if a transistor is hot enough to be noticeable, it has been damaged beyond use (except special or high-power transistors).

In the case of an electron tube, which is usually of the plug-in type, a quick test is sometimes made by the PART SUBSTITUTION method; that is, by replacing the tube suspected of being faulty with one known to be good. In transistorized circuits, the transistors are frequently soldered and part substitution becomes impractical. Furthermore, indiscriminate substitution of semiconductors should be avoided; it is preferable to test transistors IN-CIRCUIT using the Transistor Test Set TS-11001U. The first step in troubleshooting transistor circuitry, as in the troubleshooting of electron tube circuitry, is a visual inspection of the entire equipment. Loose connections, broken leads, and any other visible damage should be repaired before undertaking the next step of the troubleshooting procedure. A careful visual inspection will frequently shorten what could otherwise be a lengthy service job.

When the visible defects have been corrected, experience has shown that it is more efficient first to determine the defective stage by means of a signal-substitution or signal-tracing method and then to analyze carefully that stage for defective components. To apply the troubleshooting method recommended, a voltmeter, ohmmeter, and a signal generator are required.

The general rule to be followed in servicing electron tube or transistor equipment is: first, use the signal generator to locate the defective stage or interstage components, then apply the voltmeter and ohmmeter to determine the defective part or parts.

Most good-quality test equipment used for electron tube circuit troubleshooting may also be used for transistor circuit troubleshooting. However, before employing any test equipment, make sure that it meets the requirements given in the following paragraphs.

Signal generators, both R-F and AUDIO, may be used if the power supply in these equipments is isolated from the power line by a transformer. Before any tests are made with a signal generator, a common ground wire should be connected from the chassis of the equipment to be tested to the chassis of the signal generator before any other connections are made.

Signal tracers may be used with transistor circuits if the precautions concerning the power supplies in signal generators are observed. Many signal tracers use transformerless power supplies; therefore, to prevent damage to the transistor, an isolation transformer must be used.

Multimeters that are used for voltage measurements in electron tube equipment or transistor circuits should have a high ohms-per-volt sensitivity to provide an accurate reading. A 20,000 ohm-per-volt meter or an electronic voltmeter with an input resistance of 11 megohms or higher on all voltage ranges is preferred on transistor circuits.

Ohmmeter circuits which pass a current of more than 1 ma through the circuit under test cannot be used safely in testing transistor circuits. Therefore, before using any ohmmeter on a transistor circuit, check the current it passes on all ranges. DO NOT use any range that passes more than 1 ma.

Conventional test prods, when used in the closely confined areas of a transistor circuit, often are the cause of accidental shorts between adjacent terminals. In electron tube circuits the momentary short caused by test prods rarely results in damage but in transistor circuits this momentary short can ruin a transistor. Also, since transistors are very sensitive to improper bias voltages, the practice of troubleshooting by shorting various points to ground and listening for a CLICK must be avoided. In electron tube circuits, momentary shorts may occasionally cause a component to burn out, although they rarely affect tubes. In a transistor circuit, the transistor is usually the weakest link, and it becomes the victim.

The sensitivity of a transistor to surge currents should always be borne in mind whenever making any voltage measurements in transistor circuits.

Another change from conventional troubleshooting procedure that is required by transistors is the use of a small, low-wattage soldering iron possessing a narrow point or wedge. Wattage ratings on the order of 35 to 40 watts are satisfactory. The common type of soldering gun or iron used on electron tube circuits should never be used on transistor circuits.

Always remember that because these units are small and have many features and characteristics which differ from those of the more familiar electron tubes, the servicing of transistor equipment requires a modification of presently used and familiar techniques. REMEMBER, TRANSISTORS ARE CURRENT-OPERATED DEVICES: ELECTRON TUBES ARE VOLTAGE-OPERATED DEVICES.

TRANSISTOR SPECIFICATIONS

Semiconductors, like electron tubes, are available in a large variety of types, each with its own unique characteristics. The characteristics of each of these devices are usually presented in SPEC SHEETS, or included in tube or transistor manuals. The specifications usually cover the following items:

1. The lead paragraph of a semiconductor specification sheet is a general description of the device, and contains three specific pieces of information:

   a. The kind of semiconductor. This covers the semiconductive material used, such as germanium or silicon; whether type PNP or NPN, etc., and the type of construction, whether alloy-junction, grown or diffused junction, etc.

   b. Some of the major applications are listed, such as audio amplification, oscillator, and high-gain R-F amplification.

   c. General sales features, such as size and packaging.

2. The ABSOLUTE MAXIMUM RATINGS of voltages and collector current. These ratings should not be exceeded under any circumstances, as semiconductor failure may result.
3. Collector power dissipation. The power dissipation of a transistor is a function of its junction temperature and the ambient temperature. The higher the temperature of the air surrounding the transistor, the less power the device can dissipate. A factor telling how much the transistor must be derated for each degree of increase in the ambient temperature is usually given.
4. Current transfer ratio. This is another name for alpha or beta of the transistor.
5. Collector cutoff current. This is leakage current from collector to base when no emitter current is being applied.

Additional information is provided for engineering design purposes.

DIODE AND TRANSISTOR (DESIGNATION SYSTEM)

A standardized system of numbers and letters is used for designating diodes and transistors:

1. The first number indicates the number of junctions. Thus 1 designates a diode; 2 designates a transistor (which may be considered as made up of two diodes, the base-emitter and base-collector diodes); 3 designates a tetrode, a four-element transistor.

2. The letter N following the first number indicates a semiconductor.

3. The 2 or 3 digit number following the letter N has no particular significance, except that it indicates the order or registration. When this number is followed by a letter, it indicates a later, improved version.

Thus, a semiconductor designated as type 2N345A signifies that it is a three-element transistor of semiconductor material and that it is an improved version of type 345.

TRANSISTOR LEAD IDENTIFICATION

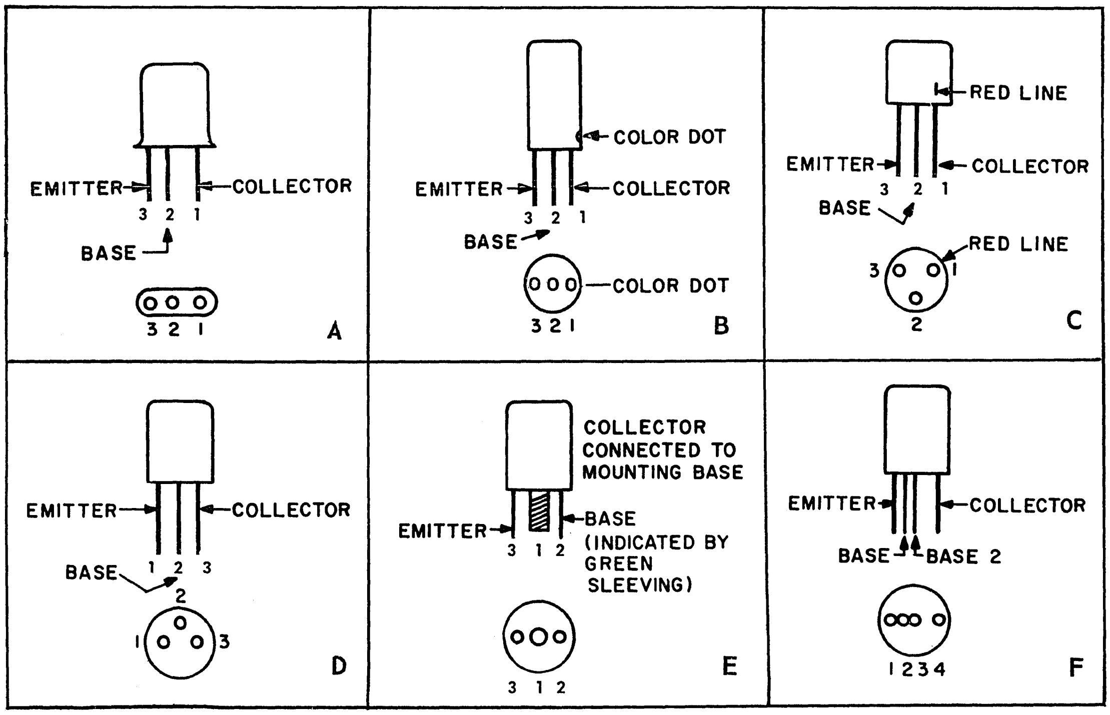
The arrangement and coding of transistor leads is shown in figure 18-3. Part A shows a transistor in an oval case. The collector lead is identified by a wide space between it and the base lead, which, in turn, is followed by the emitter lead. Part B shows a round case with three leads in line and equally spaced. The collector lead is marked on the case by means of a color dot, usually red. The other two leads are the base and emitter, in that order. In part C, the collector lead is marked by a red line on the case. The base and emitter leads follow clockwise around the circle, in that order. In part D the leads are located at three points of a quadrant. When viewed from the bottom in a clockwise direction, the first lead following the blank space is the emitter, followed by the base and collector. Part E shows a conventional power transistor where the collector is connected to the mounting base, the mounting bolt forming the conductor for the collector. The base lead is identified by its green sleeving.

It should be noted that sometimes, even where all three leads are present, one of the elements may be connected internally to the mounting base to provide additional cooling.

Part F shows a tetrode. The collector is identified by the wide space between it and the other leads, which are: base 2, base, and emitter, in that order.

TRANSISTOR HEAT DISSIPATORS

As the complexity of transistorized electronic equipment has increased, the space available for individual components has almost



1.288

Figure 18-3.—Transistor lead identification.

disappeared. This trend in design has meant more heat generated with less space in which it can be dissipated, causing an ever-increasing environmental temperature in which the transistor must operate. The reliability of a transistor, like that of an electron tube, is dependent on its ability to safely absorb and dissipate the internally generated heat while operating at the increased temperature resulting from its own heat. For this reason, transistor heat dissipating devices are finding widespread application to prevent the effects of higher operating temperature and to increase the power dissipating ability of the transistor. These devices utilize the natural methods of conduction, convection, and radiation to reduce case and junction temperatures and to increase overall reliability.

Transistor retainer and heat dissipators are relatively simple in their design and construction and require little or no maintenance other than to ensure that the mounting hardware used to attach the heat dissipator to the chassis or metal bracket is secure and in place. In some cases, the removal of the heat dissipator may be required to reach inaccessible locations in equipment. In removing the dissipator, be careful not to damage the fragile, brittle leads of the transistor. When the dissipator is disassembled, all mating surfaces should be inspected to ensure that they are free from burrs or sharp edges. Any existing burr or sharp edge must be removed, otherwise, the thin mica washer or mica material used to insulate electrically the transistor from the chassis or metal surface areas may be punctured, impairing its dielectric properties and forming a path for current leakage.

NOTE: A thin coat of silicone grease or similar compound applied to the surface areas of the mica insulating material will improve its dielectric properties.

SERVICING PRECAUTIONS

Since the transistor is probably one of the most reliable components, it should be the last to be suspected as defective. Again, this is contrary to the long-established practice used in electron tube equipment, where the tubes normally are checked first. Because of their reliability, transistors are generally soldered in the circuit, particularly in printed circuits. Removing and testing each transistor will not only unnecessarily subject the transistor to heating, but may also result in damage to some other component, particularly in the case of a printed circuit board.

However, if the transistor itself is suspected it can be removed from the equipment for testing. In sets employing sockets for the transistors, it is only necessary to remove the transistor from the socket. If the transistor is soldered and it becomes necessary to unsolder it, extreme care must be taken to prevent damage to the transistor by the heat from the soldering iron. Also, the leads must be handled carefully as they are very brittle. CAUTION: Never remove or replace a transistor while the battery or power source is connected to the set. Failure to observe this precautionary measure may result in damage to the transistor from surge currents, etc.

Although generally more rugged than the electron tube, the transistor is affected by electric shock, heat and humidity.

One of the most frequent causes of damage to semiconductor units is the electrostatic discharge from the human body when the unit is handled. Such damage may be avoided by discharging the body to the equipment before handling the unit.

A semiconductor unit may also be damaged by r-f fields. It is therefore essential that the unit be protected by a metal container until ready for use, at which time the equipment should be deenergized before the semiconductor is inserted.

When it becomes necessary to replace a transistor where the leads have been soldered the following precautions should be taken. Before removing the old transistor, note the orientation of the collector, base, and emitter leads. Preshape and cut the leads of the new transistor to the proper length, using sharp cutters to prevent undue stress on the leads entering the transistor. (Pigtail leads should have a minimum clearance of 1/16 inch between bend and transistor body. Shape any bend required in a gradual curve. Sharp (90°) bends are not acceptable. Then, with the transistor properly positioned, solder the leads to the connections, using the proper solder, soldering iron, and a heat sink.

For stability of the electrical characteristics, the maintenance of the hermetic seal of the transistor cannot be over emphasized. This seal in addition to maintaining the carefully controlled environment in which the transistor operates also excludes moisture which causes instability. (While a transistor is warming up after exposure to low temperature moisture may collect on the transistor surfaces causing a large temporary increase in the collector current.)

The minute power requirements of transistor circuits make it economically feasible to operate transistorized equipment with batteries even where the equipment is subject to continuous use. By using careful construction techniques, the transistors of today are capable of operation in excess of 30,000 hours at maximum rating without appreciable degradation. Either conventional zinc-carbon batteries or the newer mercury batteries may be used. BATTERY ELIMINATORS SHOULD NEVER BE USED AS THE SOURCE OF POWER FOR TRANSISTORS OR ANY OTHER SEMICONDUCTOR DEVICE. Because of the low current drain of transistor circuits, the voltage regulation of battery eliminators is poor.

In handling transistors, it should be remembered that temperature is the most important factor affecting transistor life, and that it is important to keep the ambient temperature as low as possible. It has been estimated that for every 10° C the junction temperature is lowered, the life of the transistor is doubled.

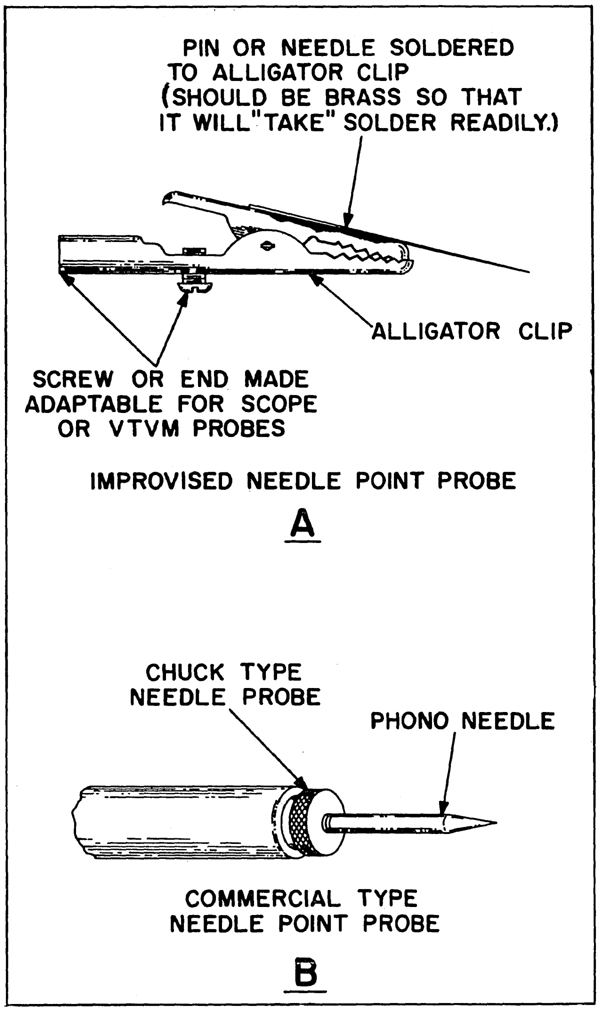
PRINTED CIRCUITS

Although the troubleshooting procedure for printed circuits are similar to those for conventional circuits, the repair of printed circuits requires considerably more skill and patience. The printed circuits are small and compact; thus personnel should become familiar with the special servicing techniques required.

In all instances, it is advisable to first check the defective printed circuit before beginning work on it to determine whether any prior servicing has been performed.

The defective part should be pinpointed by a study of the symptoms and by careful and patient analysis of the circuit (using the logical six-step method) before attempting to trace trouble on a printed circuit board. Ascertain whether the conducting strips are coated with a protective lacquer, epoxy resin, or similar substance. If so, carefully scrape it away, or, better still, use a needle or chuck type needle probe, as shown in figure 18-4, which will easily penetrate the coating so that continuity checks can be made.

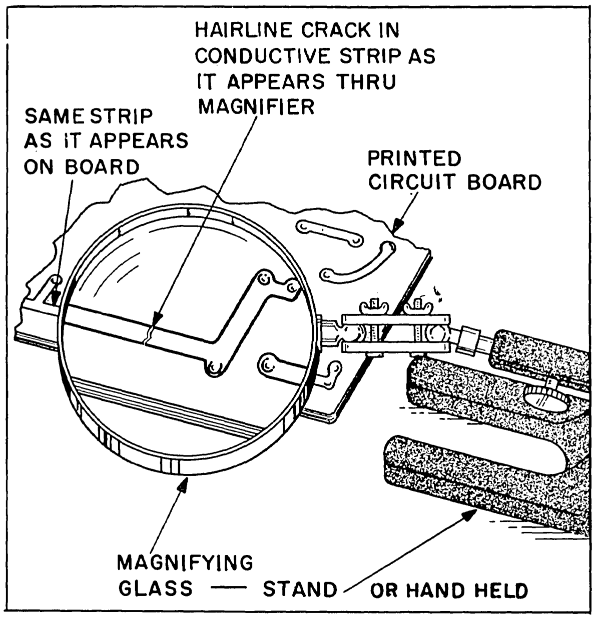
Breaks in the conducting strip (foil) can cause permanent or intermittent trouble. In many instances, these breaks will be so small that they cannot be detected by the naked eye. These almost invisible cracks (breaks) can be located only with the aid of a powerful hand- or stand-held magnifying glass, as illustrated in figure 18-5.


