NAVAL TACTICAL DATA SYSTEM

PROGRAMMERS' GUIDE for COMPUTER AND EQUIPMENT

PX 1493

VOLUME

ONE

Remington Rand Univac

DIVISION OF SPERRY RAND CORPORATION

UNIVAC PARK, ST. PAUL 16, MINNESOTA

NAVY DEPARTMENT | BUREAU OF SHIPS | ELECTRONICS DIVISIONS

CONTRACT: NObsr 72769 | NTDS NO. U-6464 | 1 MARCH 1961

INTRODUCTION

NTDS PROGRAMMERS' GUIDE

INTRODUCTION

This publication is intended as a guide to be followed by programmers and as a reference for purposes of information and familiarization. Standard programming conventions, rules and restrictions are presented for the programmer for use in the preparation of programs and the operation of the NTDS computing centers. A general treatment of the individual pieces of peripheral and auxiliary equipment supplies the information necessary for an understanding of system operation. This information is supplemented with a section of charts to be used as a rapid and easy reference.

This publication is open-ended so that the latest releases of applicable information may be included as they become available.

SECTION A

PROGRAMMING RULES AND CONVENTIONS

SECTION A1

NTDS FLOW DIAGRAMMING RULES

NTDS FLOW DIAGRAMMING RULES

1. GENERAL

These rules are intended to serve as a guide to people preparing original drafts of NTDS Service Test flow diagrams. In order that the information on these diagrams can be distributed with the required promptness and that this information be retrievable by the maximum number of interested people, it is essential that the original drafts be prepared with the utmost uniformity consistent with clarity. Aside from the uniform use of uniform symbols and a uniform page format, it is essential that the utmost uniformity be preserved in the *style* of presentation.

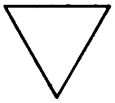
A flow diagram is a graphic representation of the sequence of logical steps used to perform some function. The amount of detail shown is determined by the eventual use for which the diagram is being prepared. Detail should be the minimum consistent with an accurate representation of the given function - the breakdown should not follow any particular scheme of implementation (e.g., machine-oriented references). A flow diagram can consist only of straight lines, branches, and loops; in the interest of clarity, these should look like straight lines, branches, and loops. In general, the originator's draft of a flow diagram should be a clear, complete, accurate, uncluttered, graphical representation of the given function; the originator is the person best qualified to know when this is the case.

Since copies of a flow diagram must sometimes be distributed prior to the execution of final artwork, it has become necessary that the originator's draft be in ozalid-reproducible form. It could be in pencil, but it must be on vellum and capable of producing *readable* ozalid prints. A supply of suitable vellum with nonreproducible cross rules is on hand for use in preparing original drafts. Standard templates are also available.

2. PAGE FORMAT

A flow diagram should occupy a single sheet - 8-1/2 inches wide and 11 inches high with approximately a one-inch margin all around. If it will not fit on a single sheet without crowding, the diagram may be split and continued as one or more sheets, using appropriately keyed discontinuity symbols. Separate amplifying notes should not accompany the symbols unless

their omission would detract from the clarity of the diagram. Figure A1-1 is an example of flow-diagram format.



An equilateral triangle, point down 1/2 inch high as shown, indicates *entrance* and exit points of the sequence.

Lines connecting successive symbols indicate sequence of flow. A closed arrowhead signifying an ***input*** terminates each line.

An open circle, 1/4 inch in diameter represents a junction point - the use of this symbol is required when more than one input to one of the symbols must be shown. Where one of the inputs involves a discontinuity the following symbol should be substituted.

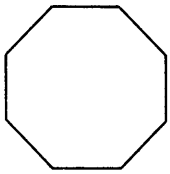
A circle, 1/4 inch diameter represents a discontinuity, it is used to enhance the clarity of a diagram when it is impossible or undesirable to indicate flow by means of a continuous line. The printed symbol will be a reversed bold-face letter corresponding to the circle with which it is to be connected. For rough drafts, normal lettering will be used. In addition to the obvious case where the diagram is split and appears on more than one sheet, this symbol should be used where a continuous line would have to cross another line in the diagram or make more than two 90 degree bends. The symbol should also be used where there are more than three inputs to a single junction point.

Each of the following symbols must contain a concise statement of the operation represented. The first word of the statement should be capitalized and there should be no final punctuation. The symbols should be large enough to contain the statement without crowding. If any explanatory note is necessary, it should appear within the symbol to which it pertains. However, if a note is excessively long (say over 10 words) or applies to more than one symbol it should be suitably referenced and placed below the diagram in the form of a foot note.

Represents an operation (within the sequence) that does not require a decision or selection.

Represents an operation which exists as an entity independent of the sequence, that is, a subroutine. The subroutine will be indicated by the CS-1 code.

Represents an operation in which a decision or a selection is made. The statement within the symbol, although normally in the form of a question, is not punctuated. Preferably the question should be in such form as to require a YES or NO answer. Where there are two alternative outputs, indicate this by two flow lines 90 degrees apart. Where there are more than two alternatives, only *one* flow line should leave the symbol - this flow line in turn should be split up into the required number of branches. The answer to the question (or result of the selection) should appear above the appropriate branch.



Represents a *stop* in the sequence flow. Any required external notification, associated condition, and/or action should be noted in the octagon when its omission would detract from the clarity of the diagram.

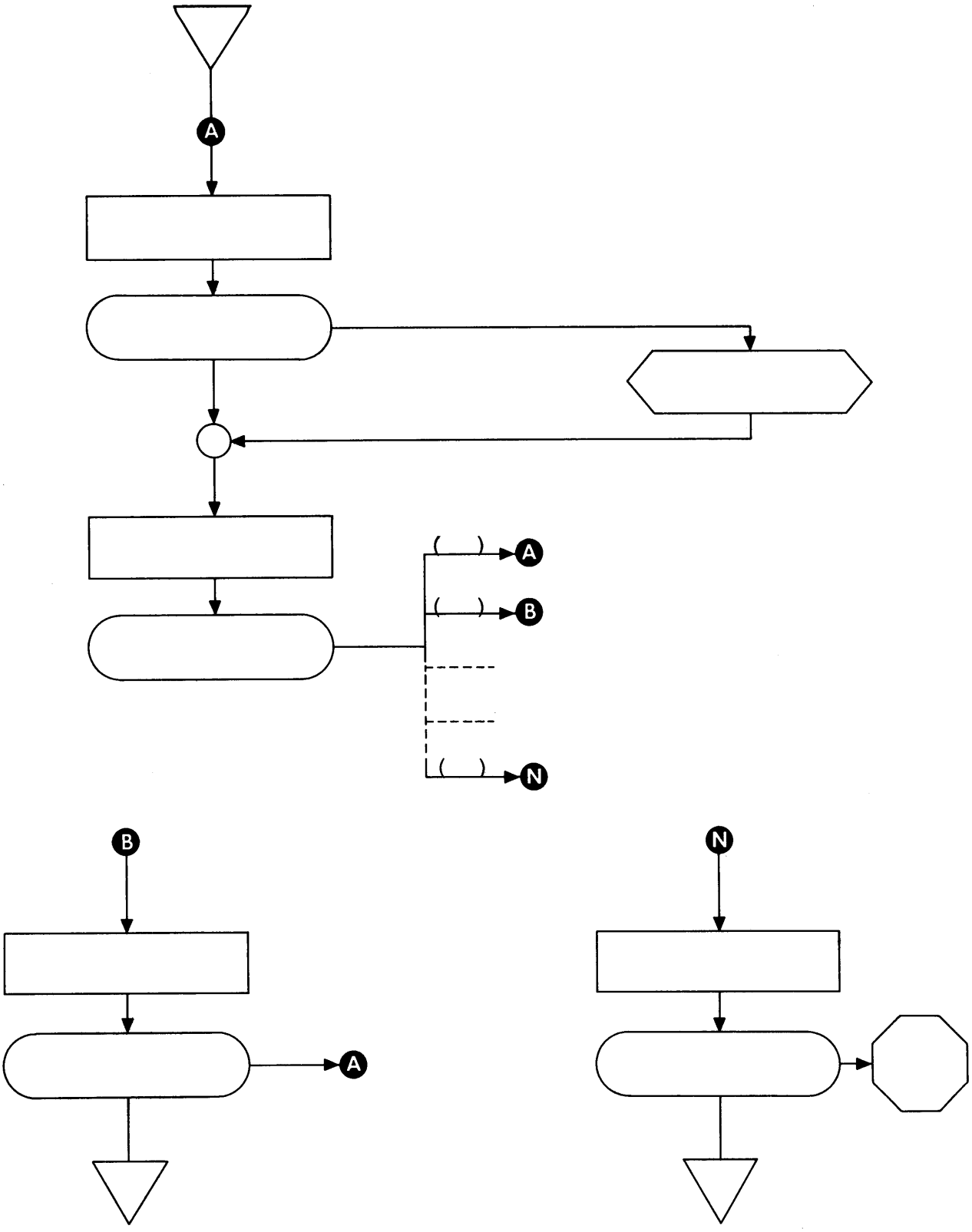


Figure A1-1. Flow Diagram Format

SECTION A2

NTDS PROGRAMMING RULES

NTDS PROGRAMMING RULES

- All programs and routines will be written in CS-1 language. Any level of expression acceptable to the compiling system may be used.
- In any subroutine, the base label applies to the first program address and should not end with a numeral. All subsequent labels of a given subroutine will consist of this base label followed by a suitable numeral to provide uniqueness.
- A PROGRAM-header label and the base label should be identical.
- No absolute addresses (except special-purpose addresses) shall be used in *any* program.
- References to initial entry points shall *not* use a tag ± increment in the Y-address portion of the instruction.
- All input/output routines should make use of MEANS or CHAN-SET operations. This permits coding of input/output operations.
- B-register 7 and the A and Q registers are to be considered transient.
- B registers shall not be used as a means of decrementing an index through zero, that is, for decrements of 1, use 72 200 NEXT INSTRUCTION instead of 12 202 77776; and for decrements greater than 1, use

$$\left\{\begin{matrix} 11\ 042 - n \\ 12\ 270\ 00000 \end{matrix}\right\}$$

  The CS-1 operator INCREMENT automatically generates the above.
- Key jumps should not appear in any operational program.
- All programs will be written as subroutines or procedures.
- The following items apply to any subroutine not written as a Procedure Statement:
  a) All subroutines will have 0•0 as the first instruction and ENTRY operator should be used.

b) Each subroutine will have only one entrance.

c) All subroutines will have exits which consist of JP•L (Entry label). The Entry label will be the base label of the subroutine; one or more EXIT operators should be used to prepare such exits.

d) Subroutines must not modify a jump instruction.

- All programs shall be written in such form that they are preset, not reset; that is, each entry should be independent of previous executions of the program.
- All external references from any subroutine will be to base labels only.
- CS-1 Compiling System operators shall be used when selecting, disabling, or enabling peripheral equipment.
- The interrupt routine shall store and re-store the contents of all operational registers used.
- A common load and dump technique will be used by all programs. Computer centers will use the NTDS Utility System routines.
- All external references from any procedure will be made with the referenced procedure's name as the operator of a procedure-linking operation.
- The procedure-linking operation assigns all input/output parameters to a linked procedure and specifies all abnormal exits.
- All system procedure packages are to use the name of the primary procedure as the library label for cataloging.
- Routines which retrieve data from a CS-1 Librarian's User Library must begin with a SYSTEM header.

SECTION A3

NTDS ABBREVIATIONS

NTDS ABBREVIATIONS

ABS	Absolute Machine Code Format
AC	Air Control
A/C	Aircraft
AD	Air Defense
A/D	Analog to Digital
ADR	Address
AEW	Airbone Early Warning
AGC	Automatic Gain Control
AI	Airborne Intercept
ALGOL	Algorithmic Language (Compiler)
ALLOC	Allocation
ALTE	A-Link Terminal Equipment
AN/USQ	A - Army N - Navy U - General Utility S - Special Q - Combination
AN/USQ-17	NTDS R&D Unit Computer
AN/USQ-20	NTDS Service Test Unit Computer
A/S	Antisubmarine
ASDEC	Applied Systems Development Evaluation
ASM	Air-to-Surface Missile
ASP	Antisubmarine Patrol
ASW	Antisubmarine Warfare
ATDS	Airborne Tactical Data System

AW	Air Warning
BI-OCT	Bioctal Format
BAM	Binary Angular Measure of Revolutions
BB	Battleship
BCD	Binary Coded Decimal
BTL	Bell Telephone Labs
BRG	Bearing
CS-1	NTDS Compiling System No. 1
CA	Heavy Cruiser
CAG	Heavy Cruiser, Missile
CAP	Combat Air Patrol
CCA	Carrier Controlled Approach
CDG	Central Display Generator
CIC	Combat Information Centers
CK	Check
CL	Light Cruiser
CLG	Light Cruiser, Missile
CM	Countermeasure
CN	Call Number
CNO	Chief of Naval Operations
COBOL	Common Business Oriented Language (Compiler)
C-CONTROL	CS-1 Compiler Control Operations
COP	Computer Oriented Programming (CS-1)
CPA	Closest Point of Approach
C/R or CR	Carriage Return
CTRL	Control
CTS	Central Track Store

CV	Carrier
CVA	Attack Carrier
CVE	Escort Aircraft Carrier
CVL	Light Aircraft Carrier
DASH	Destroyer Antisubmarine Helicopter
DCP	Data Control Panel
DD	Destroyer
DDE	Destroyer Escort
DDG	Guided Missile Destroyer
DDR	Radar Picket Destroyer
DEC or D	Decimal
DER	Destroyer Escort Radar Picket
DEW	Distant Early Warning
DF	Direction Finder
D/F	Direction Finding
DL	Destroyer Leader
DLG	Guided Missile Destroyer (Light Frigate)
DR	Dead Reckoning
DRM	Direction of Relative Motion
DSU	Data Store Unit
D/T	Detection and Tracking
ECM	Electronic Countermeasures
EDS	Electronic Data System
ESR	Effective Sonar Range
ETA	Estimated Time of Arrival
ETD	Estimated Time of Departure
ETS	NTDS Engineering Test System