PIN OR NEEDLE SOLDERED TO ALLIGATOR CLIP (SHOULD BE BRASS SO THAT IT WILL "TAKE" SOLDER READILY.)
ALLIGATOR CLIP
SCREW OR END MADE ADAPTABLE FOR SCOPE OR VTVM PROBES
IMPROVISED NEEDLE POINT PROBE
A
CHUCK TYPE NEEDLE PROBE
PHONO NEEDLE
COMMERCIAL TYPE NEEDLE POINT PROBE
B

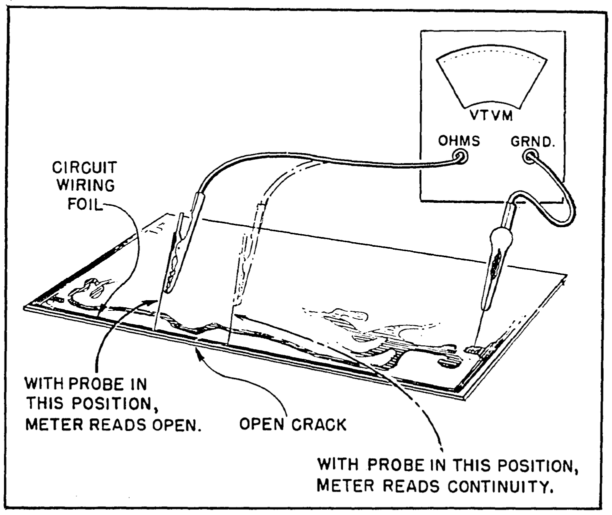
70.109
Figure 18-4.—Needle probes.

The most common cause of an intermittent condition is poorly soldered connections. Other causes are: broken boards, broken conducting strips, fused conducting strips, arc-over, loose terminals, etc.

To check out and locate trouble in the conducting strips of a printed circuit board, set up a multimeter (one which DOES NOT pass a current in excess of 1 ma) for making point-to-point resistance tests, as shown in figure 18-6, using needle point probes. Insert one point into the conducting strip, close to the end or terminal, and place the other probe on the terminal or opposite end of the conducting


HAIRLINE CRACK IN CONDUCTIVE STRIP AS IT APPEARS THRU MAGNIFIER
SAME STRIP AS IT APPEARS ON BOARD
PRINTED CIRCUIT BOARD
MAGNIFYING GLASS — STAND OR HAND HELD

70.110
Figure 18-5.—Using a magnifying glass to locate a hairline crack.


VTVM
OHMS
GRND.
CIRCUIT WIRING FOIL
WITH PROBE IN THIS POSITION, METER READS OPEN.
OPEN CRACK
WITH PROBE IN THIS POSITION, METER READS CONTINUITY.

70.111
Figure 18-6.—Using a VTVM to locate a break in a conductive strip.

strip. The multimeter should indicate continuity. If the multimeter indicates an open circuit, drag the probe along the strip (or if the conducting strip is coated, puncture the coating at intervals) until the multimeter indicates continuity. Mark this area and then use a magnifying glass to locate the fault in the conductor as shown in figure 18-5.

CAUTION: Before using an ohmmeter for testing a circuit containing transistors or other voltage-sensitive semiconductors, check the current it passes under test on all ranges. DO NOT use a range that passes more than 1 ma.

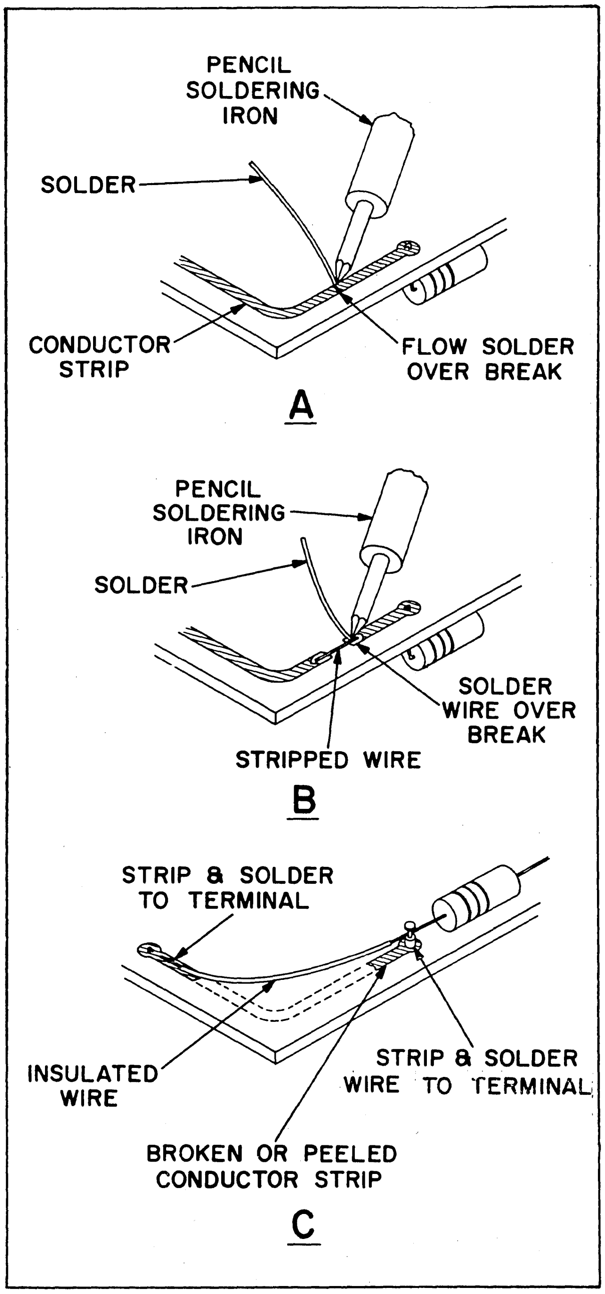
If the break in the conducting strip is small, lightly scrape away any coating covering the area of the conducting strip to be repaired. Clean the area with a firm-bristle brush and approved solvent (see Handbook of Cleaning Practices, NavShips 250-342-1), then repair the cracked or broken area of the conducting strip by flowing solder over the break (fig. 18-7A). If there is any indication that the strip might peel, bridge the break with a small section of bare wire (approximately 2 inches) by the method shown in figure 18-7B. Apply solder along the entire length of the wire to bond it solidly to the conducting strip. Considerable care must be exercised in applying the solder to prevent it from flowing onto or near an adjacent strip. Keep the solder within the limits of the strip that is being repaired.

If a strip is burned out, or fused, cut and remove the damaged strip. Connect a length of insulated wire across the breach or from solder-point to solder-point (fig. 18-7C).

It is best not to glue or bond a conducting strip that has been lifted or peeled from the board at a terminal or solder point. Instead, clip off the raised section and replace it with insulated hookup wire from solder-point to solder-point.

Printed conductor-to-conductor circuit boards are frequently subject to leakage and shorts, especially if the spacing between conductors is very close, or by the careless formation of a solder bridge between the conducting strips during soldering. NOTE: After repairs, always scrutinize the board for solder droppings that may cause possible shorts.

Frequently, a low-resistance leakage path will be created by moisture and/or dirt that has carbonized onto the phenolic board. This leakage can be detected by measuring the suspected circuit with a multimeter. To overcome this condition, thoroughly clean the carbonized area with solvent (methyl Chloroform GM 6810-664-0387) and a stiff brush. If this does not remove it, use a scraping tool (spade end of a solder-aid tool or its equivalent) to



70.112
Figure 18-7.—Three methods of repairing broken conducting strips.

remove the carbon, or drill a hole through the leakage path to break the continuity of the leakage. When the drilling method is used, be careful not to drill into a part mounted on the other side.

Occasionally, a conductor will rupture or fuse, usually because of a current overload. Generally, the rupture, or fusing, is the result of limited spacing and narrow conductors. Do not try to repair this type of damage, other than to bridge the rupture, or fused area, with a length of insulated wire (fig. 18-7C).

Most printed circuit boards have areas of conduction, known as GROUNDING CONDUCTORS, at each edge of the board or on the parts-mounted side of the board. These grounding conductors are conducting strips, used for grounding parts as a mounting contact for the chassis or common ground. Sometimes an intermittent condition will result if the grounding, screws or mounting screws, become loose. If this occurs tighten the screws and then solder a good bond directly from the grounding strip to the chassis or equipment ground. If this is not practical, bond the screws (after tightening) with an epoxy resin or similar compound.

The most common cause of broken boards is droppage. Some boards are broken because of careless handling by service personnel while the equipment is under repair. Be extremely careful at all times while handling a board. Do not flex the board indiscriminately; be especially careful when removing the board or replacing parts; do not force anything associated with the board.

A printed circuit board can be flexed to a certain extent; however, flexing may break the board which must then be replaced at a considerable loss of time. To prevent this possibility, it is always a good policy to use a chassis-holding jib or vise (shown later) when servicing printed boards.

Before repairing a broken printed-circuit board, assess the damage. Inspect the condition of the board and the extent of the break. If the board is not too complicated or the damage not too extensive the board can probably be repaired.

After the repairs are completed, clean the repaired area with a stiff brush and solvent. Allow the board to dry thoroughly, and then coat the repaired area with an epoxy resin or similar compound. This coating protects the repaired area and strengthens it.

NOTE: When a board is broken, it is much better to replace the entire board. The repair techniques given above are for temporary repair only.

SPECIAL TECHNIQUES

It is always desirable to replace parts on a printed circuit board without applying heat directly to the conducting strip. This procedure prevents damage to the printed circuit conductors, feed-through devices, eyelets, or terminals, and saves time in repair. It also prevents damage to semiconductors and other heat-sensitive parts that may be in close proximity to the part being repaired.

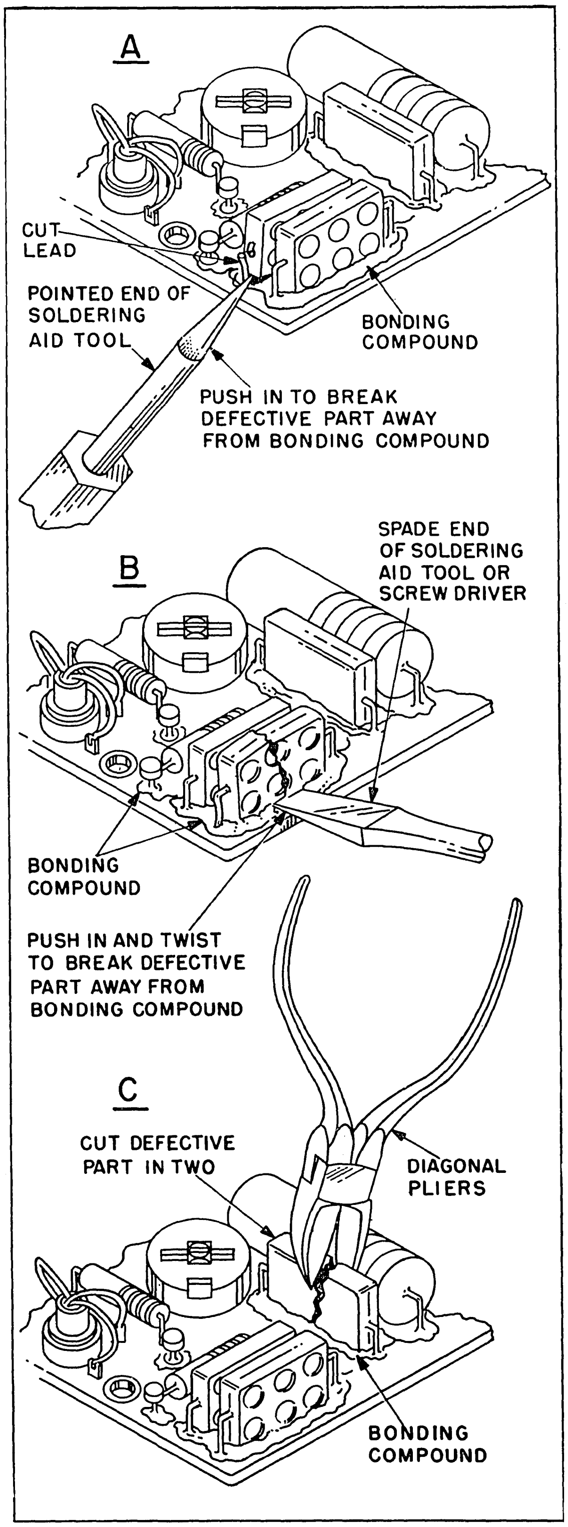
Replacing parts requires that each type of part mounting be considered individually for the best method of removal.

A part to be removed may be too close to a heat-sensitive semiconductor or other part to allow the hot pencil-soldering iron to be applied. A quick test to determine this safe distance is to place your finger between the semiconductor (or heat-sensitive part) and the part to be removed. Place the hot soldering iron in the position to be used. If the heat is too great for your finger it is too hot for the semiconductor. After determining that the heat-sensitive part is too close, place a shield (asbestos or like substance) between the parts before applying the hot soldering iron, and place heat sink clamps on all leads from the heat-sensitive part.

Solid-state parts and their associated circuitry are extremely sensitive to thermal changes. Therefore, particular care must be taken to prevent exposing them to heat. Heat sink and shunts must be applied with shields inserted to protect the associated parts any time repair or removal of a part requires the use of a hot soldering iron. Solid-state parts and associated assemblies require the same care in handling and skill of repairing that is applied to assemblies in equipment of unitized or modular construction containing transistors, tantalum capacitors, crystals, etc.

Removal of an axial-lead part that has been bonded to a printed circuit board (with an epoxy resin or similar compound) can be accomplished by breaking the defective part or by applying heat to the bonding compound. The method to be used depends upon the part itself and its location.

If the defective axial-lead part cannot be removed by heat, cut or break the part away from the bonding compound. Figure 18-8 illustrates two different methods of breaking the part away from the bonding compound where the part is too close to other parts to use cutting pliers. In some instances, the part to be replaced is so closely positioned between other parts that one lead must be cut close to the body of the defective part to permit application of the prying tool. Wherever possible, cutting the defective part with end-cutting pliers


A
CUT LEAD
POINTED END OF SOLDERING AID TOOL
BONDING COMPOUND
PUSH IN TO BREAK DEFECTIVE PART AWAY FROM BONDING COMPOUND
B
SPADE END OF SOLDERING AID TOOL OR SCREW DRIVER
BONDING COMPOUND
PUSH IN AND TWIST TO BREAK DEFECTIVE PART AWAY FROM BONDING COMPOUND
C
CUT DEFECTIVE PART IN TWO
DIAGONAL PLIERS
BONDING COMPOUND

70.113
Figure 18-8.—Removing a defective part from bonding compound.

or diagonals, as shown in part C, is the preferred method to use.

Regardless of which tool is employed (round-pointed or spade type), great care must be used in its application to prevent the printed circuit board or other parts from being damaged or broken. Apply the point of the tool against the bonding compound, between the part and the printed circuit board. Use the tool in such a manner that it works away the bonding compound from the part to be broken away until enough has been removed for the tool to exert pressure against the part. Keep the leverage surface area of the tool flat against the surface of the printed circuit board; this helps to prevent the tool from gouging or breaking the board.

CAUTION: NEVER APPLY MUCH PRESSURE AGAINST A PRINTED CIRCUIT BOARD.

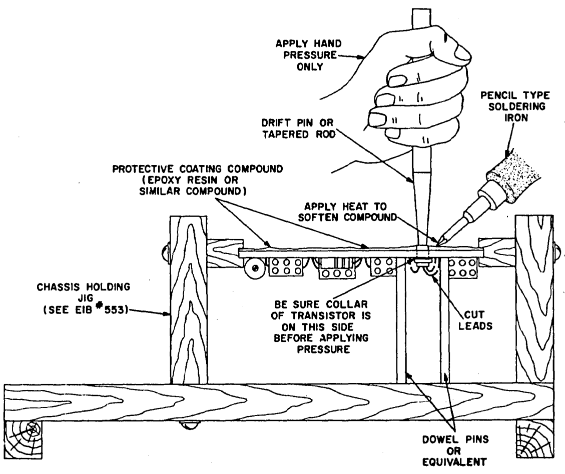
After the defective part has been removed from the bonding compound, remove the leads or tabs from their terminals on the printed circuit board. Clean the area thoroughly before installing the new part. Do not remove the compound left on the board under the removed part unless its condition requires it. The mold left in the compound should be the same as the new part; thus, inserting the new part in this mold helps to secure it from vibration. After the repairs have been completed and the circuit tested, spray the newly soldered area with an insulating varnish or equivalent. Coat the new part or parts with a bonding compound (ECCO-BOND-"55" by Emerson and Cuming, Inc.; relix-R-313 by Carl H. Briggs Co.) or equivalent.

To replace a proven defective transistor, first cut all of its leads, and then remove it from the assembly. Transistors are mounted on circuit boards in many different ways; thus it is necessary to study how a particular transistor is secured before attempting to remove it. A transistor with clamp-type mounting requires only a pointed tool between the clamp and the transistor to remove it. A transistor mounted in a socket may have a wire or spring clamp around it. Remove this clamp before pulling the transistor out of the socket. In some instances the transistor is bolted through the board. Remove the nut and washer, and then remove the transistor. Where vibration is a prime factor, the manufacture mounts the transistor through the circuit board and bonds it (with epoxy resin or similar compound). For this type, a flat-ended round-rod type tool (drift punch) of a diameter less than that of the transistor case is required. Be sure that the printed circuit board on which the transistor is mounted is secured in a proper device, and in such a way that pressure exerted against the board will be relieved by a proper support on the other side (fig. 18-9). Apply a hot-pencil soldering iron to the bonding compound and simultaneously apply the drift punch against the top of the transistor, exerting enough pressure to remove the transistor from the softened compound, and then on through and out the board (fig. 18-9).

Before installing the new transistor, great care must be taken to prepare the part for installation.

Test the transistor in a transistor tester (TS-1100/U or equivalent) before installing. This precaution will assure that the transistor is good before it is installed. For several reasons transistors can and do become defective in storage. Therefore, always check them before installation.

Pre-shape and cut the new transistor leads to the shape and length required for easy replacement. Use sharp cutters, and do not place undue stress on any lead entering the transistor. The leads are fragile, and are therefore susceptible to excessive bending or too sharp a bend. Shape any bend required in a gradual curve, and at least 1/4 inch to 3/8 inch from the base of the transistor. A safety measure which can be taken to ensure that the lead will not break off at the base is to use two pairs


APPLY HAND PRESSURE ONLY
PENCIL TYPE SOLDERING IRON
DRIFT PIN OR TAPERED ROD
PROTECTIVE COATING COMPOUND (EPOXY RESIN OR SIMILAR COMPOUND)
APPLY HEAT TO SOFTEN COMPOUND
CHASSIS HOLDING JIG (SEE EIB #553)
BE SURE COLLAR OF TRANSISTOR IS ON THIS SIDE BEFORE APPLYING PRESSURE
CUT LEADS
DOWEL PINS OR EQUIVALENT

70.114
Figure 18-9.—Removing a transistor that has been through-board mounted.

of needle-nose pliers. With one pair grasp the lead close to the transistor base, while shaping the rest of the lead with the other pair.