E/U	Engaged/Unengaged
EX-FCT	External Function (Equipment Command Code)
FAAWTC	Fleet Anti-Air Warfare Training Center
FAD	Fighter Air Director
FBM	Fleet Ballistic Missile
FC	Fleet Center
FC	1) Function Code (Utility System) 2) Univac File Computer
FCU	Format Control Unit
FCS	Fire Control System
FF	Fine Frequency
FLD	Field
FLEX	Flexowriter
FP	Floating Point
FPC	Fleet Programming Center
GCI	Ground Controlled Interceptor
HS	A/S Helicopter
HSI	High-Speed Intercept
HUK	Hunter/Killer
HSPu	High-Speed Punch
HSPr	High-Speed Printer
IAS	Indicated Air Speed
ID	Identifier
IDAC	Interconnecting Digital and Analog Converter
IFF	Identification Friend Foe
ITSA	Interim Tape System Adapter
KT	Knot

KSC	Keyset Central
L_0	The CS-1 input language level consisting basically of English words and abbreviations. Expressions use the English alphabet and numeric and algebraic terms in which the problem is defined.
L_1	An intermediate CS-1 language level in which L_0 language is converted into a standardized CS-1 code and each statement is identified and assigned to a specific location, and item, within the compiler's storage.
L_2	An intermediate CS-1 language level in which L_1 language has been translated into machine code (with the exception of the address structure).
L_3	An intermediate CS-1 language level consisting of the final machine code stored within the compiler.
L_4	A CS-1 language level consisting of the object program after it has been produced as output.
L_1ID	CS-1 L_1 Operation Identification Number
LRM	Long-Range Missile
LSB	Lower Sideband
LSD	Least Significant Digit
LTS	Local Track Store
MAD	Magnetic Airborne Detection
MAG TAPE	Magnetic Tape
MCM	Monitor Control Message
MER	Maximum Effective Range
MIFI	Missile In Flight
MRM	Monitor Reply Message
MTU	Magnetic Tape Unit
NELIAC	Naval Electronic Laboratory International Algebraic Compiler
NObsr	Naval Order Bureau of Ships Radio (BuShips Contract)

NTDS	Naval Tactical Data System
NTDSUC	Naval Tactical Data System Unit Computer
NWP	Naval Warfare Publications
OCT	Octal
OPCON	Operational Control Center
OPTEVFOR	Operational Training and Evaluation Force
OTC	1) Officer in Tactical Command 2) Officer in Tactical Communications
PCR	Program Checkout Routine
PE	Peripheral Equipment
PER	Photoelectric Reader
PKG	Package
POP	Problem Oriented Programming (CS-1)
PPI	Plan-Position Indicator
PRGM or PROG	Program
PRR	Pulse Repetition Rate
PTCP	Paper-Tape Control Program
PW	Pulse Width
R-3	An Absolescent NTDS Assembly System
RAF/USN/RN	Royal Air Force/United States Navy/Royal Navy
RCP	Resident Control Program
RDF	Radio Direction Finding
REL	Relative Load Format
RF	Radio Frequency
RHI	Range Height Indicator
RTB	Return-to-Base
SAGE	Semiautomatic Ground Environment

SAI	1) Size Analysis Indicator 2) Storage A-Scope Indicator
SF	Scale Factor
SIF	Selective Identify Feature
SINS	Ships Inertial Navigation System
SMP	System Monitoring Panel
S/R	Subroutine
SSB	Single Sideband
SSR	Subroutine Radar Picket
STS	1) Service Test System 2) System Track Store
SYS-DD	System Data Designs
SYS-PROC	System Procedure
TAB	Tabulator
TAC	Tactical Air Commander
TACC	Tactical Air Control Center
TACAN	Tactical Air Navigation Equipment
TADC	Tactical Air Detection Center
TAS	True Airspeed
TASC	Tracking and Sorting Computer
TBL	Table
TCC	Tactical Control Console
TCS	Table Control System
TDB	Table Design Base
TDC	Tactical Display Console
TDD	Target Detecting Device
TDS	Tactical Data System

TDX	Target Data Exchange
TEC	Threat Evaluation Console
TEL	Terminal Equipment Logic
TEWA	Threat Evaluation and Weapons Assignment
TH	True Heading
TIDE	Tactical International Data Exchange
TQ	Track Quality
TSC	Tracking Supervisor Console
TTY	Teletype
TWS	Track While Scan
UNIVAC	Universal Automatic Computer
USB	Upper Sideband
UTLP	Utility Tape Load Program
UTM	Universal Test Message
UTUP	Utility Tape Update Program
USNMI	US Naval Maneuvering Instructions
VA	Attack Squadron
VF	Fighter Squadron
VP	Video Processor
W/Arm	With Armament
WCC	Weapons Control Center
WDE	Weapons Direction Equipment
WDS	Weapons Data System
WSEG	Weapons System Evaluation Group
XS3	Excess Three Code

SECTION B

NTDS COMPUTING CENTERS

SECTION B1

SAN DIEGO OPERATIONS

SAN DIEGO OPERATIONS

1. SERVICES AVAILABLE

The following services relative to computer operations and program preparation are available to programmers.

A. *OFF-LINE PROCESSING*

(1) *Program Filing*

Before a job is submitted to the San Diego Operations for processing, a unique 4-character alphanumeric identifier must be assigned to each subroutine. To allow each system to adapt this identifier to its needs and yet allow San Diego Operations to file routines in groups according to the probability of their being integrated into the same system, only the extreme left character will be controlled. This character will be a letter, and each system will be assigned one or more of these letters as needed. Letter assignments will be made by the Chief of Off-Line Operations. The remaining three characters will be numbers controlled by and assigned within individual systems. If this control is not desired by a system, San Diego Operations will assume such control upon request.

The following assignments have been submitted:

NTDS Utility System	A
NTDS Engineering Test System	E
NTDS Environmental Simulation System	N
NTDS Service Test System	S
Evaluation System	V

When a magnetic-tape reel not currently in use is needed for a job, a system or programmer should request reel assignment from the Control Center before (or at the time of) submitting the job. A reel number will be assigned to the system or programmer, thereby reserving the use of that reel for that system or programmer as long as it continues to be used actively.

All magnetic-tape reels will be housed by reel number except those being used for ASDEC. Reels for ASDEC may be checked out at the Control Center as needed and should be returned when not being used regularly.

Each system will have a system procedure library on file that will contain the L_0 versions of all subroutines integrated into that system. If a subroutine is added or changed, notice will be given to the person who is responsible for that system. In addition, a running tape that contains all actively integrated versions of such system will be on file for each system. Copies of the system running tape may be made and used in other areas, but the original must be kept in the Interim Computing Center.

Programs on cards will be housed according to alphanumeric deck IDs in a fireproof card safe. If a programmer desires, he may keep card decks in his possession; however, when a program is integrated into a system procedure library, a deck must be filed in the Interim Computing Center and only that file copy will be used to compile the system.

Paper tapes will be kept by each system or programmer unless a request is made to house them in the Interim Computing Center where they will be housed by deck ID.

(2) *Program Preparation*

The Off-Line Operations group transcribes program manuscripts to paper tape or cards. All paper-tape work is proofed and all card work is verified. All programs on cards are converted to magnetic tape before they are submitted to computer processing.

Normally, work should be routed through the San Diego Operations with a Service Request Form. Priority authorization is available for rush jobs.

It may be more expedient for a programmer to do the punching himself in a case where only a small amount of punching is involved and when his job is to go on the computer soon. The programmer should request use of a machine from the Control Center.

(3) *Program Listing*

Paper Tape Mode

All program outputs, dumps, etc., that cannot be obtained on the High-Speed Printer are on paper tape which must be listed on *off-line* Flexowriters.

Magnetic Tape Mode

Most outputs will be on magnetic tape and will be listed on the High-Speed Printer as an *off-line* operation. The High-Speed Printer may be used *on-line*. This will be an abnormal mode of operation; therefore, the programmer *must specify* that he wants to use the High-Speed Printer *on-line*. Any paper-tape outputs will be listed *off-line*.

B. *ON-LINE PROCESSING*

(1) *Program Compilation*

Computer operators compile all symbolic programs. Magnetically taped symbolic and absolute programs are kept on file as described previously. Listings and paper tapes are returned to the programmer.

(2) *Program Operation*

Closed Shop Mode

Jobs are performed by computer operators according to instructions on each service request. Standard sets of instructions will be developed (when possible) for frequent runs so that a programmer need specify only a certain standard run number, if it is applicable.

Any production jobs which require the use of ASDEC equipment will be performed in that area as time is made available. Programmers who anticipate the need to submit such orders should notify the Chief of Computer Operations, San Diego Operations Section, at least one week ahead of time so that ASDEC time may be requested. All other runs, except system integration runs, will be performed by the operators in the Interim Computing Center (if they wish, programmers may be present on pre-integration runs). Each programmer must provide adequate instructions to ensure that his job is performed to his specifications.

Open Shop Mode

Certain programmers from each system will, with operator assistance, perform integration runs.

If required by equipment, integration runs will be performed in the ASDEC area. The programmer must obtain ASDEC time for these runs in the usual manner. Operator assistance for ASDEC will be furnished by the ETS Section upon request.

All other integration runs will be performed in the Interim Computing Center. A normal service request is submitted but will carry a *flag* to denote an integration run. As the job nears computer time, the programmer will be notified so that he will have sufficient time to complete his preparations. If a programmer is not prepared by the time his job reaches the computer, other jobs will be performed until he is ready.

Maintenance personnel will run their own programs during maintenance time.

Operations for the Operational Programmers Training Courses for NTC personnel will be

performed by the course instructors and students.

C. *METHOD OF JOB TRANSMITTAL*

(1) *Job Request*

The proper form to use is the NTDS-SD Computer Center Service Request. All necessary directions are included on the form or are published for easy reference. The programmer should submit this request with his manuscript and/or tapes or cards for processing. A job will be completed as requested and returned to the programmer; normally, 24-hour (or better) service will be provided for any request.

Manuscripts and paper tapes should be transmitted in suitable envelopes accompanied by card decks in stacks held by rubber bands --- or contained in boxes --- clearly marked for easy identification. If a programmer finds that his instructions have not been followed, he should return the job to the Control Center for re-processing as soon as possible. A new service-request form is not necessary.

2. FLOW OF WORK THROUGH SAN DIEGO OPERATIONS

Jobs should be submitted to the pickup stations (these are located conveniently for all programmers). A messenger checks these stations for pickups several times each day for delivery to the control desk where the requests are checked for completeness and clarity before being logged in and assigned a job number. Jobs may be submitted directly to the control desk if desirable. In case of priority jobs, the latter action would ensure faster processing.

Jobs to be punched on the Flexowriter or keypunch are routed to Off-Line Operations where they are punched, proofed or verified, and cards are converted to magnetic tape. Then the job is returned to the control desk for further routing or to be logged out and returned to the programmer.

Jobs ready for the computer are submitted to a precheck of instructions and materials before being sent to the computer to be compiled and/or run to programmer specifications. The jobs are then returned to the control desk for further routing or to be returned to the programmer.

Jobs to be listed *off-line* are submitted to Off-Line Operations for listing on the Flexowriter or the printer and then returned to the control desk for routing to the programmer.

If a job cannot be completed at any station within San Diego Operations, the programmer is notified as soon as possible, and if possible the difficulty is corrected and processing is re-

sumed. If the error cannot be corrected easily, the job is logged out for return to the programmer.

Priority authorization is available from section heads, as needed. All jobs properly authorized (stamped) will be placed ahead of non-priority work at all stations.

Any programmer who wishes to learn the status of a job may contact the Control Center and obtain the information desired. Similarly, if a programmer should establish that he wants to stop processing on a job already submitted, he need only to inform the Control Center.

3. INTERNAL RECORDS OF SAN DIEGO OPERATIONS

The internal records of San Diego Operations consist of the following items:

Log-In/Log-Out Record (maintained by control desk).

Computer Log (Operators' Log; shows time on computer, job number, job name, name of person submitting, and results).

File of all Service Requests processed (control desk).

Catalog of all programs on file.

Operating procedure handbook for all standard programs (*Operators' Guide*).

Copies of *on-line* type-outs obtained during compilation are retained for one month.

SECTION B2

ST. PAUL OPERATIONS

ST. PAUL OPERATIONS

1. SECTION RESPONSIBILITIES

St. Paul Operations is responsible for the following activities:

1) Scheduling and operating NTDS computers and peripheral equipment in the CIC room.

2) Hiring and training sufficient computer operators to operate the NTDS computer center on an engineering service support basis for the NTDS Department.

3) Processing all computer input data.

4) Providing facilities and methods of filing magnetic tape, paper tapes, and cards necessary for the proper utilization of the computing center.

5) Maintaining all equipment and supplies such as cards, paper, etc.

6) Maintaining a library of standard utility routines.

7) Preparing and maintaining a computer operator's manual that covers the operating procedures of utility routines that have been established for use in the NTDS computer center.

8) Assigning C registers.

9) Coordinating all activities with the San Diego installation.

2. SERVICES AVAILABLE

The following services relative to computer operations and program preparation are available to the Naval Tactical Data System Department:

A. *OFF-LINE PROCESSING*

(1) *Distribution Center Services*

Each Service Request that enters St. Paul Operations is given individual attention. Handling includes

a) Individual job-number assignment;

b) Pre-processing inspection of material to ensure against loss;

c) Constant control to provide current status information on each job.

(2) *Program Preparation*

Program preparation facilities are available for two modes of input/output processing. These are the paper-tape mode and the magnetic-tape mode. Facilities available for preparation for these modes are

a) Manuscript conversion to paper tape or 80 column cards;

b) Paper-tape proofing and card verifying;

c) Card to magnetic tape conversion;

d) Off-line High-Speed Printer operations;

e) Off-line paper-tape listing.

B. *ON-LINE PROCESSING*

(1) *Library and Storage Facilities*

St. Paul Operations Computer processing provides the following services:

a) Magnetic-tape supply facilities;

b) Magnetic-tape library of utility routines;

c) Paper-tape library of utility routines;

d) Magnetic-tape storage area for tapes checked out to individuals or systems;

e) Punched-card storage area.

(2) *Program Operations*

St. Paul Operations computer processing operates in two basic modes.

Closed Shop Mode

1) St. Paul Operations runs primarily on a closed-shop basis; that is, St. Paul Operations provides all necessary operational needs for computer operations.

2) The programmer originating a service request may be present during the operation of his job. All processing will be performed by assigned computer operators.

Open Shop Mode

1) Designated programmers from each system will, with operator assistance, perform

integration runs.