NOTE: The above procedure and precaution should be applied to any and all semiconductors, tantalum capacitors, and other miniaturized parts in equipment of modular or unitized construction.

After the remaining pieces of the defective transistor-terminal leads have been removed and the terminals on the board cleaned and prepared connect the new transistor to its proper terminals.

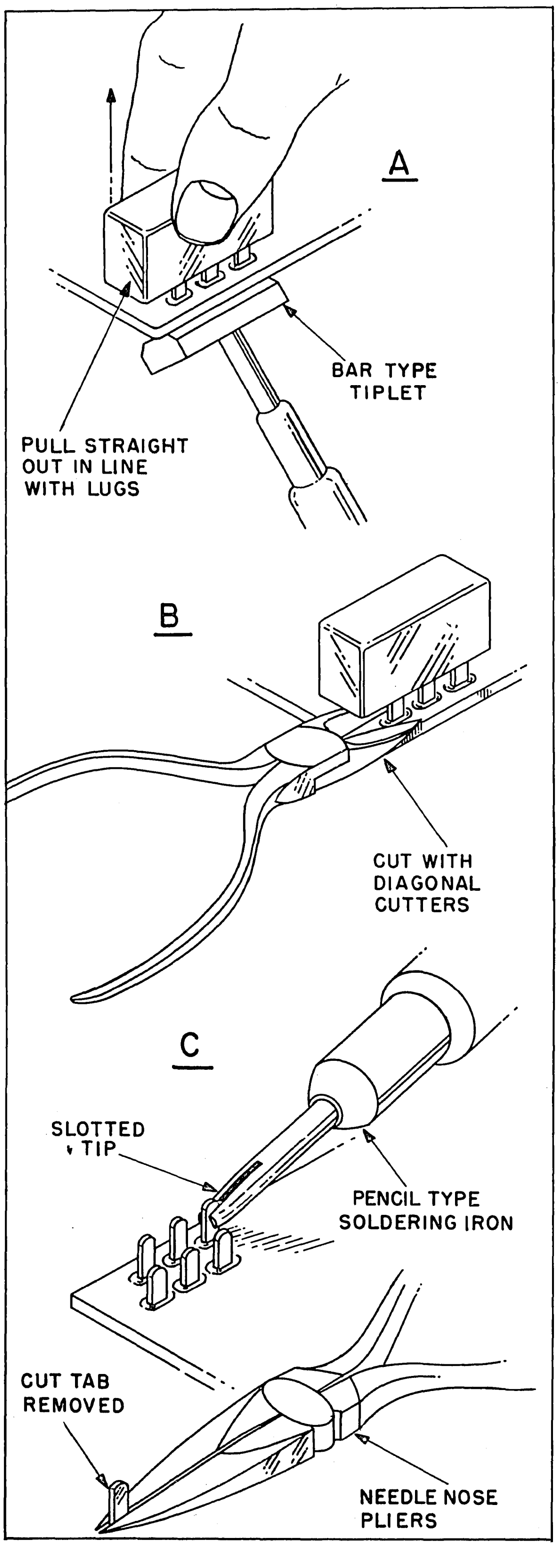
REMEMBER: Handle any semiconductor or miniaturized part carefully; be gentle and be precise.

When the defective transistor is removed from a through-board mounting, and bonded, care must be taken that the new transistor clears the hole before it is connected to its terminals. If the hole is too large, shim with a thin plastic sleeve (fabricated). If the hole is too small, ream it to accept the new transistor. Rebond the fitted transistor after TESTING the repaired circuit, and it is proven to be operative.

CAUTION: DO NOT USE HEAT TO REBOND REPLACED SEMICONDUCTORS.

To remove and replace a multilug part, such as a transformer, choke, filter, or other similar potted, canned, or molded part, release the part from its mounting before disconnecting or cutting its conductors. Before applying pressure to remove such a part, inspect it carefully to be sure that the part is completely free of all its connections to the printed circuit board, and that all bent or twisted mounting lugs have been straightened; otherwise, you may break the board by applying undue pressure to it. Never wrench or twist a multilug part to free it, because this will cause the conducting strip to become unbonded from the board. Work this type of part in and out in line with its lugs, while applying a hot-pencil soldering iron (fig. 18-10A) using a bar type tiplet adapter or similar desoldering tool.

Whenever possible, cut the conducting or mounting leads and lugs of the defective multilug part on the mounting side of the board (fig. 18-10B). Heat and straighten the clipped leads with a hot-pencil soldering iron and slotted soldering-aid tool (or slotted soldering-iron tiplet adapter or similar desoldering tool) applied to the circuit side of the board; pull the leads or tabs through with pliers as shown in 18-10C.


A
BAR TYPE TIPLET
PULL STRAIGHT OUT IN LINE WITH LUGS
B
CUT WITH DIAGONAL CUTTERS
C
SLOTTED TIP
PENCIL TYPE SOLDERING IRON
CUT TAB REMOVED
NEEDLE NOSE PLIERS

70.115
Figure 18-10.—Removing a defective multi-lug part.

To replace the new multi-lug part, check to be sure that all of the lead holes or slots are free and clean, allowing easy insertion of the multilug part. DO NOT FORCE ANY PART INTO POSITION ON A PRINTED CIRCUIT BOARD. If the part does not position easily, check and rework the terminals and holes (or slots) until it does seat freely then proceed to solder.

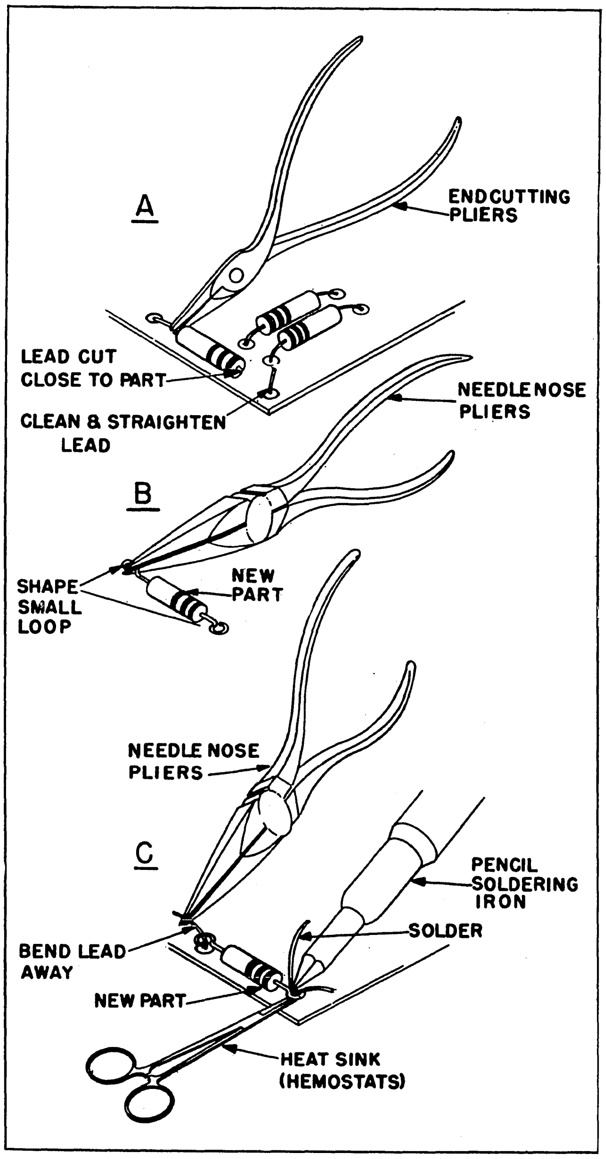
Be very careful when replacing defective parts that have leads terminating on stand-offs, feed-through terminals, etc. In most instances, stand-offs, feed-through terminals, are very small, and mounted on a thin phenolic board; thus they are susceptible to damage by heat and undue pressure.

EMERGENCY TECHNIQUES

In many instances there is a need for a time-saving technique and procedure for electronic assembly emergency repair. It is desirable, when making an emergency repair, to avoid unnecessary disassembly to expose the defective part when testing and/or repairing. In many instances this can be accomplished by removing only the cover from the assembly.

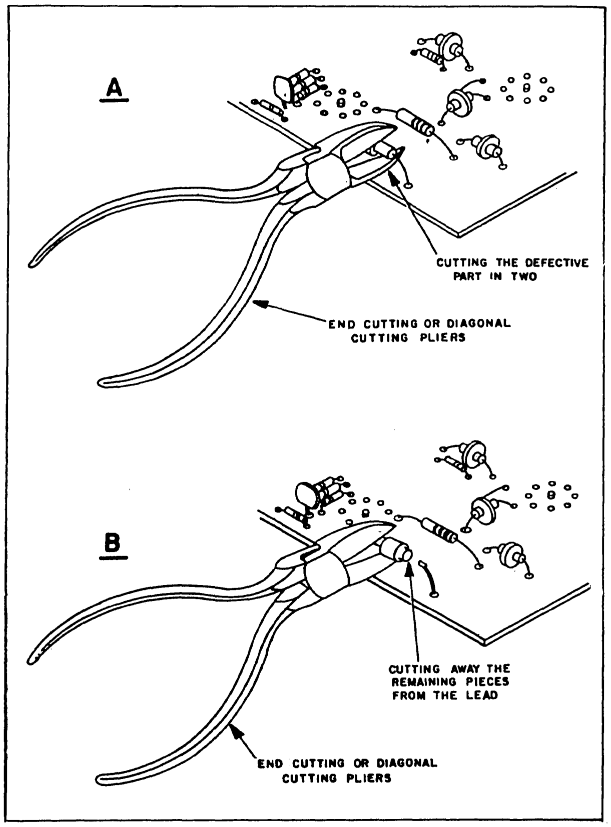
To remove and replace an axial-lead part (a part mounted by leads that extend from each end, such as a common resistor or capacitor), cut the leads as close as possible to the body of the part, and then connect the leads of the replacement part to the leads remaining on the board. The cutting is accomplished with a pair of end-cutting pliers (fig. 18-11A). Clean and straighten the leads remaining on the board. Fashion small loops in the leads of the replacement part (fig. 18-11B) making the loop size and lead length such that the loops slip easily over the leads projecting from the board. Secure these connections by bending the old leads away from the part. Place a heat-sink clamp on the lead from the board, between the board and the connection to be soldered, and then solder the connection (fig. 18-11C). The heat sink prevents the leads connected to the board from becoming unsoldered and causing a short, or open circuit. Always check to be sure that the old leads are properly connected to the conducting strip.

If cutting the leads of a defective axial-lead part would result in leads that are too short for the replacement part to be connected properly, cut the faulty part in half with a pair of diagonal or end-cutting pliers (fig. 18-12A). Then carefully cut away the pieces of the part from each lead (fig. 18-12B). This will yield leads of sufficient length to permit the replacement part to be fitted and soldered as shown in figure 18-11.



70.116
Figure 18-11.—Replacing a defective part by cutting its leads.

Considerable care must be taken when replacing a defective part that terminates on miniaturized standoffs, feed-through terminals, etc. These small terminals break easily from applied pressure, or they may melt loose from

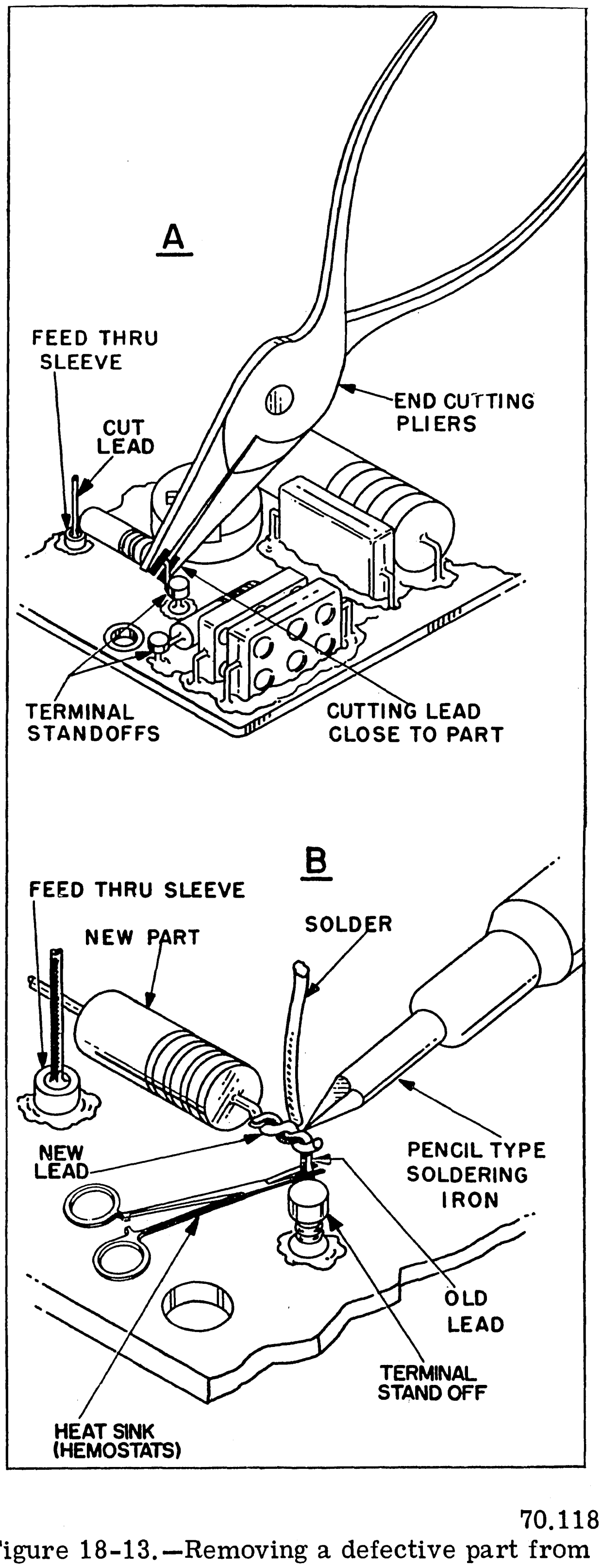

A
CUTTING THE DEFECTIVE PART IN TWO
END CUTTING OR DIAGONAL CUTTING PLIERS
B
CUTTING AWAY THE REMAINING PIECES FROM THE LEAD
END CUTTING OR DIAGONAL CUTTING PLIERS

70.117
Figure 18-12.--Cutting the defective part for maximum lead length.

excessive application of the hot soldering iron. Do not attempt this type of repair on an assembly unless there is no replacement available.

For emergency or temporary repair purposes, the following techniques may be used. Cut the lead close to the defective part (fig. 18-13A). Use a heat-sink clamp (or pliers) next to the terminal, then solder a spliced lead from the terminal to the new part (fig. 18-13B).

A helpful heat control technique is to place a small piece of beeswax (W9160-253-1172) on the terminal behind the heat sink. When the beeswax melts, the temperature limit has been reached. This is a warning to remove the source of heat immediately. Allow the area to cool thoroughly before attempting to complete the soldering of the connection. Apply a new piece of beeswax to the terminal, repeating this procedure until the connection is satisfactorily soldered.



70.118
Figure 18-13.—Removing a defective part from a miniature stand-off terminal.

It is best not to glue or bond a conducting strip on a printed circuit board that has been lifted or peeled from the board at a terminal or solder point. Instead, clip off the raised section and replace it with insulated hookup wire from solder-point to solder-point. However, for temporary or emergency repair, a loose or peeled strip may be bonded back onto the board, using a nonconductive bonding compound ECCOBOND "55" epoxy adhesive, or its equivalent. A silver conductive paint or similar material can also be used to repair printed circuit conductive strips. This technique is satisfactory for temporary or emergency repair, but is not satisfactory for permanent repair.

A broken printed circuit board may have to be repaired in an emergency where no replacement is available. Before repairing the broken board, assess the damage for the extent of the break and the amount of damage to the parts involved. If the board is not too complicated or the damage too extensive, the board can probably be repaired.

If the board is not completely broken but is only cracked, drill a hole at the end of each crack (fig. 18-14A) to prevent further lengthening of the break. Then repair the crack by placing a conductive material across the defective area.

If a small portion or corner of the board is broken off, it may be rebonded to the larger section with a nonconductive cement or its equivalent. If cementing is not feasible or does not hold satisfactorily, the pieces can be fastened together with wire staples cut from solid conducting wire of the diameter and length required, depending on the width of the conducting strip to be repaired.

To insert the staples, drill holes about 1/4 inch in from each side of the break (fig. 18-14B). The holes should be just large enough to accommodate the wire used for stapling. (This may vary, depending on the width of the conductive strip to be repaired.) Drill the holes through the conducting strips so that the staples will provide a good electrical contact across the break; this method will permit the use of enough staples to hold the pieces together without danger to shorts between conductors. If the break is sufficiently large, position additional staples at all points possible to give the board more support.

Where the adhesive and stapling methods described above do not provide structural strength or sufficient rigidity, splints or a doubler may be used. Strips of thin card material are glued across the fracture with a nonconductive adhesive. Where needed, additional strength may be obtained by gluing a plate of the card material to the splints with the nonconductive adhesive.

Rebond any loose conducting strips with a non-conductive bonding cement; then apply non-conductive cement to both sides of the break, and join the sections together (fig. 18-14B). Insert half of the measured and precut wire staples from top to bottom, and the other half from bottom to top, bending the ends flush against the board (fig. 18-14D). Solder these staples to the conducting strip (Fig. 18-14E).

After the repairs are completed, clean both sides of the repaired area with a stiff brush and an approved solvent. Allow the board to dry thoroughly, then coat the repaired area with an epoxy resin or similar compound. This coating protects and strengthens the repaired area.