2) Programmers may operate during initial training periods while assigned operators are learning the general flow and manipulation of new routines.

3) Maintenance personnel will run their own programs during scheduled maintenance time.

4) Operations for formal training courses will be performed by the course instructors and students.

C. *METHOD OF JOB TRANSMITTAL*

(1) *Job Request*

To request St. Paul Operations processing, a programmer should submit his material with an NTDS Computer Service Request Form A-1270C (Rev 3-60). Each job is processed in a numerical sequence. Under normal conditions, each job will be completed within 24 hours. If a job has not been processed properly, it should be returned to St. Paul Operations where it will be given special handling and priority. A new Service Request will not be necessary.

3. OPERATING PROCEDURES

A. *DISTRIBUTION CENTER PROCEDURES*

The following procedure for distribution of material for St. Paul Operations will provide documental control in handling Service Request Material (see Figure B2-1). The procedure is

1) All mail that comes into St. Paul Operations for processing will be logged in at the Distribution Center by the Material Controller.

2) Incoming material for processing will be separated into two groups:

   a) Material for computer processing;

   b) Material for *off-line* reproduction work or *off-line* preparation for later computer processing.

3) During this initial sort operation, the Material Controller includes necessary magnetic tapes or cards required to complete the operation. The job is then logged out to the appropriate group and placed in Box 1C or 1R (see Figure B2-2).

4) As material is completed by each group, it is placed in Box 2C or 2R. Completed material may be logged at any time for further distribution.

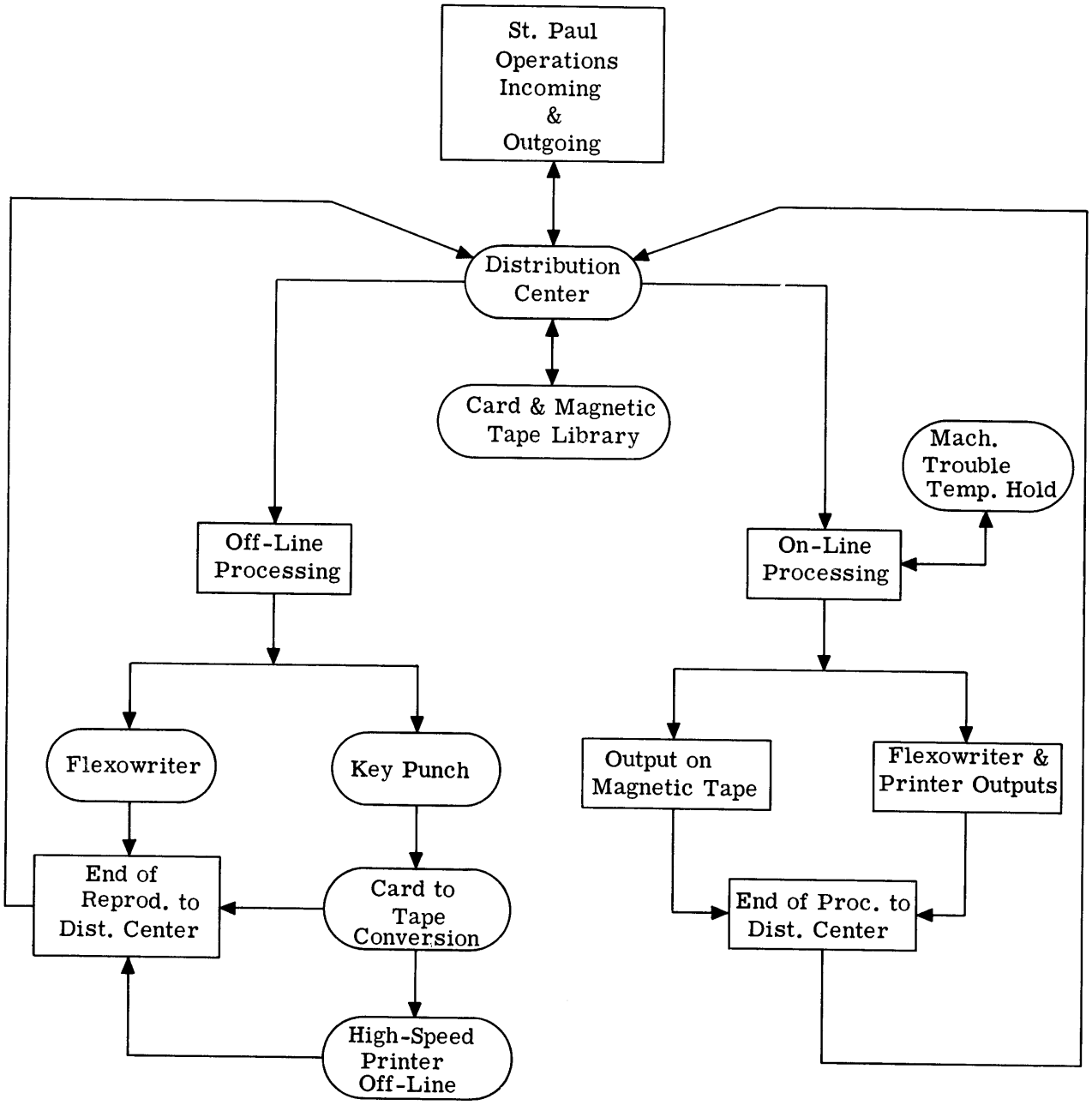


Figure B2-1. Flow of Material - St. Paul Operations Section

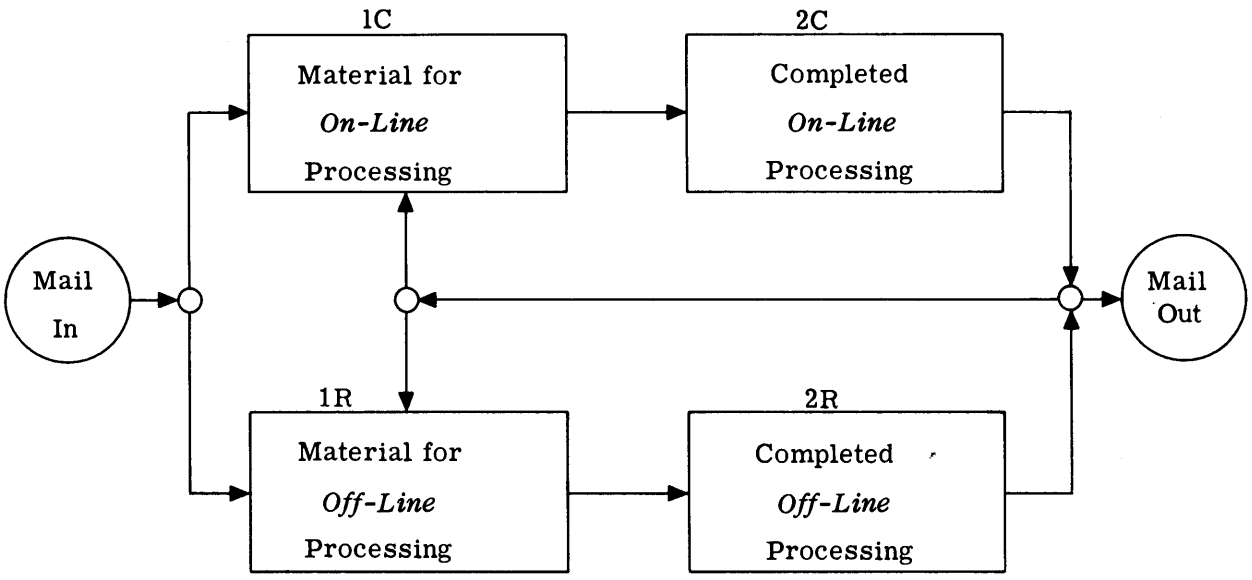


Figure B2-2. Job Logging

5) When material is returned to the Distribution Center, the Material Controller logs the time the material is returned and mails it or sorts it for further processing.

B. *OFF-LINE PROCESSING PROCEDURES*

All incoming material for program preparation (card preparation work, Flexowriter work, card-to-tape conversion, off-line printer work) is received from the Distribution Center. Material received is in one of the following forms:

- Paper Tape
- Manuscripts
- Magnetic Tape
- Punched or Blank Cards
- Flexowriter or High-Speed Printer Sheets

The above-mentioned types of material are contained in a processing container in numerical job-order sequence with the exception of cards. Cards are issued with the job. These cards are properly marked and noted as to what job they are associated with and the programmer's

name or section. Blank cards are drawn directly from the Distribution Center by key-punch operators.

Each job is completed separately. The operator who receives a request is responsible for completion of all necessary reproduction work required on the request. If a job is held up in processing, the following is noted on the request sheet:

Example: Time Production Delay - 0900 1/8/61
Completed Work - Key Punch Completed 1/9/61
Reason - C/T 1/9/61 Down; 1/10/61 Up
Continued - C/T Completed 1/10/61

When *off-line* processing is completed, the operator who accomplished the work checks the program with the Material Controller or another operator. After the material is checked, the operator and the checker initial the service request, make appropriate remarks, and place the program in the completed material container.

C. *ON-LINE PROCESSING PROCEDURES*

All material ready for computer processing is received from the Distribution Center. Material received is on either magnetic or paper tape. This material is kept in a processing container in a numerical job-identifier sequence. All job orders are handled separately and chronologically. Only one job is out of the container at any time. Job priority will be recognized when established by proper authority.

When a program is completed, it is briefly checked by the operator who then makes appropriate comments and log entries and places the program in the completed processing container.

D. *CLASSIFIED MATERIAL HANDLING PROCEDURES*

1) Classified material is transferred to and from St. Paul Operations by special courier service.

2) Classified material pickup and delivery stations are at Plant 5 (NTDS Central File Area) and Plant 2 (St. Paul Operations Distribution Center).

3) The classified-material courier personally hands all material for processing to the Material Controller.

4) Upon receipt of classified material, the Material Controller inspects contents of the envelope, logs in the request in the usual manner, and adds a red pencil "**X**" below the

job-request number. This red "X" will indicate that the material being processed is classified.

5) The Material Controller procures any necessary cards or tapes from the card or magnetic-tape library and places the job material in the "classified material processing container".

6) Classified service-request material is never taken from the Distribution Center until immediate processing can be accomplished.

7) The authorized typist and/or the computer operator who performs the work on a classified service request initials the service request when it is received from the Material Controller.

8) When the typist or the operator finishes work on the classified service request, he immediately returns the job to the Material Controller for further distribution.

   a) If the High-Speed Printer is used for any classified service request, the printer cover is covered during the operation.

   b) During a classified service-request operation, the computer operator performing the operation is responsible for keeping the area clear of all unauthorized individuals.

   c) The St. Paul Operations operator assumes full security responsibilities for any classified service-request job while it is in his possession.

9) When completed, classified service-request jobs are immediately returned to the Distribution Center. The Material Controller checks in the job and places in storage material that is to remain at St. Paul Operations. The Material Controller then personally hands outgoing material to the classified material courier for return to the originating station.

4. INTERNAL RECORDS

A. *NTDS COMPUTER SERVICE REQUEST A*-1270C (*REV* 3-60)

The NTDS Computer Service Request (see Figure B2-3) is a three-part form prepared by NTDS programmers and operators. It consists of

   a) White Original - returned to the originator when the program is completed;

   b) Yellow Carbon - filed at St. Paul Operations;

   c) Pink Carbon - retained by originator.

A-1270C (REV. 3-60)

JOB # _____

DATE _____

LOGGED IN BY _____

NTDS
COMPUTER SERVICE REQUEST

SECRET	CONF.	COMP. CONF.	UNCLASS	DATE	SERVICE REQUEST NO.

MATERIAL ENCLOSED

REQUESTED BY

SECT.	ROOM OR M.S.	PLANT	EXT.

☐ CHECK THIS BOX IF YOU WOULD LIKE TO BE PRESENT WHILE YOUR JOB IS RUN.

JOB DESCRIPTION:

FAULT PROCEDURE

☐ TRACE

DUMP ☐ CORE _____ TO _____

☐ DRUM _____ TO _____

☐ MNEMONIC ☐ ABSOLUTE

☐ CALL _____ PLT. _____ EXT. _____

SPECIAL INSTRUCTIONS: _____

COMMENTS: (FOR ST. PAUL OPER. USE ONLY)

Figure B2-3. NTDS Computer Service Request (REV 3-60)

This form provides NTDS programmers and operators with a documented method for requesting computer processing service and a means of communicating between programming operations and machine operations. Any special processing directions or comments are indicated on this form.

B. *ST. PAUL OPERATIONS DISTRIBUTION CENTER LOG SHEET*

The Distribution Center Log Sheet (see Figure B2-4) is a processing control form. This log is maintained by the Material Controller.

This log sheet is used to maintain constant control on all material processed through St. Paul Operations. This sheet provides

a) The job-request number and the time the request is received;

b) The material that is received with the request and the material added by St. Paul Operations to complete the operation;

c) The type of processing to be accomplished;

d) The location of each job being processed;

e) A record of the time utilized to complete each operation.

C. *NTDS COMPUTER LOG SHEET*

The NTDS Computer Log (see Figure B2-5) is a computer processing record maintained by computer operators and other NTDS personnel who operate NTDS computers.

The purpose of the NTDS Computer Log is to provide a brief summary of each computer operation. The information recorded in this log is

a) Operator's name, section, and the peripheral equipment utilized;

b) The time the operation was started, completed, and total amount of time used;

c) The time code indicating productive, idle, lost, pm, etc;

d) The St. Paul Operations operator's and the duty maintenance man's initials as verification of log entries;

e) The remarks section for general comments on the operation.

D. *DAILY COMPUTER UTILIZATION REPORT*

The Daily Computer Utilization Report is a daily progress report of computer operations.