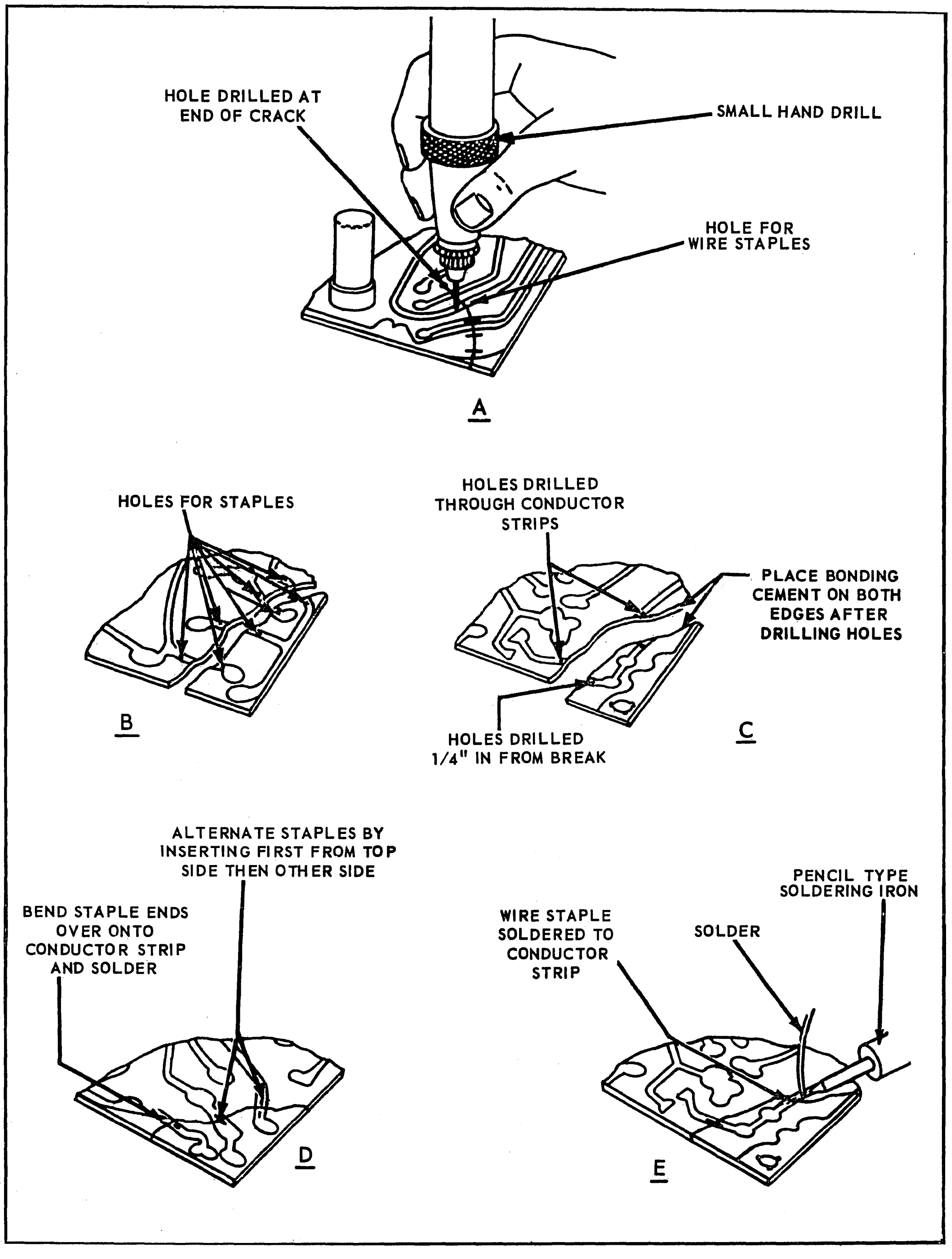
NOTE: The repair techniques given above are for emergency repair ONLY.

MODULAR ASSEMBLIES

This section provides information so that a technician, using the techniques and procedures discussed in this chapter, can repair and restore modular constructed equipment quickly and efficiently.

The following established definitions will be helpful in understanding the terms involved. A MODULE is defined as A UNIT or STANDARD of measurement-a fixed dimension. A MODULAR ASSEMBLY has outline dimensions which are multiples of a MODULE. An equipment which consists of replaceable assemblies (any type tubes, transistors, etc.), is said to be of unitized construction. MODULAR CONSTRUCTION, then, is a type of unitized construction consisting predominately of MODULAR ASSEMBLIES.

The original concept of many modular assemblies was that they should not be maintained in the field. The intention was to replace the assembly and ship it back to some repair facility. As assemblies became more complex, the point was soon reached where the extensive supply system required for the replacement concept was too costly. Many equipments built during this initial stage were potted with some secret ingredient to discourage maintenance personnel from tampering with the insides of a black box. When the Navy reassessed this concept, realizing that the fleet must maintain everything it could, most of the equipment manufacturers began to


HOLE DRILLED AT END OF CRACK
SMALL HAND DRILL
HOLE FOR WIRE STAPLES
A
HOLES FOR STAPLES
B
HOLES DRILLED THROUGH CONDUCTOR STRIPS
PLACE BONDING CEMENT ON BOTH EDGES AFTER DRILLING HOLES
HOLES DRILLED 1/4" IN FROM BREAK
C
ALTERNATE STAPLES BY INSERTING FIRST FROM TOP SIDE THEN OTHER SIDE
BEND STAPLE ENDS OVER ONTO CONDUCTOR STRIP AND SOLDER
D
PENCIL TYPE SOLDERING IRON
WIRE STAPLE SOLDERED TO CONDUCTOR STRIP
SOLDER
E

70.119

Figure 18-14.—Repairing a broken printed circuit board.

make components accessible. However, many technicians are still convinced that modular assemblies are impossible to repair.

This conviction may stem from a lack of experience in working with printed circuits and the other components in modular assemblies. While it is true that special tools and techniques are required, it is also true that satisfactory repairs can be made to any printed circuit by using just a little care and common sense. Actually, with a little experience, repairs can be made as easily as in conventional assemblies—often more easily because of improved accessibility.

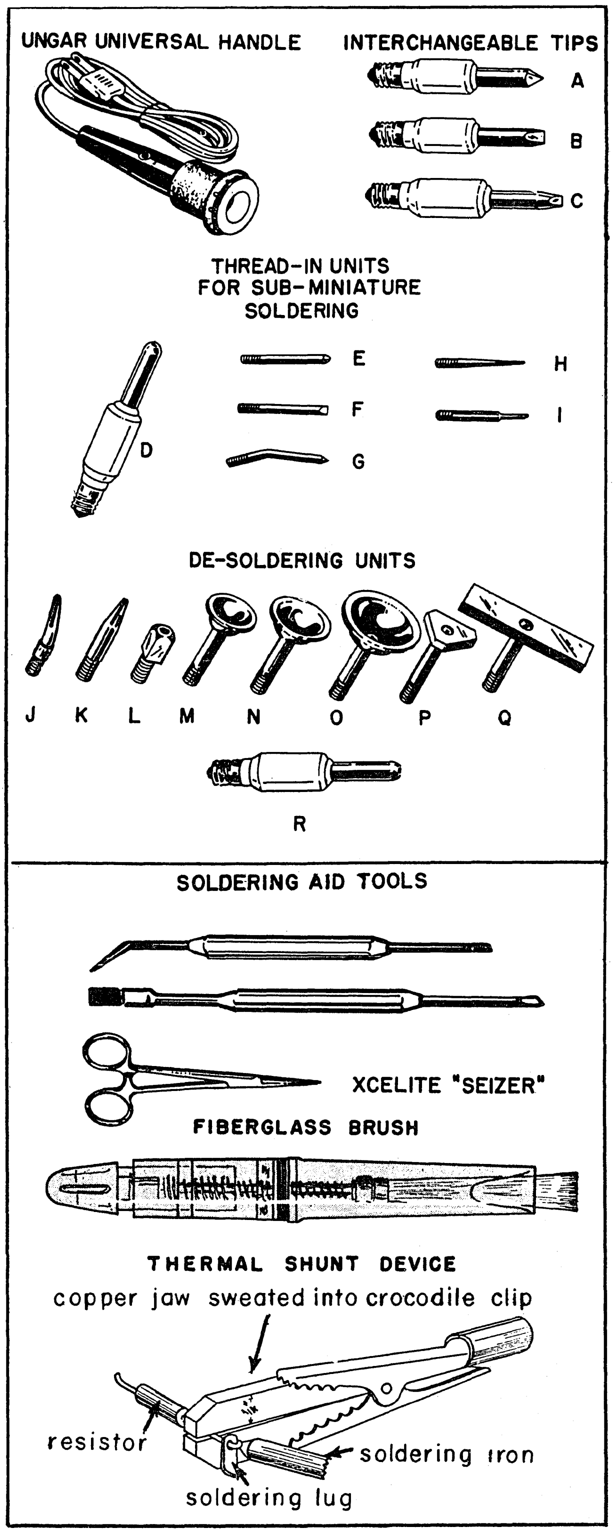
The techniques and procedures previously discussed concerning soldering techniques, transistors (and heat dissipators), printed circuits (and printed wire, etc.), removing and replacing components and/or parts, and special and emergency techniques, are applicable to all modular constructed MODULAR ASSEMBLIES.

A few examples of techniques and tools needed for the repair and maintenance of modular assemblies are shown in figures 18-15, 18-16, 18-17, and 18-18. Figure 18-15 shows some of recommended tools and aids for maintenance; figure 18-16 shows proper methods of applying and removing solder; figure 18-17 is an improvised tip for modular repair; and figure 18-18 gives a few additional soldering iron adaptations.

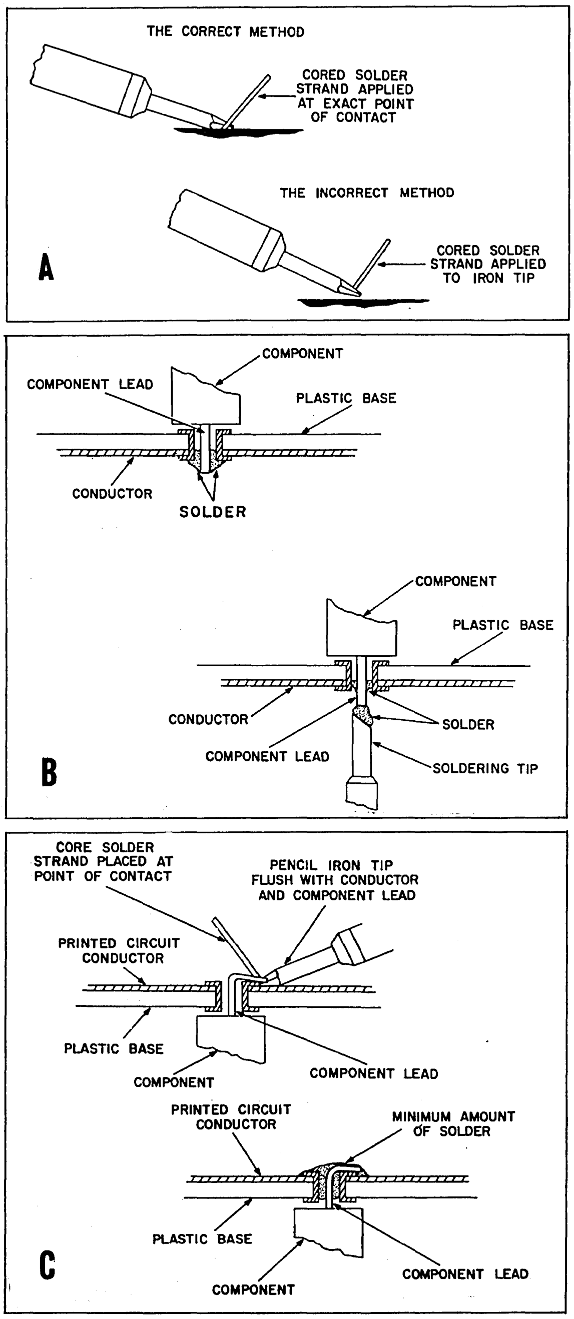
An easy way to reduce the number of hands required for working on printed circuit boards is to construct a chassis holding jig. The one shown in figure 18-19 is versatile enough to accommodate most types of modular assemblies. Along with supporting the board during repair, this jig will prevent slipping or flexing which could result in damage to the board.

The jig in figure 18-19 is constructed of 1-inch by 4-inch milled lumber of which only 2 feet 3 inches are required if cut to the dimensions given. Three round-head slotted machine screws (10-32, 1 3/4 inches long), three flat washers (0.199 ID, 3/8 inch OD—.064 inch TK), and three common hexagon nuts (No. 10-32) are required. The fixed head and feet are dowel fitted and glued as shown in the illustration. The jig illustrated will hold a modular assembly up to approximately 10 inches wide. The jig should be secured to the work area by utilizing the existing threaded holes on the top of the work bench or by use of "C" clamps or a vise.

Most of the other tools required for working with modular assemblies are readily available.



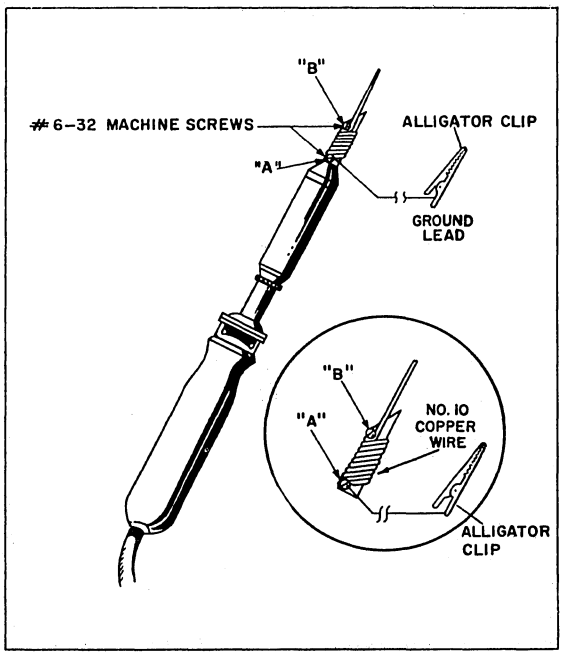
70.121
Figure 18-15.—Recommended tools and aids.



A. The correct and incorrect methods of soldering application.
B. Correct method for removing solder from component without damaging the printed wiring circuits.
C. Correct method for applying solder to a replaced component.

70.123

Figure 18-16.—Soldering techniques.



70.123A

Figure 18-17.—Improvised soldering tip for modular repair.

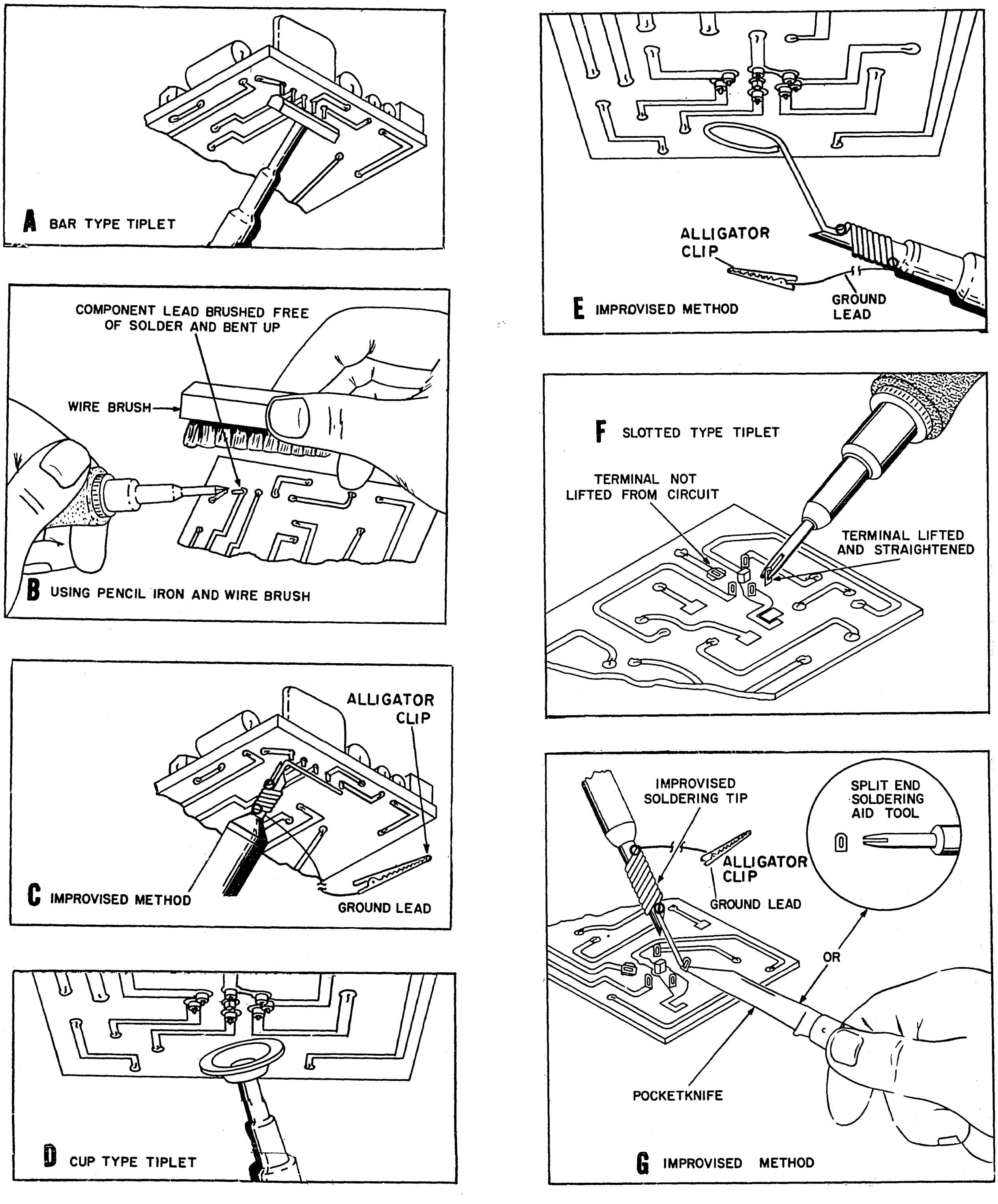
Items such as diagonal cutters, long-nose pliers, curved needle-nosed pliers, flush-cutting pliers, and tube socket adapters should already be in the shop.

Care in handling and proper packaging, to provide adequate protection against damage in transit or storage, is a must for an electronic assembly or associated repair part. In many cases, misplacement, improper identification or nomenclature, and damage to equipment repair parts in transit and storage are a direct result of thoughtless, careless action in the handling or packaging of the replacement or repairable part by the shipping activity.

REMEMBER: Assembly parts are fragile; careless handling and packaging may damage a replacement or repairable electronic assembly or associated part beyond use.

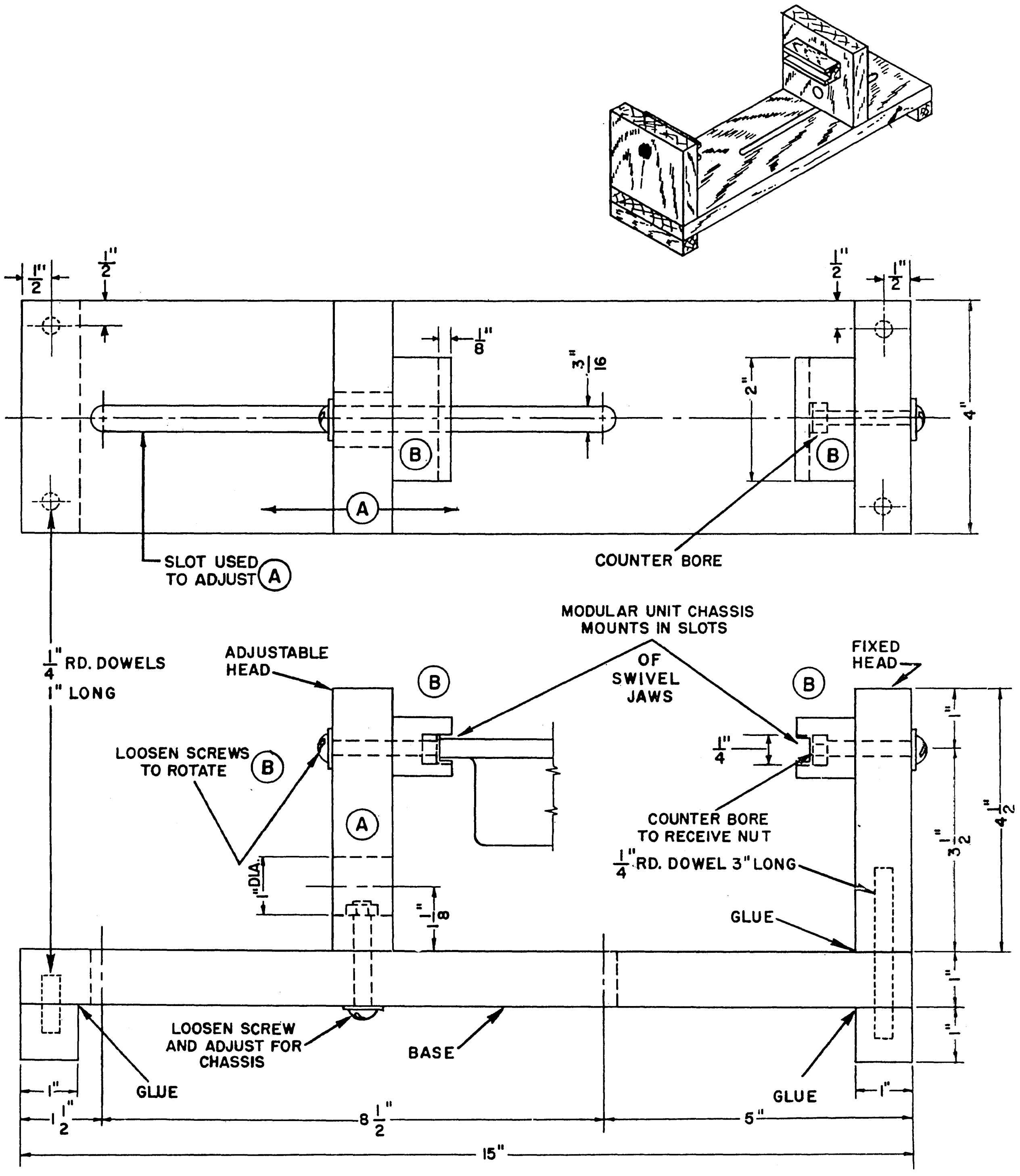
Care in handling and protection from damage are just as important for a defective module that can be repaired as for a new module.

Modular assemblies are shipped in accordance with the applicable packaging specifications. When the issuing activity receives the assembly, the outer casing (crate or carton with



70.124

Figure 18-18.—Special soldering iron adaptations.



SLOT USED TO ADJUST (A)
COUNTER BORE
MODULAR UNIT CHASSIS MOUNTS IN SLOTS OF SWIVEL JAWS
1/4" RD. DOWELS 1" LONG
ADJUSTABLE HEAD
FIXED HEAD
LOOSEN SCREWS (B) TO ROTATE
COUNTER BORE TO RECEIVE NUT
1/4" RD. DOWEL 3" LONG
GLUE
LOOSEN SCREW AND ADJUST FOR CHASSIS
BASE
15"
8 1/2"
5"
4"
4 1/2"
3 1/2"

70.125

Figure 18-19.—Chassis holding jig.

the paper packing) is removed and the assembly is stored in a watertight package until drawn by the using activity. Thus the using activity receives with a new module the necessary packaging material to properly protect a defective module.

The correct methods and the proper material to use for protective packaging of defective modular assemblies are shown in figures 18-20, 18-21 and 18-22. The material shown is available to all activities and should be used as prescribed for storing or transferring defective modules until they are received by a shipping facility, which will properly package them for the trip to the factory or restoration facility.

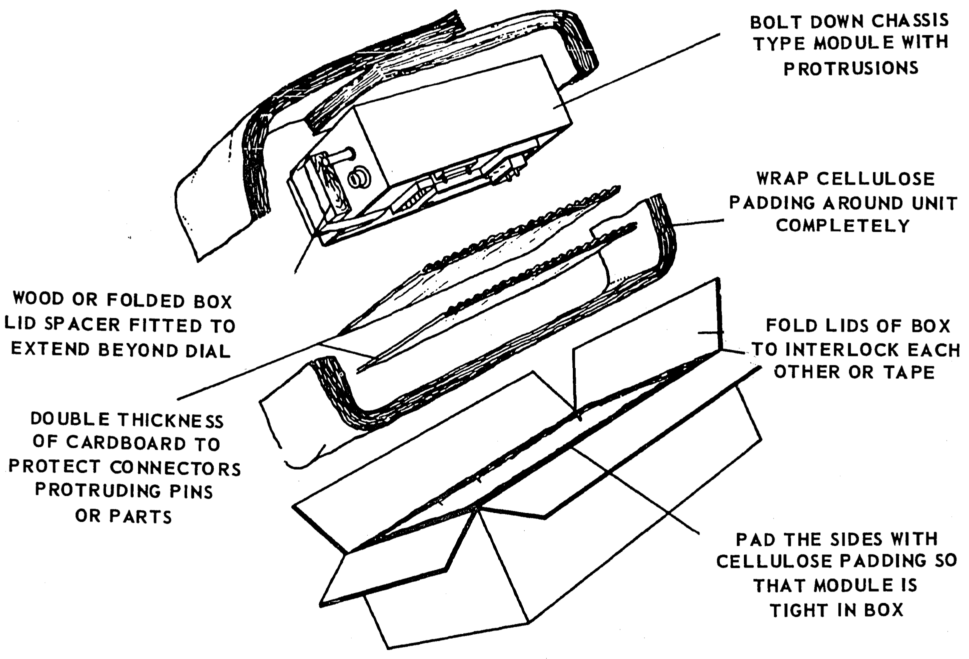
The using activity will have done its part in preventing transport damage to the modular assembly if the pins, shafts, dials, protruding parts, and so forth, are adequately fitted with packing spacers and if the module is properly wrapped with protective cellulose (Kimpack or similar material).

Desiccant crystals are normally packaged with assembled equipment crated for shipping. These are retained in a bag and placed within the crated or packaged equipment in such a manner as to prevent them from coming loose. Do not use these desiccant crystals when packaging DEFECTIVE modules. The modules must be packaged too tightly for the use of crystals in bag form, and loose crystals may cause unnecessary damage—plus a cleaning problem.

If a modular assembly becomes exposed to loose desiccant crystals, clean the assembly immediately.

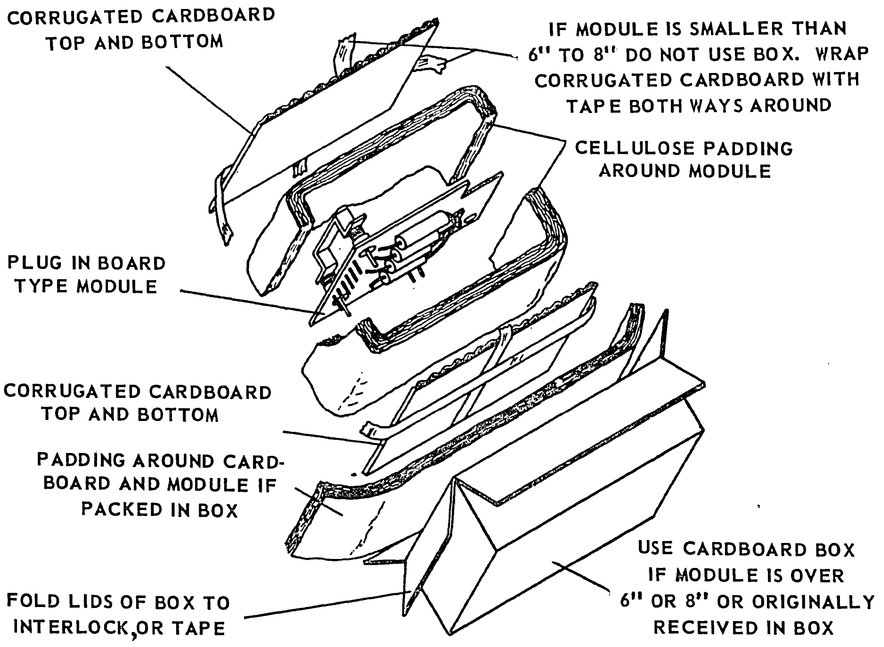
Much unnecessary damage has occurred to modular assemblies because of rough handling. Particular care must be given to the method of removing or inserting a module into the equipment. If the module is a plug-in board assembly, be sure the guide pins are properly aligned before pressing the assembly in place. If the board should tilt while it is being inserted, do not continue to press it into position; straighten it, and then apply even pressure to avoid tilting. Forcing any tilted or cocked modular assembly into position may result in bent or broken pins.

When removing a modular assembly, be sure to pull it straight out from the equipment. Do not cock, twist, pry, or carelessly jerk a module or modular assembly to remove it from its mounting or connector. Sometimes it may be necessary to loosen each screw little by little consecutively to prevent damaging by cocking.


BOLT DOWN CHASSIS TYPE MODULE WITH PROTRUSIONS
WRAP CELLULOSE PADDING AROUND UNIT COMPLETELY
WOOD OR FOLDED BOX LID SPACER FITTED TO EXTEND BEYOND DIAL
FOLD LIDS OF BOX TO INTERLOCK EACH OTHER OR TAPE
DOUBLE THICKNESS OF CARDBOARD TO PROTECT CONNECTORS PROTRUDING PINS OR PARTS
PAD THE SIDES WITH CELLULOSE PADDING SO THAT MODULE IS TIGHT IN BOX

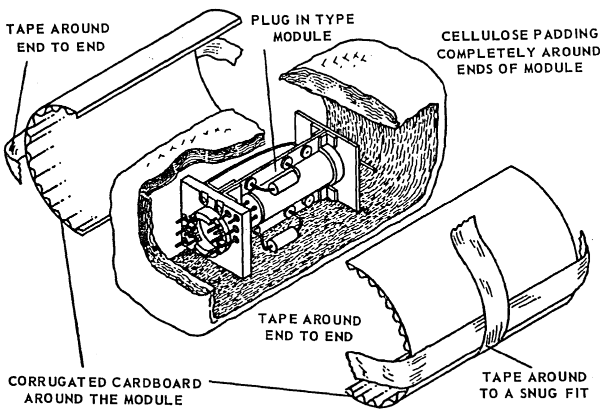
70.126

Figure 18-20.—Packing a bolt-down chassis.


CORRUGATED CARDBOARD TOP AND BOTTOM
IF MODULE IS SMALLER THAN 6" TO 8" DO NOT USE BOX. WRAP CORRUGATED CARDBOARD WITH TAPE BOTH WAYS AROUND
CELLULOSE PADDING AROUND MODULE
PLUG IN BOARD TYPE MODULE
CORRUGATED CARDBOARD TOP AND BOTTOM
PADDING AROUND CARD-BOARD AND MODULE IF PACKED IN BOX
USE CARDBOARD BOX IF MODULE IS OVER 6" OR 8" OR ORIGINALLY RECEIVED IN BOX
FOLD LIDS OF BOX TO INTERLOCK,OR TAPE

70.127

Figure 18-21.—Protective packaging of a plug-in board.


TAPE AROUND END TO END
PLUG IN TYPE MODULE
CELLULOSE PADDING COMPLETELY AROUND ENDS OF MODULE
TAPE AROUND END TO END
CORRUGATED CARDBOARD AROUND THE MODULE
TAPE AROUND TO A SNUG FIT

70.128

Figure 18-22.—Packaging method for a plug-in module.

Because of the miniaturization of parts for modular construction, leads, connectors, pins, and so forth, have been stiffened to ruggedize them. As a result, these fragile parts are brittle and will break easily if bent too often or pulled too hard. When handling a module that has been removed from its chassis, be careful not to press against the leads and pins; if a lead or pin is accidentally bent, do not try to straighten it unless it is absolutely necessary.

When repairing a modular assembly, be very careful that the tool employed does not inadvertently press against leads, pins, or other parts that are easily bent, for such pressure can destroy a good part, and cause needless repair.

One of the time-consuming elements of troubleshooting is the identification of specific components. In conventionally wired equipment, components are not always easy to locate; even the circuitry in the chassis can become confusing since related components are often positioned in decentralized areas of the chassis.

In equipment which includes printed circuit boards, identification of circuitry and components may be relatively simple; this type of circuit construction allows uniform placement of components and complete sectionalization of related circuitry. Just a quick, once-over glance of such circuitry is often all that a technician requires to formulate the overall layout of the chassis in his mind and quickly focus his attention on the area of particular concern.

Many of the commercial manufacturers have developed methods of quick identification. One of the most common ways is to impose a grid over a drawing of the board, and then furnish a table which lists the part location. Another technique is to number points of interest on the schematic, then provide a pictorial guide to locate the points on the board.

Circuit tracing of the printed wiring board may be simpler that that of conventional wiring due to increased uniformity. If the wiring board is translucent, a 60-watt light bulb placed underneath the side being traced will facilitate circuit tracing. Test points can be located in this manner without viewing both sides of the board.

Resistance or continuity measurements of coils, resistors, and some capacitors can be made from the component side of the board. In some cases, a magnifying glass will help in locating very small breaks in the wiring. Voltage measurements can be made on either side of the board. However, a needle point probe is needed to penetrate the protective coating on the wiring. Hairline cracks can be located by making continuity checks as shown in figure 18-6.

A number of general precautions are necesessary when working with modular assemblies.

Observe power supply polarities when measuring the resistance of the circuits of modular assemblies containing transistors, or other semiconductors. Such parts are polarity-and voltage-conscious. Reversing the plate-voltage polarity of a triode electron tube will keep the stage from operating; but generally will not injure the tube; however, reversing the voltage applied to a transistor, or other semiconductor, will ruin it, INSTANTLY AND PERMANENTLY.

Since transistors and similar components require different power supply connections, the personnel who work with these parts must always be alert in connecting test equipment. Make sure that the correct polarity and range are observed. Recheck your work before turning on the power-the WRONG POLARITY will DESTROY the part.

GUARD AGAINST HIGH TRANSIENT CURRENT OR VOLTAGES when testing or servicing. A damaging transient pulse may be caused in a number of ways. The list that follows represents some of the most frequent accidental acts that should be and can be prevented.

1. Use of a-c power operated test equipment or soldering iron without first making certain that power line leakage current is not excessive. Use of an isolation transformer is a good precaution to employ with all test equipment and soldering irons operated on a-c power, unless it has been determined that the equipment contains a transformer in its power supply or shows no current leakage. With all test equipment (whether transformer-operated or not), it is good practice to connect a common ground lead first from the ground of the circuit to be tested, and then to the test equipment ground.

2. Application of high amplitude pulses from test equipment. The safest procedure is to start with a low output signal setting, and then proceed to apply the required signal levels. Be sure that the signal applied is below the rating given for the circuit under test. Relatively high current transients can occur when test equipment is connected to a circuit where low-impedance paths exist.

3. Moving loose connections, disconnecting parts, inserting or removing transistors or similar components, and changing modular units, while the equipment power is on or while the circuit is under test. Moving a loose connection, or any of the actions mentioned, will cause an inductive kickback (due to stray inductance, if nothing else). This can be prevented by being sure that all parts in the circuit are secure before starting the test or turning on the equipment power. Be sure to remove all possible capacitance charges from parts and test equipment before applying them to a modular assembly. When changing modular assemblies, be sure the equipment power is off.

TRANSISTORIZED TRAINING AIDS

There are two general requirements for a skilled electronics technician. First, he must have a good knowledge of the theory, construction, and design features of the electronic equipment; and second, he must have sufficient mechanical skill and knowledge to successfully install, repair, and maintain the equipment. No matter how methodically and quickly the technician can locate a defect in the equipment, the final results will be unsatisfactory unless he has the necessary skill to repair the equipment in a workmanlike manner. Under proper supervision, the necessary skill can be obtained by the personnel who complete the following recommended Transistor Training Aid Program.

TRANSISTOR TRAINING AID PROGRAM

There are two general requirements for a skilled data systems technician. First, he must have a thorough understanding of the theory, construction and design features of data processing equipment; and second, he must have sufficient mechanical skill and knowledge to successfully install, repair, and maintain the equipment. No matter how methodically and quickly the technician can locate a defect in the equipment, the final results will be unsatisfactory unless he has the necessary skill to repair the equipment in a workmanlike manner. Under proper supervision, the necessary skill can be obtained by the personnel who complete the following recommended Transistor Training Aid Program.

The training aid series in the EIB consists of five separate articles; (1) Phase I, keying oscillator (EIB 585); (2) Phase II, amplifier (EIB 586); (3) Phase III, AM radio receiver (EIB 587); (4) Phase IV, transceiver (EIB 592), and (5) Phase V, FM receiver (EIB 631). After satisfactory completion of all five phases of the training aid program, the person completing it should have acquired the experience needed to qualify him to make emergency repairs on equipment of modular or unitized construction.