JOB REQUEST NO.	RECEIVED DIST. CENTER	NAME:	REPRODUCTION PROC. IN	OUT	*	COMP.PROC. IN	OUT	*	OUT PUT PROC. IN	OUT	*	OUT OF DIST. CENTER
	DATE:	MATERIAL: PREP.SHEETS:	DATE:	DATE:		DATE:	DATE:		DATE:	DATE:		DATE:
		M TAPE										TIME:
	TIME:	PAPER TAPE	TIME:	TIME:		TIME:	TIME:		TIME:	TIME:		TOTAL PROC.TIME
		OTHER:										

* REQUIRED ROUTING (√)

JOB REQUEST NO.	RECEIVED DIST. CENTER	NAME:	REPRODUCTION PROC. IN	OUT	*	COMP.PROC. IN	OUT	*	OUT PUT PROC. IN	OUT	*	OUT OF DIST. CENTER
	DATE:	MATERIAL: PREP.SHEETS	DATE:	DATE:		DATE:	DATE:		DATE:	DATE:		DATE:
		M TAPE										TIME:
	TIME:	PAPER TAPE	TIME:	TIME:		TIME:	TIME:		TIME:	TIME:		TOTAL PROC.TIME
		OTHER:										

* REQUIRED ROUTING (√)

JOB REQUEST NO.	RECEIVED DIST. CENTER	NAME:	REPRODUCTION PROC. IN	OUT	*	COMP.PROC. IN	OUT	*	OUT PUT PROC. IN	OUT	*	OUT OF DIST. CENTER
	DATE:	MATERIAL: PREP.SHEETS	DATE:	DATE:		DATE:	DATE:		DATE:	DATE:		DATE:
		M TAPE										TIME:
	TIME:	PAPER TAPE	TIME:	TIME:		TIME:	TIME:		TIME:	TIME:		TOTAL PROC.TIME
		OTHER:										

* REQUIRED ROUTING (√)

JOB REQUEST NO.	RECEIVED DIST. CENTER	NAME:	REPRODUCTION PROC. IN	OUT	*	COMP.PROC. IN	OUT	*	OUT PUT PROC. IN	OUT	*	OUT OF DIST. CENTER
	DATE:	MATERIAL: PREP.SHEETS	DATE:	DATE:		DATE:	DATE:		DATE:	DATE:		DATE:
		M TAPE										TIME:
	TIME:	PAPER TAPE	TIME:	TIME:		TIME:	TIME:		TIME:	TIME:		TOTAL PROC.TIME
		OTHER:										

* REQUIRED ROUTING (√)

Figure B2-4. St. Paul Operations Distribution Center Log Sheet

CODE
1. PROD 3. IDLE 5. E.M. 7. MISCL.
2. LOST 4. LOST A 6. PM

NTDS COMPUTER LOG

NTDSUC SERIAL ____________

OPERATOR	SECTION	TYPE OF PROG.	TIME: DATE				INITIALS
			START	STOP	TOTAL	CODE	MAINT.
							S.P.O.
EQUIPMENT USED (√) ☐ DRUM ☐ NTDS TAPE ☐ CHARACTRON ☐ VIDEO PROC. ☐ DISPLAY ☐ FILE COMP.TAPE ☐ HS PRINTER			REMARKS				

OPERATOR	SECTION	TYPE OF PROG.	TIME: DATE				INITIALS
			START	STOP	TOTAL	CODE	MAINT.
							S.P.O.
EQUIPMENT USED (√) ☐ DRUM ☐ NTDS TAPE ☐ CHARACTRON ☐ VIDEO PROC. ☐ DISPLAY ☐ FILE COMP.TAPE ☐ HS PRINTER			REMARKS				

OPERATOR	SECTION	TYPE OF PROG.	TIME: DATE				INITIALS
			START	STOP	TOTAL	CODE	MAINT.
							S.P.O.
EQUIPMENT USED (√) ☐ DRUM ☐ NTDS TAPE ☐ CHARACTRON ☐ VIDEO PROC. ☐ DISPLAY ☐ FILE COMP.TAPE ☐ HS PRINTER			REMARKS				

OPERATOR	SECTION	TYPE OF PROG.	TIME: DATE				INITIALS
			START	STOP	TOTAL	CODE	MAINT.
							S.P.O.
EQUIPMENT USED (√) ☐ DRUM ☐ NTDS TAPE ☐ CHARACTRON ☐ VIDEO PROC. ☐ DISPLAY ☐ FILE COMP.TAPE ☐ HS PRINTER			REMARKS				

Figure B2-5. NTDS Computer Log Sheet

This report is not required in maintaining an efficient operation but does provide information for progress studies or organizational improvement planning. The report is prepared by the Senior Computer Operator and is extracted from the Computer Operator's Log and the Maintenance Log.

It provides supervising personnel with a daily summary of computer processing which consists of information or actual computer time utilized and operation accomplished during that time.

SECTION B3

FLEET PROGRAMMING CENTER

NOT AVAILABLE AT PRESENT

SECTION C

NTDS COMPUTERS

SECTION C1

AN/USQ-17 (SERIALS 1-5)

UNIT COMPUTER CHARACTERISTICS

AN/USQ-17 (SERIALS 1 - 5) UNIT COMPUTER CHARACTERISTICS

PART 1

GENERAL DESCRIPTION

1. BASIC INFORMATION

The AN/USQ-17 (Serials 1 through 5) Unit Computer is a stored program computer intended for the rapid handling of large quantities of complex data. Relative to other general purpose computer systems, this computer emphasizes random access storage and communication with external devices. The internal operations are performed in a parallel binary mode with a 30-bit instruction word and either a 15- or 30-bit data word. Instructions are of the 1-address type with an average execution time of 20 microseconds. Synchronous logic is used with a 2-megacycle clock rate.

The internal storage of the computer consists of a maximum of eight modular units. Each of these units consists of 4096 magnetic-core storage locations with a 30-bit capacity per location. Each of the possible 32,768 storage locations may be interpreted as a single 30-bit word or as two 15-bit words individually addressed. The control, arithmetic, and input/output sections of the computer have independent access to the storage section. Information flows over parallel transmission paths between the storage unit and other units in the system. A complete storage cycle of *read* followed by *write* requires eight microseconds.

The arithmetic and logical operations are performed in a parallel binary mode. In general, the result appears in a 30-bit accumulator register. Arithmetic is *ones* complement subtractive with a modulus $2^{30}-1$. An auxiliary arithmetic register, designated as the Q register, is used in multiply, divide, and logical operations. The Q register holds the multiplier at the beginning of a multiply operation, the least significant half of the double-length product at the end of the operation. The Q register also holds the least significant half of the dividend at the beginning of a divide operation; the quotient at the end of the operation.

Computer operation is controlled by a stored program capable of self-modification. Each program instruction contains a function code (6 bits), one storage address (15 bits), and three

execution modifiers (3 bits each). The execution modifiers provide for address incrementation, operand interpretation, and branch point designation. A storage address may be incremented by 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. The next sequential program step may be skipped under control of the contents of the arithmetic register.

Communication between the AN/USQ-17 (Serials 1 - 5) Computer and associated external equipment is normally handled by a block transfer of data with the timing under control of the external device. Operating asychronously with the main computer program, such transfers of data employ independent access to storage.

A communication path is established by a sequence of external-request and response signals between the external equipment and the computer. Such signals may originate either at the computer or at the external equipment. The main computer program is interrupted by external-request signals and establishes the communications channel. Once the link is established, the computer returns to the main program sequence. The block transfer of input or output data proceeds without program reference until completed.

A total of 18 input and 12 output channels is provided in the computer. These channels vary in size from 6 to 30 parallel lines per channel. The 30 channels are divided into three groups: input, output, and function lines. Buffering registers and control circuits are provided to permit concurrent activation of seven channels. The maximum possible transfer rate of input or output data over a given channel is 1,500,000 bits per second.