The training aids to be built during this program will be duplicates of modular or unitized units in parts, material, and compactness, and their construction will involve most of the problems encountered in the repair of this type of equipment.

Additional procedures and techniques are also included in this training aid program to give participating personnel further insight into the principles employed in the design and fabrication of a printed circuit.

An additional source of information that will acquaint repair facility personnel and ship's data systems technicians with some of the more important techniques required for the repair and restoration of electronic assemblies is recent editions of the EIB's entitled "Repair and Maintenance Techniques for Electronic Assemblies". This series consists of eight articles, the first appearing in the 3 January 1961 (551A) issue. Article numbers two through eight are contained in the following EIBs-552, 553, 566, 567, 568, 569, and 570. It is recommended that these articles be read by all who are participating in the training aid program, and before attempting to repair electronic assemblies. These articles also serve as an educational guide for a better understanding of the techniques and procedures required in the design, mounting, and soldering of miniature and subminiature components onto plug-in board assemblies similar to those used in the training aid program.

PROTECTIVE DEVICES

Most protective devices are designed to interrupt the power to a circuit or unit when abnormal conditions such as short circuits, overlaods, high or low voltage, and excessive current occur. The most common types of protective devices are fuses, circuit breakers, and overload relays.

FUSES

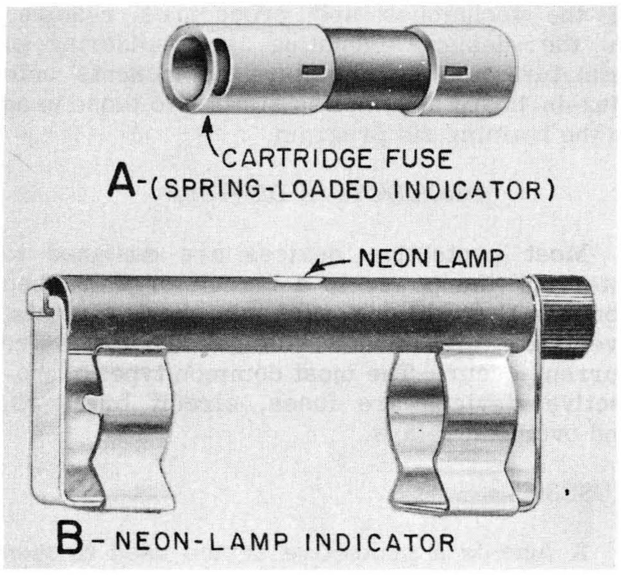
A fuse is a protective device used to open an electric circuit when the current flow exceeds a safe value. Fuses are made in many styles and sizes for different voltages and currents, but they all operate on the same general principle. Each fuse contains a soft metal link that melts and opens the circuit when overheated by excessive currents.

Plug Fuse

A plug fuse (the commercial-type home fuse) has a piece of zinc-alloy wire mounted in a procelain cup with a metal cover. A threaded contact base similar to a lamp socket is provided so that the fuse can be screwed into a socket in the fuse block. Plug fuses are used on small-capacity circuits ranging from 3 through 30 amperes at not more than 250 volts. Some plug fuses have small mica windows so that the fusible link can be observed. The plug fuse is seldom used in naval ships but is extensively used in commercial applications and shore stations.

Cartridge Fuse

A cartridge fuse (fig. 18-23) consists of a zinc alloy link enclosed in a fiber, plastic, ceramic, or glass cylinder. Some fiber and plastic fuse cylinders are filled with nonconducting powder. The smaller fuses are used up to 60 amperes and are made in the FERRULE, or round-end-cap type. Large sizes with short



77.95
Figure 18-23.—Cartridge fuse and blown-fuse indicators.

flat blades attached to the end caps are rated from 65 through 600 amperes. These blades fit tightly into clips on the fuse block similar to knife-switch clips.

Cartridge fuses are made in ratings of 1 through 1000 amperes for voltages of 125,250, 600, and 1000 volts. Fuses intended for 600-and 1000-volt service are longer and do not fit the same fuse holders as fuses intended for 250-volt service. Fuses of different ampere rating are also designed for different sizes of holders. For example, fuses of 1 through 30 amperes fit one size of holder, and fuses with capacities of 35 through 60 amperes fit a different size holder.

Before fuses of greater than 10 ampere rating are pulled, the switch for the circuit should be opened. Approved fuse pullers must be used for removing fuses. Fuses should never be short-circuited or replaced with fuses of larger current rating.

Time Delay

The time delay fuse is used for loads such as motor supply circuits in which overloads and motor-starting surges of short duration may be encountered. Common trade names for such fuses are Fusetron and Slo Blo. A conventional fuse of much higher rating would be required to prevent blowing of the fuse during surges. This rating would be too high to provide necessary protection for the normal steady state current of the circuit.

The time delay fuse is rated as to its time lag characteristic with a minimum blowing time at some overload current. A typical rating for this type fuse would specify "12 seconds minimum blowing time at 200 percent rated current."

Blown-Fuse Indicators

It is not always possible to detect a blown fuse by a visual examination. Hence, fuses are often equipped with a device that will provide a visual indication so that a blow-fuse condition can be readily detected (fig. 18-23). These devices consist of the spring-loaded and the neon-lamp types of blow-fuse indicators.

In the spring-loaded type (fig. 18-23A) when the link opens, it releases a spring that is held under tension. The action exposes an indicator, which makes the visual location of the blown fuse possible.

The neon-lamp type (fig. 18-23B) is designed to be mounted on the fuse. When the link opens, a neon lamp glows to show a blown fuse.

When no indicator is used, it is necessary to test the fuse continuity with a megger, ohmmeter, or voltmeter. Various methods of testing will be described later in this chapter.

Most fuse panels and switchboards are of the enclosed panel type. The term, "dead-front," means that all fuses and bus connections are enclosed in a metal cabinet when the cover is closed. The use of this type of construction reduces the possibility of property damage and danger to personnel. Modern switchboards are of the "dead-front" type.

However, the complete enclosure of the equipment makes it less accessible for test purposes. Therefore, most fuses used on "dead-front" switchboards have indicators that show when a fuse is blown. The fuse holder consists of a molded phenolic base, plug, and cap with a built-in indicator lamp (blown-fuse indicator). The lamp is usually a small neon bulb, which normally is shunted by the fuse element. When the fuse opens, the shunt is removed, causing an increase in the voltage across the neon lamp. The lamp then glows, indicating the open fuse.

Selection Of Proper Fuses

Separate fuses are provided on the I.C. switchboards for each associated circuit. A separate fuse is used in each line of each circuit. This has the effect of considerably increasing the maximum short circuit current that the fuses can safely interrupt. It also provides greater protection to the remaining circuits energized from the same bus in case of a possible defect in one fuse.

In general, fuse ratings should be approximately 10 percent above the maximum continuous connected load. In no case should the fuse rating be greater than two and one-half times the rated capacity of the smallest cable in the circuit. If too large a fuse were used, a fire hazard would exist.

CIRCUIT BREAKERS

Circuit breakers are used to provide circuit protection, to perform normal switching operations, and to isolate a defective circuit while repairs are being made. The types installed on naval ships are ACB, AQB,AQB-LF, NQB, ALB, and NLB.

ACB

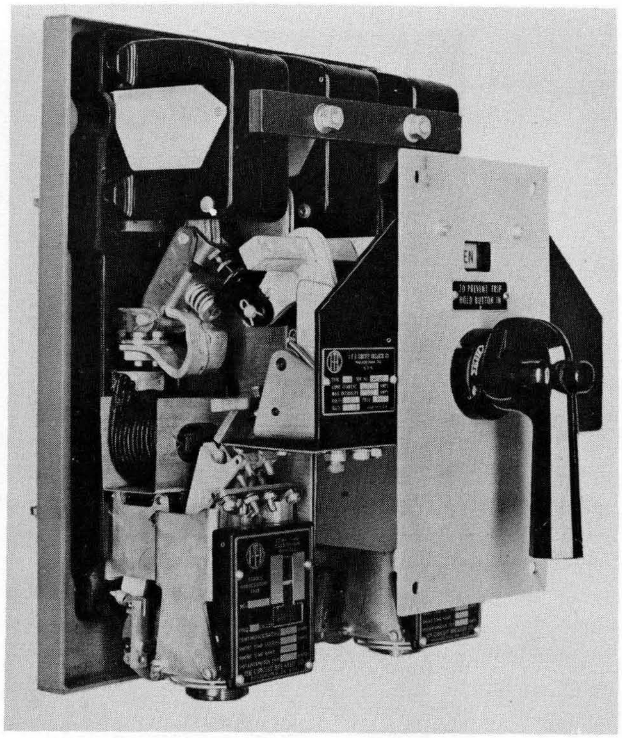
The ACB circuit breaker (fig. 18-24) may be used for either manual local closing or electrical remote closing. It has an open metallic frame mounted on a drawout mechanism and is normally used where there may be heavy load current, and where high short circuit currents exist.

When operated electrically, the operation is usually in conjunction with a pilot device such as a relay or switch. Electrically operated circuit breakers use an electromagnet, (solenoid), to trip a release mechanism that causes the breaker contacts to open. The energy to open the breaker is derived from a coiled spring. The electromagnet is controlled by the contacts in the pilot device.

Type ACB circuit breakers are used to connect ship's service and emergency generators to the power distribution system. They are also used on all feeder circuits from the main switchboard.

AQB

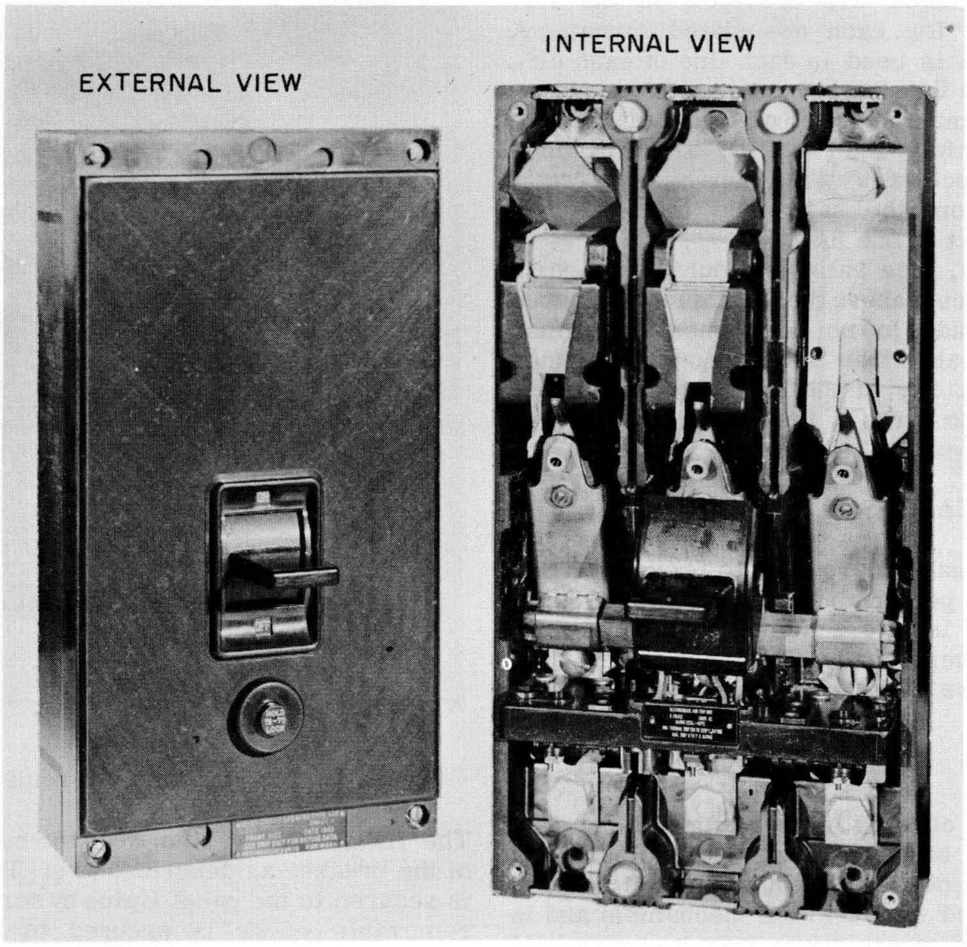
The AQB circuit breaker (fig. 18-25) is housed in an insulated enclosure, or frame.



27.73
Figure 18-24.—Type ACB circuit breaker.

The frame size varies with the current rating of the breaker as described later. The enclosure is secured to the panel frame by screws, and the removable cover is secured to the breaker housing by screws. Some circuit breakers have terminals at the front, and others at the rear. The rear terminals (not shown) are arranged to form a plug-and-socket connection to the bus assembly so that the whole circuit breaker can be removed for repairs. The trip device is a combined thermal and magnetic type of unit in which the thermal part operates on overloads and the magnetic part operates on short circuits. The tripping action allows momentary surges of current, such as those produced when induction motors are started. It protects apparatus from sustained overloads, and it acts instantaneously on short circuits. The tripping unit can be made inoperative by a hold-in button. When an overload trips the breaker, the handle moves to a point between the ON and OFF positions. The breaker is reset by moving the handle first to the OFF, and then to the ON position.

Circuit breakers of this type may have either two poles or three poles. Each pole is



27.74

Figure 18-25.—Type AQB circuit breaker.

provided with a trip unit, which, in case of excessive current, simultaneously trips all of the poles. These breakers are rated from 15 through 600 amperes in frame sizes of 100, 225, and 600. Trip units rated at 100 amperes and below operate with frame 100; units rated at 125 through 225 amperes operate with frame 225; and units rated at 250 through 600 amperes operate with frame 600. The AQB breaker is used extensively in distribution switchboards and load centers.

AQB-LF

The type AQB-LF circuit breakers (not shown) is similar in design to the AQB. High short-circuit current interrupting fuses have been incorporated on the circuit breaker in order that the breakers may be applied on circuits where the short circuit current may exceed the interrupting rating of the breaker.

NQB

Type NQB circuit breakers are mounted on insulated supports contained within molded insulation enclosures. They are similiar to the type AQB except that the type NQB has no automatic tripping devices. They are used for circuit isolation and manual transfer applications. They are not suitable for use with pilot devices.

Technically the NQB circuit breakers are used simply as ON-OFF switches.

ALB

Type ALB circuit breakers are designated low-voltage, automatic circuit breakers. The continuous duty rating ranges from 10 to 50 amperes at 125-volts a-c or d-c. The breaker is provided with a molded enclosure, draw-out type connectors, and nonremovable and non-adjustable thermal trip elements.

This circuit breaker is a quick-make, quick break type. If the operating handle is in the tripped position, indicating a short circuit or overload, the operating handle must be turned to the OFF position, which automatically resets the overload unit.

NLB

Circuit breakers type NLB are small and are used on low voltage systems (24 VDC, 120 VAC, and 120 VDC). They have no automatic tripping device and are used only as switches for circuit isolation.

Maintenanace

Circuit breakers require careful inspection and cleaning at least once a year (more frequently if subjected to unusually severe service conditions).

A special inspection should be carefully made of each pair of contacts after a circuit breaker has opened on a heavy short circuit. Before working on a circuit breaker, deenergize all control circuits to which it is connected.

Clean all surfaces of the circuit breaker mechanism, particularly the insulation surfaces, with a dry cloth or air hose. Before directing the air on the breaker, be certain that the water is blown out of the hose, that the air is dry, and that the pressure is not over 30 psi, Check the pins, bearings, latches, and all contact and mechanism springs for excessive wear or corrosion and evidence of overheating.

Slowly open and close breakers manually a few times to be certain that trip shafts, toggle linkages, latches, and all other mechanical parts operate freely and without binding.

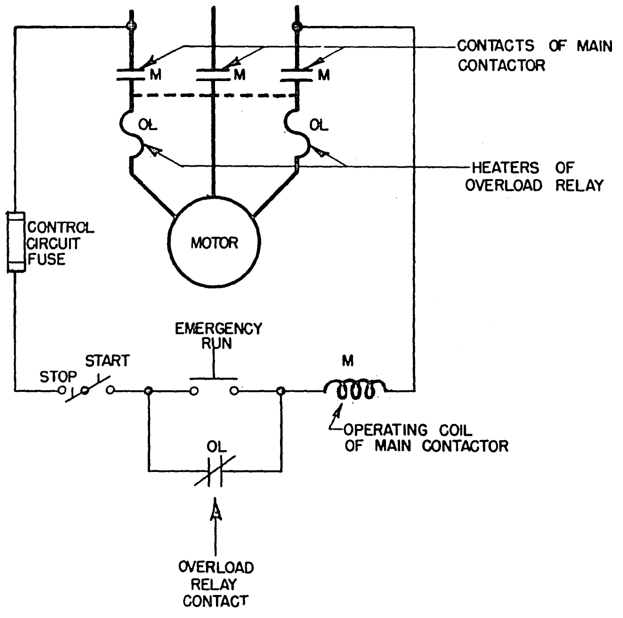
Operating tests that consist of operating the circuit breakers in the manner in which they are intended to function in service should be conducted regularly. For manually operated circuit breakers, simply open and close the breaker to check the mechanical operation. To check both the mechanical operation and the control wiring, electrically operated circuit breakers should be tested by means of the operating switch or control. Exercise care not to disrupt any electric power supply that is vital to the operation of the ship, or to endanger personnel by inadvertently starting motors and energizing equipment under repair.

OVERLOAD RELAYS

Overload relays are provided in motor controllers to protect the motor from excessive currents. Excessive motor current causes the normally closed overload relay contacts (fig. 18-26) to open, thereby breaking the circuit to the operating coil of the main contactor, which disconnects the motor from the line. Overload relays are of the thermal or magnetic (induction) type.

Thermal Type

The thermal type of overload relay has a heat-sensitive element and an overload heater connected in series with the motor circuit as shown in figure 18-26. When the motor current is excessive, heat from the heater causes the heat-sensitive element to open the overload


CONTACTS OF MAIN CONTACTOR
M
M
M
OL
OL
HEATERS OF OVERLOAD RELAY
CONTROL CIRCUIT FUSE
MOTOR
EMERGENCY RUN
START
STOP
M
OPERATING COIL OF MAIN CONTACTOR
OL
OVERLOAD RELAY CONTACT

124.249
Figure 18-26.—Schematic diagram of motor controller with thermal type overload.

relay contacts. As it takes time for the heat-sensitive element to heat up, the thermal type of overload relay has an inherent time delay. Thermal overload relays may be of the solder-pot, bimetal, single metal, or induction type.

SOLDER-POT TYPE.—The heat sensitive element is a strip of solder. Excessive motor current causes the solder to melt thus releasing a spring tension which opens the circuit to the main coil.

BIMETAL TYPE.—The heat-sensitive element is a strip or coil of two different metals fused together along one side. When heated, one metal expands more than the other causing the strip or coil to bend or deflect, and open the overload relay contacts.

SINGLE METAL TYPE.—The heat-sensitive element is a metal tube around the heater. The tube lengthens when heated and opens the overload relay contacts.

Induction Type

The heat-sensitive element is usually a bimetal strip or coil. The heater consists of a coil in the motor circuit and a copper tube inside the coil. The copper tube acts as a short circuited secondary of a transformer, and is heated by the current induced in it. This type of overload relay is used only in a-c controllers, whereas the previously described types of thermal overload relays may be used in a-c or d-c controllers.

MAGNETIC TYPE.—The magnetic type of overload relay has a coil connected in series with the motor circuit and a tripping armature or plunger. When the motor current is excessive, the armature opens the overload relay contacts. Magnetic overload relays may be of the instantaneous or time delay type.

INSTANTANEOUS TYPE.—This type operates instantaneously when the motor current becomes excessive. The relay must be set at a tripping current higher than the motor starting current to prevent tripping when the motor is started. This type of overload relay is used mostly for motors that are started on reduced voltage then switched to full line voltage after the motor comes up to speed.

TIME DELAY TYPE.—This type is essentially the same as the instantaneous type with the addition of a time delay device. The time delay device may be an oil dashpot with a piston attached to the tripping armature of the relay. The piston has a hole through which oil passes when the tripping armature is moved due to excessive motor current. The size of the hole can be adjusted to change the speed at which the piston moves for a given pull on the armature. For a given size hole, the larger the current, the faster the operation. This allows the motor to carry a small overload current for a longer period of time than a large overload current.

SAFETY

In the performance of your duties as a Data Systems Technician you will install, maintain, and repair electrical and electronic equipment in which dangerously high voltages are present. This work is often done in very limited spaces. Among the hazards of this work are electric shock, electrical fires, contamination by radioactive particles, harmful gases which are sometimes generated by faulty electrical and electronic devices, and injuries which may be caused by the improper use of tools.

Because of these dangers, you should be aware that the formation of safe and intelligent work habits is fully as important as your knowledge of electrical equipment. One of your primary objectives should be to train yourself to recognize and correct dangerous conditions and to avoid unsafe acts. You must also know authorized methods for dealing with fires of electrical origin, for treating burns and radioactive contamination, and for giving artifical respiration to persons suffering from electric shock. Some of these methods may be found in Basic Electricity, (NavPers 10086-A), others are included herein.

SAFETY PRECAUTIONS TO BE OBSERVED WHEN WORKING ON ENERGIZED ELECTRONIC EQUIPMENT

When military considerations require that electrical repairs be done on energized electronic equipment permission must be obtained from the commanding officer. The work should be done only by adequately supervised personnel fully cognizant of the dangers involved, and the following precautions observed.

1. Provide ample illumination.
2. The person doing the work should not wear wristwatch, rings, watch chain, metal articles, or loose clothing which might make accidental contact with live parts or which might accidentally catch and throw some part of his body into contact with live parts. Clothing and shoes should be as dry as possible.
3. Insulate the worker from ground by means of insulating material covering any adjacent grounded metal with which he might come in contact. Suitable insulating materials are dry wood, rubber mats, dry canvas, dry phenolic material, or even heavy dry paper in several thicknesses. Be sure that any such insulating material is dry, has no holes in it, and no conducting materials embedded in it. Cover sufficient areas so that adequate latitude is permitted for movement by the worker.
4. Cover working metal tools with insulating rubber tape (not friction tape) as far as practicable.
5. Insofar as practicable, provide insulating barriers between the work and any live metal parts immediately adjacent to the work to be done.
6. Use only one hand in accomplishing the work, if practicable.
7. A rubber glove should be worn on the hand not used for handling the insulated tools. If the work being done permits, rubber gloves should be worn on both hands.
8. Station men by circuit breakers or switches, and man a telephone if necessary, so that the circuit or switchboard can be deenergized immediately in case of emergency.
9. A man qualified in first aid for electrical shock should be immediately available while the work is being done.

RESIDUAL CHARGES IN DEENERGIZED EQUIPMENTS

Capacitors in high voltage circuits (or any component which passes the voltage-storing characteristics of a capacitor) retain voltage values which are potentially dangerous to personnel even when all source power is removed from the equipment which contains the component.

Before touching a capacitor, which is connected to a deenergized circuit (or which is disconnected entirely), short-circuit the terminals to make sure that the capacitor is completely discharged. A suitable insulated shorting stick should be used for this purpose (fig. 18-27).

The primary function of the shorting stick is to pass the discharge current from a capacitor through the ground wire to ground, NOT through the body of the person discharging the capacitor. The hook enables the technician to fasten the stick to the high-voltage terminal so that it can serve as an added protection while work is in progress. Connect the ground clamp to ground BEFORE using the hook.

Some shore stations have provided a shorting stick at each transmitter enclosure. In each case it is so placed that the technician must remove the shorting stick before he can gain access to the equipment.

No person should reach within or enter energized electronic equipment enclosures for the purpose of servicing or adjusting, except when prescribed by offical applicable technical manuals and then not without the immediate presence and assistance of another person capable of rendering aid in an emergency.

When the ship is in drydock, the electronic equipment on board may be energized only with the permission of the docking officer.

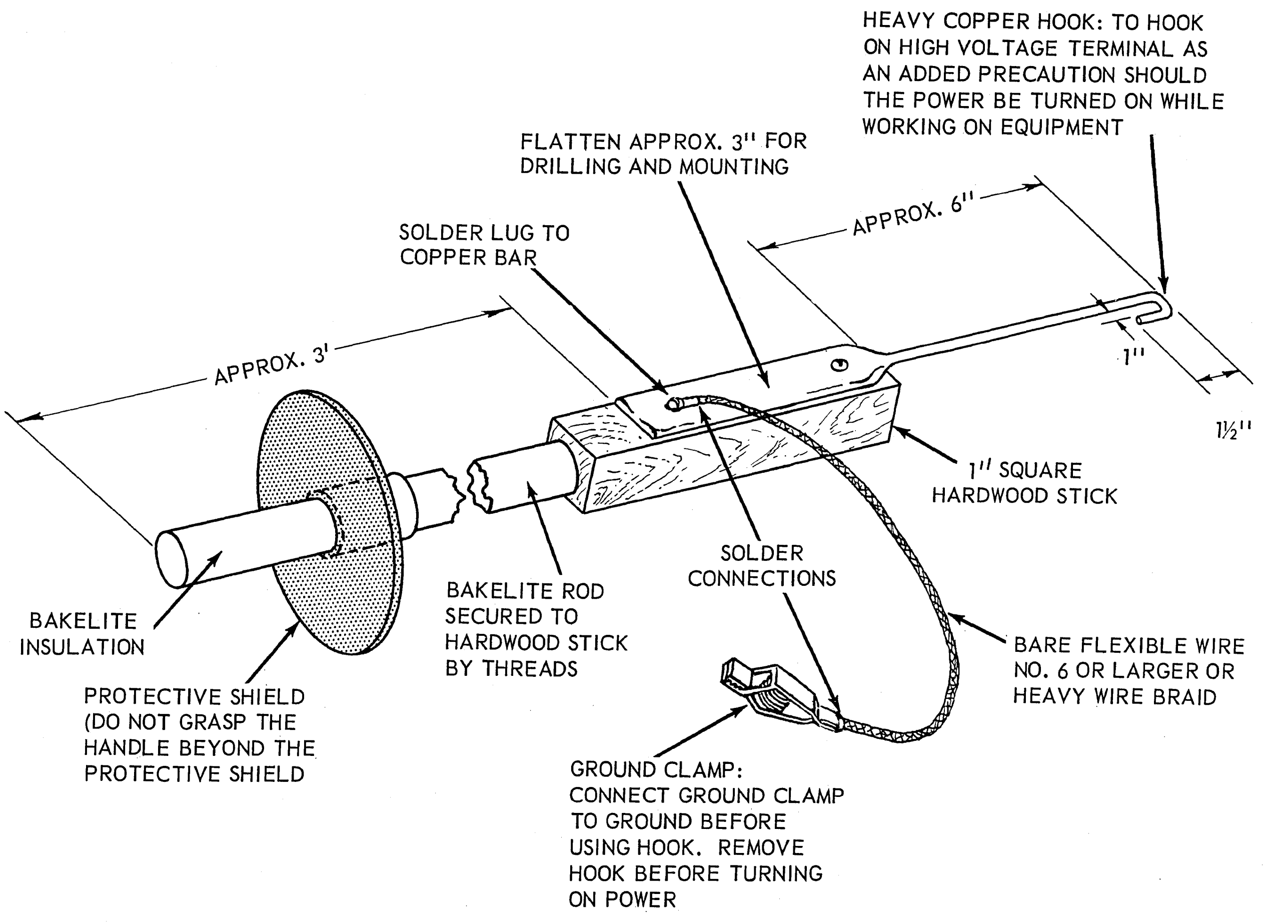
PRECAUTIONS WHEN HANDLING ELECTRON TUBES

The use of high-powered and special purpose electron tubes in radar and microwave equipments presently being used in conjunction with digital computing equipments makes it necessary for the DS to have a thorough knowledge of the operation and dangers involved when working with these tubes.

Cathode-Ray Tubes

The use of larger cathode-ray tubes has increased the danger of implosion, flying glass, and injury from high voltage. The danger is greatly reduced if the tubes are properly handled. If they are handled carelessly, struck, scratched or dropped, they can very well become an instrument of severe injury or death. The following precautions should be taken: (1) Goggles should be worn to protect the eyes from flying glass particles, (2) suitable gloves should be worn, and (3) no part of the body should be directly exposed to possible glass splinters caused by implosion of the tube. (The coating on some tubes is poisonous if absorbed into the blood stream.)

Cathode-ray tubes must not be unnecessarily exposed to possible damage. When a tube is being unpacked, remove it from the packing box


HEAVY COPPER HOOK: TO HOOK ON HIGH VOLTAGE TERMINAL AS AN ADDED PRECAUTION SHOULD THE POWER BE TURNED ON WHILE WORKING ON EQUIPMENT
FLATTEN APPROX. 3" FOR DRILLING AND MOUNTING
APPROX. 6"
SOLDER LUG TO COPPER BAR
APPROX. 3'
1"
1½"
1" SQUARE HARDWOOD STICK
SOLDER CONNECTIONS
BAKELITE ROD SECURED TO HARDWOOD STICK BY THREADS
BAKELITE INSULATION
BARE FLEXIBLE WIRE NO. 6 OR LARGER OR HEAVY WIRE BRAID
PROTECTIVE SHIELD (DO NOT GRASP THE HANDLE BEYOND THE PROTECTIVE SHIELD
GROUND CLAMP: CONNECT GROUND CLAMP TO GROUND BEFORE USING HOOK. REMOVE HOOK BEFORE TURNING ON POWER

1.1

Figure 18-27.—Diagram of shorting stick.

with caution, taking care not to strike or scratch the envelope. Insert it into the equipment socket cautiously, using only moderate pressure. When the tube must be set down, it is important that it be placed on a clean, soft padding. If special tube-handling equipment is available, it should be used according to instructions.

Radio Active Electron Tubes

Poisoning from radioactive materials contained in electron tubes such as radiac, spark gap, TR, glow lamp, and cold cathode tubes may be of 3 types:

1. ASSIMILATION—Eating, drinking, or breathing radium or radium compounds or absorbing them through cuts. Radium-bearing dust, which may be present in certain tubes, is dangerous in this respect.

2. BREATHING RADON—Radon is a tasteless, odorless, colorless gas that is given off by radium and radium compounds at all times. When breathed into the lungs it may cause severe injury.

3. RADIATION—Radium and radium compounds give off harmful, invisible radiations that can cause dangerous burns.

Useless unbroken electron tubes containing radioactive material are treated as any other radioactive waste material. Broken radioactive electron tubes are disposed of in accordance with BuShips Instruction 5100.5 of 28 November 1955. Additional instructions concerning handling, storage, and disposition of radioactive electron tubes are contained in BuShips letter S67/9-11 (871C), ESO Instruction 5100.1, and NavMed P-1325.

TROUBLESHOOTING LOGIC AND MEMORY CHASSIS

Servicing the computer consists mainly of removing and replacing defective card assemblies. Other repairs may consist of replacing defective wiring or components. Before attempting any repair, refer to Section 5 of the instruction manual for troubleshooting procedures and routines.

Since the card assemblies are classed as unrepairable items, no attempt is made to outline a procedure for component replacement on the card. When attempting any repairs, always make a thorough visual inspection of the chassis, card jacks, and associated wiring. When replacing blown fuses, determine the cause of the failure before installing a new fuse.

All of the card assemblies in the computer are color coded according to type number. A colored plastic tape (not shown) on the top edges of each card gives the last four digits of the card type. The color code used is the standard JAN resistor code. For example, a 250010 card is color coded black, black, brown, black. A 251010 card is color coded brown, black, brown, black. In this way a cross check can be made between the chassis maps, the functional schematics, and the computer chassis to ensure the correct placement of card assemblies.

LOGIC CHASSIS

When a failure occurs in any of the eight logic chassis, it is usually indicated by a typeout during a maintenance routine, or a failure of a programmed routine to run. Once the defective area of the computer is isolated, the defective circuit or circuits must be isolated. Use the diagrams in Section 9 of the instruction manual to locate the suspected circuits and their associated test points. Use a multimeter and/or an oscilloscope (such as the AN/USM-105A) to measure the voltage levels at these test points.

On all logic chassis, exclusive of the input/output lines, the voltage levels measured at the output of each logical circuit shall be 0 VDC for a "0" and -3 VDC for a "1". If the voltage measured at any test point does not correspond to the assigned levels, the card on which the circuit is located should be removed from the computer. A card known to be good is then substituted for the removed card. If time permits, the removed card can be checked in the AN/USM-142 (Module Test Set, discussed in chapter 16) to verify the failure. If the failure is verified the defective card should be destroyed.

If the voltage levels on the input/output lines (measured at the input to an input amplifier or the output from any control line or data line driver) do not correspond to the assigned levels, the card should be removed from the computer and checked on the test set, because, in this case, the external equipment could be at fault.

When a logic chassis power supply is to be checked for improper operation, refer to Section 2 of the instruction manual for the applicable voltage and resistance readings. Also refer to the pertinent schematic diagrams of the logic chassis power supply in Section 9.

MEMORY CHASSIS

When a failure occurs in any one of the five memory chassis, a typeout will occur during a maintenance routine or incorrect information will appear in registers during the running of a regular program. Once the defective area of the memory chassis has been isolated, the failing section or sections must be isolated. These sections are the drive line, inhibit, and sense. An oscilloscope should be used to check the waveforms for each section. When a waveform does not correspond to the waveform as shown in Section 6 of the instruction manual, remove the card on which the circuit is located. Check the card in the test set to verify the failure. Make replacements as necessary.

When a memory chassis power supply fails, refer to Section 2 of the instruction manual for the applicable voltage and resistance readings. Refer to Section 9 for the schematic diagram of the memory chassis power supply.

DEBUGGING A PROGRAM

After a coded program has been written out completely and the manuscript reviewed carefully, the program must be put on some input medium for loading into the computer. Once the program has been stored in the computer in binary form, either directly from the input medium or by means of some translation, the computer is ready to execute the program. If the program is lengthy, it is likely the coded program, as stored, has errors. Three common types of errors are:

1) Tape preparation errors made in the program for input.

2) Coding errors, such as listing incorrect or incomplete addresses, tranposition of digits in the y addresses or operation codes, and transcription errors.

3) Logical errors: incomplete or erroneous method used to obtain the solutions.

Debugging is the term applied to the process of locating errors in a program and correcting them. Debugging a program usually involves a series of trial runs of the program on the computer. Each time the program fails to run properly, the failure must be analyzed, and the error corrected. Frequently one can immediately discover the error, correct it manually from the console, and proceed with trial runs. At other times the detection of errors is more difficult and requires a thoughtful reconsideration of the program, keeping in mind any clues which may have been provided as to the origin of the error during the run.

The methods that can be used to debug a program vary depending on the nature of the programs, the service routines available, the availability of computer time, and personal preference of the individual for one procedure over another. The following discussion suggests possible patterns to follow in debugging.

PAPER TAPE PREPARATION ERRORS

This type of error can be minimized by systematic checking of all input tapes before their information is read into the computer. The usual method used to prepare input tapes is as follows:

Step 1. Use the typewriter or teletype equipment to punch tape from the coder's manuscript.

Step 2. Discard the resulting typed manuscript from the typewriter.

Step 3. Prepare a new typed manuscript from the punched paper tape.

Step 4. Compare the typed manuscript with the coder's manuscript to detect errors in punching.

Step 5. Correct all detected punching errors, and prepare a new tape.

Step 6. Obtain from the corrected punched tape a new manuscript which should be retained by the coder to use while debugging.

MANUAL DEBUGGING

Manual debugging is the process used to locate errors in a program by controlling the operations of the program from the console and visually checking these operations as they occur and indicate on the control panel. Although manual debugging does not make for efficient use of computer time, it can, if done correctly, reduce the overall time required to debug a program because versatility is afforded a programmer. In preparing a finished program that can be inserted into the computer in machine code, the following steps must be taken:

Step 1. Analyze the entire problem.

Step 2. Construct the flow diagram which shows the general computational steps to be taken.

Step 3. Create the program written in terms of computer instructions.

Step 4. If the program in Step 3 is coded in octal, the program is ready for execution after being put on the paper tape and loaded into the computer.

Step 5. Perform the debugging of the program by means of trial runs on the computer.

PREVENTIVE MAINTENANCE

Maintenance procedures used with the computer differ from the normal procedures used with other electronic equipment. Computer maintenance requires a logical analysis of a malfunction rather than an electronic analysis. It is essential that maintenance personnel become familiar with transmission paths within the computer. This is helpful in determining the failing area of the computer by taking note of the visual indications on the console when a malfunction occurs. It is also essential that maintenance personnel become familiar with the programs and routines being run in the computer. In this way, when a malfunction occurs, the correct maintenance routine (program) can be loaded into the computer to check the failing area.

The maintenance routines isolate malfunctions to a particular section or functional part of the computer. When a malfunction is isolated to a particular section, it can be further isolated to a printed circuit card or cards by using an oscilloscope and a multimeter.

A main executive routine is used to test performance characteristics of the computer circuits. The executive routine consists principally of subexecutive maintenance routines as follows: J CONTROL, K CONTROL, COMMAND, CONTROL, and ARITHMETIC.

Because of the extensiveness of these routines, and because they vary from one computer to another, maintenance personnel should refer to the appropriate instruction manual for this information.

Maintenance routines are used for preventive and corrective maintenance on the computer. When the routines have been loaded into the computer, the control routine (see the technical manual) takes command and runs all tests that check the various sections of the computer. Ideally all sections of the computer should be checked on a daily basis except the I/O sections which should be checked weekly.

When a preventive maintenance routine is being run, a malfunction occurring in the section of the computer being checked causes an error typeout. The explanation of the error typeouts accompanies the explanation of the specific routine being run.

The routines used for daily preventive maintenance are run at normal clock frequency, normal clock pulse width, and with normal bias voltages.

Marginal checking of memory must be done on a weekly basis. Marginal checking of each of the logic chassis must also be done on a weekly basis. The marginal checks are performed while running the daily preventive maintenance routines.

By using marginal checks, any cards or components deteriorating from age or usage can be detected before an actual failure occurs. This tends to reduce the actual downtime of the computer. When a malfunction occurs during marginal checking, the defective component should be replaced immediately. If time does not permit an immediate replacement of the defective part, operation of the computer can be attempted under normal operating conditions until time allows for the replacement of the defective part.

APPENDIX I

TRAINING FILM LIST

Certain training films that are directly related to the information presented in this training course are listed below under appropriate chapter numbers and titles. Unless otherwise specified, all films listed are black and white with sound, and are unclassified. For a description of these and other training films that may be of interest, see the United States Navy Film Catalog, NavPers 10000 (revised).

Chapter 2

INTRODUCTION TO COMPUTERS

MN8969A Digital Computer Techniques—Introduction. (20 min—1962—(Unclassified.)

Chapter 3

NUMBER SYSTEMS

MN8969B Digital Computer Techniques Computer Logic Part I Binary Numbers. (20 min—Color—1962.)

MN8969C Digital Computer Techniques Logic Part II—Logic Symbology.

Chapter 4

CONTROL UNIT

MN8969D Digital Computer Techniques—Computer Units. (24 min—Color—1962.)

MN8969E Digital Computer Techniques—Logic Element Circuits. (16 min—Color—1962.)

Chapter 5

ARITHMETIC UNIT

MN8969D Digital Computer Techniques—Computer Units. (24 min—Color—1962.)

MN8969E Digital Computer Techniques—Logic Element Circuits. (16 min—Color—1962.)

Chapter 6

MEMORY AND STORAGE UNITS

MN8969D Digital Computer Technique-Computer Units. (24 min—Color—1962.)

MA9820A Magnetic Cores—Part I—Properties. (29 min—1962.)

MA9820B Magnetic Cores—Part II—Basic Circuits. (30 min—1962.)

Chapter 7

INPUT/OUTPUT DEVICES

MN8969D Digital Computer Techniques—Computer Units. (24 min—Color—1962.)

Chapter 8

PROGRAMMING

MN 8969F Digital Computer Techniques—Programming. (14 min.—Color—Unclassified—1962.)

Chapter 9

ANALOG-DIGITAL AND DIGITAL-ANALOG CONVERSIONS

MN 8969A Digital Computer Techniques—Introduction. (20 min.—Color—Unclassified—1962.)

Chapter 10

NTDS COMPUTER CONTROL SECTION

MN 8969D Digital Computer Techniques—Computer Units. (24 min.—Color—Unclassified—1962.)

MN 8969F Digital computer Techniques—Programming. (14 min.—Color—Unclassified—1962.)

Chapter 11

NTDS COMPUTER ARITHMETIC SECTION

MN 8969B Digital Computer Techniques—Computer Logic—Part 1—Binary Numbers. (20 min.—Color—Unclassified—1962.)

MN 8969D Digital Computer Techniques—Computer Units. (24 min.—Color—Unclassified—1962.)

Chapter 12

NTDS COMPUTER MEMORY SECTION

MN-8969D Digital Computer Techniques—Computer Units. (24 min.—Color—Unclassified—1962.)

MN 9820A Magnetic Cores—Part I—Properties. (20 min.—Color—Unclassified—1962.)

MN 9820B Magnetic Cores—Part II—Basic Circuits. (20 min.—Color—Unclassified—1962.)

Chapter 13

NTDS COMPUTER INPUT/OUTPUT SECTION (PART I)

MN 8969D Digital Computer Techniques—Computer Units. (24 min.—Color—Unclassified—1962.)

MN 8969E Digital Computer Techniques—Logic Element Circuits. (16 min.—Color—Unclassified—1962.)

Chapter 14

NTDS COMPUTER, INPUT/OUTPUT SECTION (PART II)

MN 8969D Digital Computer Techniques—Computer Units. (24 min.—Color—Unclassified—1962.)

MN 8969E Digital Computer Techniques—Logic Element Circuits. (16 min.—Color—Unclassified—1962.)

Chapter 15

OTHER NAVY COMPUTERS

MN 8969A Digital Computer Techniques—Introduction. (20 min.—Color—Unclassifed—1962.)

MN 8969B Digital Computer Techniques—Computer Logic—Part I—Binary Numbers. (20 min.—Color—Unclassified—1962.)

MN 8969D Digital Computer Techniques—Computer Units. (24 min.—Color—Unclassified—1962.)

MN 8969F Digital Computer Techniques—Programming. (14 min.—Color—Unclassified—1962.)

Chapter 16

TEST EQUIPMENTS

MA-7812A	Circuit Testing with Meters and Multimeters—Theory. (35 min.—Sound 1951.)
MA-7812B	Circuit Testing with Meters and Multimeters—Practical Applications. (33 min.—Sound—1951.)
MN-8687B	Reading Multimeter Scales. (6 min—Sound—1956.)
MN-2104B	The Cathode Ray Oscilloscope. (23 min—Sound 1944.)
MN-2104A	The Cathode Ray Tube—How it Works. (15 min—Sound—1943.)

APPENDIX II

GLOSSARY

Special computer terms and their specific meanings as applied to this computer are given below.

ABORT—The condition in the computer that results in the skipping of the next sequential instruction.

ACCESS TIME—The time interval, characteristics of a memory or storage device, between the instant information is requested from memory and the instant the next request for information from memory can be made.

ACKNOWLEDGE—Indication of the status of data on the input/output lines. Abbreviated as ACK.

ADDRESS—A coded number that specifically designates a computer register or other internal storage location. Information is referenced by its address. Portions of computer control are responsible for directing information to or from an addressed location.

ADDRESSABLE—Capable of being referenced by an instruction; e.g., Enter A (f = 11).

ARITHMETIC—A section within the computer where reasonable processes such as addition, subtraction, multiplication, and division are performed, and operands and results are stored temporarily.

BINARY NUMBER SYSTEM—A number system with two symbols ("0" and "1") that has two as its base just as the decimal system uses ten symbols ("0, 1, ..., 9") and a base of ten. See also POSITIONAL NOTATION and RADIX.

BIT—A binary digit, zero or one, represented in the computer by the condition (set or clear) of a stage.

BIT PLANE—Two memory boards that contain the same relative bit position for each of 32,768 memory locations. Bit position is defined by the associated stage in the Z register. Bit plane control is 'concerned with the parallel transmission (flow) of information into and out of memory on a bit plane level.

BOOTSTRAP—A routine, normally input, contained in the 16-word wired memory.

BORROW—A borrow in subtraction is the additional subtraction of a one from the next partial difference and is initiated when a digit of the minuend is zero and the corresponding digit of the subtrahend is one. In a binary system of modules $2^k - 1$, where k is the number of stages in a register, the borrow produced from the leftmost digit $2^k - 1$ of the minuend is called the end-around borrow. A final correction is made by applying the end-around borrow to the partial difference of the rightmost digits.

BRANCH POINT—A point in a program or instruction where a decision is made on the basis of arithmetic results. The result of the decision indicates whether the main program is to be continued or branched to a different program. See also JUMP.

BUFFER—Mode of operation that involves inter-equipment data transfer; the final holding register on the computers output i.e., the C registers.

CAPACITY—The upper and lower limits of the numbers that may be processed in a computer register.

CENTER BOARD—The printed circuit board that distributes the outputs of the secondary windings of the drive current (section) transformers to the X and Y drive lines on the memory boards. One printed circuit connects eight drive lines to one secondary winding of one selector transformer.

CLEAR (verb)—To restore a storage or memory device to the zero state.

CODED PROGRAM—A procedure for solving a problem by means of a digital computer. The program may vary in detail from a mere outline of the procedure to an explicit list of instructions coded in machine language. See also PROGRAM.

COLUMN—In positional notation, a position corresponding to a given power of the radix. A digit located in any particular column is a coefficient of a corresponding power of the radix.

COMMAND—One of a set of signals or groups of signals resulting from an instruction. Commands initiate the individual steps of the instruction. See also INSTRUCTION.

COMPLEMENT—A number that meets conditions (1) or (2) as follows: (1) True complement: a number (n) with "d" digits which, when added to a starting number(s), produces a sum having all zeros in the "d" digit columns and a one in the d + 1 digit column. Method: to find the complement, subtract one from the radix, then subtract each digit from this difference, and then add one to the least significant digit, executing necessary carries. (2) Radix minus 1's complement: a number (n) with "d" digits which, when added to a starting number(s), produces a sum having radix less one in each of the "d" digit columns. Method: to find the complement, subtract each digit from the radix less one. (3) The one, when added to the radix minus 1's complement, produces the true complement.

CONTROL—The computer circuits that affect the carrying out of instructions in the proper sequence, the interpretation of each instruction, and the application of the proper commands to other sections and circuits in accordance with the interpretation.

COUNTER—A device capable of increasing or decreasing its own contents upon receipt of separate input signals.

CORE MATRIX—An array of cores, each of which represents the same column for each storage register in the magnetic core storage system.

CORE STORAGE—A type of storage system in which the magnetic core is the basic memory element.

DEBUG—To isolate and remove all malfunctions from the computer, or all mistakes from a routine or program.

DIGIT—One of a set of characters used as coefficients of powers of the radix in the positional notation of numbers.

DUMP—Transfer of information from one piece of equipment to another (normally from computer to external equipment such as paper tape, high-speed printer, etc.).

ENABLE—A "0" signal allowing other conditions to be sampled. (Verb) - The application of "0".

END-BOARD ASSEMBLY—The assembly of eight printed circuit diode boards, and the included diodes, mounted on each end of the memory stack. The end-board assembly connects each line selector to eight drive lines and isolates the seven lines that are not driven from the drive line.

FAULT—The condition resulting from the execution of an improper instruction, as follows: (1) Illegal function code (00, 77) jump to fault entrance address (00000) and light FAULT light. (2) Divide instruction (f = 23) with the divisor equal to zero, or inappropriate size numbers in quotient overflow.

FLOW DIAGRAM—A graphical representation of a sequence of operation.

FUNCTION CODE—The portion of the instruction word (bits 2^{29} - 2^{24}) that specifies to the control section the particular instruction which is to be performed.

HALF-SUBTRACT—The bit-by-bit subtraction of two binary numbers with no regard for borrows. Abbreviated as HS. The complement of the half-subtract is "half-subtract not", abbreviated as $\overline{\text{HS}}$.

INSTRUCTION WORD—Designators specifying a particular function. The designators are listed below as they appear from highest to lowest order significance in the instruction word:

f	=	Function Code (6 bits)
j	=	Branch Condition (3 bits)
k	=	Operand Interpretation (3 bits)
b	=	Address Modification (3 bits)
y	=	Operand Address (15 bits)

INPUT/OUTPUT—A section providing the means of communication between the computer and external equipment or other computers. Input and output operations involve units of external equipment, certain registers in the computer, and portions of the computer control section. Abbreviated as I/O.

INTERRUPT—(1) Internal: indicates the termination of an input or output buffer; (2) External: signal on the data lines that requires computer attention.

JUMP—An instruction that specifies the location of the next instruction and directs the computer. A jump is used to alter the normal sequence control of the computer. Under certain conditions, a jump may be contingent upon manual intervention.

LINE SELECTOR—The part of the address selection circuit that includes the end boards and provides circuit continuity for one of the eight drive lines selected by the primary and secondary selectors.

LOAD—To enter information into either the computer or a storage location.

LOGICAL SUM—The bit-by-bit addition of two binary numbers with no regard for carries. Abbreviated as LS. The complement of the logical sum is "logical sum not", abbreviated as $\overline{\text{LS}}$.

LOOP—Repetition of a group of instructions in a routine.

MALFUNCTION—Nonoperation of the computer due to component failure.

MARGIN—A measure of the tolerance of a circuit; the range between an established operating point and the point at which the circuit first starts to fail.

MASTER CLOCK—The primary source of timing signals.

MEMORY—Any device into which information can be introduced, stored, and then extracted at a later time. See also STORAGE.

MEMORY BOARD—A mechanical assembly containing a 128 x 128 (16,384) array of cores and their associated sense windings, inhibit windings, and drive lines.

MEMORY STACK—A memory stack is comprised of 12 memory boards, two end boards, and a center board. A memory stack is divided into two half-stacks, referred to as the front stack (on the card side of the memory chassis) and the back stack (on the wiring side of the memory chassis) or, stack 0 and stack 1, respectively. Each half-stack is comprised of six memory boards and an end board and shares the common center board.

MODULUS—The number of permissible numbers used in a process or system. For example, if only the integers from -15 to +15 inclusive are considered, 31 is the modulus of this set of numbers.

MOST SIGNIFICANT DIGIT—The first digit from the left, different from zero.

NONVOLATILE STORAGE—Storage media that retains information during the absence of power. These media include magnetic tapes, drums, and magnetic cores.

OCTAL NUMBERS—Numbers in a system using eight symbols, 0, 1, 2, ..., 6, 7, with eight as its base.

OPERAND—Coded data representing a word or number that is involved in computer operations or results from computer operations.

OPERATION—(1) The activity resulting from an instruction. (2) The execution of a set of commands.

OVERFLOW—(1) The condition that arises when the result of an arithmetic operation exceeds the capacity of the number representation in the computer. (2) The condition that results during a real-time problem when the lower half of the Z register contains all "1's" (approximately 32 seconds).

PARALLEL TRANSMISSION—The system of information transfer in which the characters of a word are transmitted simultaneously over separate lines.

PARTIAL CARRY—A system of executing the carry process in which the carries that arise as a result of a carry are not allowed to be transmitted to the next higher stage.

POSITIONAL NOTATION—One of the schemes for representing real numbers, characterized by the arrangement of the digits in sequence. The successive digits are interpreted as the coefficient of successive integral powers of a number called the radix of the notation.

PRIMARY SELECTOR—That part of the address selection circuit that is comprised of a selector transformer and the control circuits associated with its primary windings.

PROGRAM—A sequence of coded computer instructions and necessary operands for the solution of a problem. See also CODED PROGRAM.

QUADRANT—One-fourth of a memory board. A quadrant is comprised of a 64 x 64 (4,096) array of cores, a sense winding, an inhibit winding, and associated X and Y drive lines.

RADIX—The number of individual characters used in a number system. Decimal uses 10 characters (radix 10), 0 through 9, octal uses eight characters (radix 8), 0 through 7, and binary uses two characters (radix 2) "0" and "1".

RADIX POINT—The index that separates the digits associated with negative powers from those associated with the zero and positive powers of the radix of the number system in which a quantity is represented; e.g., binary point, decimal point.

READ—To extract information.

READY—The condition existing between two computers where the computer initiating the ready signal has data on the output lines. This signal is further interpreted as an Input Data Request signal by the receiving computer.

REAL-TIME—Computer operation with regard to a specific time or event.

REAL-TIME CLOCK—The device by which actual elapsed time is measured in seconds or fractions thereof. Abbreviated as RTC.

REGISTER—A quantity of stages. The regulation number of stages determines the modulus of the number system allowable to representation by the computer system. The nature of a register determines its use as a device for information storage. Frequently other storage devices for information are also referred to as registers.

RESUME—The signal generated when the receiving computer has sampled its input lines and has generated an Input Acknowledge which is interpreted as the Resume signal by the transmitting computer.

ROUTINE—A sequence of operations a digital computer may perform, or the sequence of instructions which determine these operations.

SECONDARY SELECTOR—The part of the address selection circuit that includes the secondary windings of the selector transformers, their associated control circuits, and the printed circuit wiring on the center board. As determined by the secondary selector translator, the selected secondary winding furnishes the drive current to the center board for distribution to eight drive lines.

SELECTOR TRANSFORMER—A transformer, also called the read/write or drive current transformer, used in the address selection circuit. It has two primary windings (one read winding and one write winding) and four secondary windings. All windings have an equal number of turns.

SERIAL TRANSMISSION—A system of information transmission in which the characters of a word are transmitted in sequence over a single line, in contrast to parallel transmission.

SET (verb)—To change the state of a stage from zero to one.

SHIFT—Displacement of an ordered set of characters one or more columns to the right or left. In the case in which the characters are the digits of a number, as in a fixed point digital computer, a left shift is equivalent ordinarily to multiplication by a power of the radix.

SIGN DIGIT—A character used to designate the algebraic sign of a number.

SIGNIFICANT DIGITS—Digits to the right of the most significant digits.

SINGLE-ADDRESS (INSTRUCTION) CODE—In general, an instruction consists of a coded representation of the operation to be performed and of one or more addresses of words in storage. The instructions of a single-address code contains only one address.

STAGE—An electronic device that periodically may be in unique states (conditions). The number of possible states in the device determines the radix of the number system allowable to representation by the computer. It is possible for a stage in the computer to have two unique transient states: (1) a state assigned to represent zero, (2) a state assigned to represent one.

STORAGE—A device in which information is set aside, or stored, for immediate or future use. See also MEMORY.

VOLATILE STORAGE—Opposite of nonvolatile storage; information is lost in the event of a power interruption.

WORD—Information coded for computer representation as a series of bits. The normal word length is 30 bits in the computer (15-bit length is optional).

WRITE, STORE, or REPLACE (verb)—To introduce information into some form of storage.

INDEX