The computer is constructed primarily of diodes, transistors, and magnetic cores. The logical sections are implemented by transistor unit packages; however, the storage section contains combinations of transistors and magnetic-core arrays. The basic computer is contained in a single main cabinet 3 ft. by 3 ft. by 6 ft. Total power consumption is 1200 watts. Forced air cooling is provided without a heat exchanger.

2. REGISTERS

The AN/USQ-17 (Serials 1 - 5) Computer contains a number of registers in which data are held during computation. These registers are designated by a letter or by a letter and a numeral and are interconnected by parallel transmission paths over which information flows during processing. The registers are identified below in two categories: operational and transient.

Operational registers hold information from one instruction to another and are referenced in the operational description of each instruction. On the other hand, transient registers are temporary storage locations which are always cleared at the end of an instruction.

A. *OPERATIONAL REGISTERS*

The **A Register** (30 bits), or accumulator, is the principle arithmetic register. It is provided with parallel addition and shifting properties. In the majority of arithmetic instructions, the result of the operation is left in the A register for use in later program steps; thus, after addition or subtraction, the sum or difference remains in the accumulator. After multiplication, however, the most significant half of the product remains in the accumulator. After division, the remainder is left in the accumulator.

The contents of the A register may be shifted either to the left or to the right as described in the shift instructions. When the shift is to the left, the transfer is circular; that is, the rightmost digits are replaced with the leftmost digits. When the shift is to the right, the sign bit is extended by the number of bit positions shifted, and the lower order digits are discarded.

On certain instructions, the A and Q registers are shifted as a single 60-bit register. In all such cases, the A register represents the most significant half of the double-length quantity.

The **Q Register** (30-bits), or auxiliary arithmetic register, assists the accumulator in multiply, divide, and logical operations. The Q register has shifting and logical properties, but it has no addition or counting functions.

The contents of the Q register may be shifted to the right or left as a 30-bit register or as the lower order half of a 60-bit register in conjunction with the accumulator. With the exception of the shift paths, all communication is via the X register. Logical multiplication is performed on a transmission path between the Q and X registers.

The Q register holds the multiplier at the beginning of a multiply operation. As the product is formed by repeated additions and shifts, the multiplier digits are shifted to the right and discarded. In their place, the lower order digits of the double-length product are shifted into the Q register from the accumulator.

During a divide operation, a process essentially the reverse of multiplication takes place. The double-length dividend is shifted to the left, and quotient bits are inserted in the rightmost position of the Q register. At the end of the divide sequence, the quotient is assembled in the Q register, and the remainder is left in the accumulator.

The **P Register** (15-bits) is the program-address counter. This register holds the address of the next sequential instruction throughout the program. As each program address is transferred from the P register to the S register, the contents of the P register are increased by one. When executing jump instructions, the P register is cleared and a new program address is entered.

The **B Registers** (15 bits each) are address modifying registers generally used for indexing minor loops in a program. The contents of one register may be used to increment the operand address before execution of an instruction. A total of seven such registers are provided with one of these serving additional control functions as described in instruction 70.

The **C Registers** (6 to 30 bits each) are communication buffer registers through which input and output data are synchronized. A total of eight C registers is provided. C-register 0 has a special function as described below. Each of the other seven registers serves a group of input and output channels.

C-Register 0 (15 bits) is a communication-channel switching register, which by its content controls the channel selection for the other seven registers. The various bit positions in this register may be interpreted in seven groups: three bits for C-register 3 and two bits for each of the other C registers. The various binary combinations in each group then determine the channel selection for the corresponding C register.

C-Register 1 (6 bits) is an input/output register used to communicate monitoring information. The monitoring typewriter and keyboard entry communication is via this register. In addition, perforated paper tape may be processed through this register.

C-Register 2 (15 bits) normally is assigned to input and output devices requiring relatively few bits of information per message. Such devices as analog-to-digital converters fall in this category.

C-Register 3 (15 bits) normally is assigned as an external-function-request and response register. Computer function requests are sent to external equipment via this register and response signals return via this register.

C-Registers 4, 5, 6, and 7 (30 bits each) are the principle input/output registers. Information may be transferred through these registers over 30 parallel lines. Communication with displays, magnetic tapes, magnetic drums, other computers, etc., is handled through these registers.

B. *TRANSIENT REGISTERS*

The following registers are used in the manipulation of instruction words and data words during the execution of an instruction. These registers are not referenced in the description of the instructions and do not retain information from one operation to the next.

The **X Register** (30 bits) functions as an arithmetic communication register. It has complementing but no shift properties. The X register holds the operand from storage during all arithmetic operations. All communication between the A and Q registers and the rest of the operational registers is via the X register.

The **K Register** (6 bits) functions as a shift counter for all arithmetic operations involving shifts. The maximum shift count is 63. Multiply and divide operations are controlled by pre-setting the K register to 30 and by counting the operational steps.

The **S Register** (15 bits) holds the storage address during memory references. At the beginning of a storage-access period, the address is transferred to the S register. The contents of the S register are then translated to activate the storage selection system.

The **Z Register** (30 bits) serves as an operand buffer for storage references. During the *read* portion of the storage-access period, the Z register is cleared. The digit-reading amplifiers are then sampled to set the contents of Z corresponding to bits in storage. During the *write* portion of the storage-access period, the Z register controls the inhibit circuits to write or restore the disturbed storage register.

The **U Register** (30 bits) is the program control register. In other words, it holds the instruction word during the execution of an operation. The operation code and the various execution modifiers are translated from appropriate sections of the register. The lower order 15 bits of the U register have addition properties, modulus $2^{15}-1$. If an address modification is required before execution, the contents of the appropriate B register are added to the contents of the lower order 15 bits of the U register before execution.

The **R Register** (15 bits) functions as a communication register for the B registers. All internal transmissions to or from the B registers pass through the R register. It also holds the incrementing quantity during address modification. This register has complementing and counting provisions for incrementing the contents of the B register.

3. INSTRUCTION REPERTOIRE

Each step in the operation of the AN/USQ-17 (Serials 1 - 5) Computer is controlled by an instruction in a stored program. An instruction is a 30-bit word which is read from magnetic-core storage at the appropriate time and entered in the U register where it controls the particular operation required. Various portions of an instruction word are interpreted for various aspects of the required operation while the instruction resides in the U register. These portions, or designators, are identified by a lower case letter for reference in the description of the instructions. They are listed below as they appear from left to right in the U register.

f = function code designator (6 bits)
j = branch condition designator (3 bits)
k = operand interpretation designator (3 bits)
b = address modification designator (3 bits)
y = operand address designator (15 bits)

Normally, these designators become effective in the reverse of the order listed.

- The first action following the entry of an instruction in the U register is the modification of **y** - the operand address designator - by the contents of a B register as specified by the **b** designator. This modification consists of adding the contents of the B register to the **y** designator.
- The second action is normally a storage reference to obtain the quantity at the resultant storage address and to enter it in the Z register.
- The third action is a transmission of the contents of the Z register to the X register with the character of the transmission under the control of the **k** designator.
- Subsequent action affects the contents of the operational register as specified by the **f** designator and procures the next instruction as specified by the **j** designator.

The quantity which enters the X register in the above process is called the *operand.* Since it is dependent on a number of designators, the operand is identified by a symbol, (Y), in describing the various instructions. The storage location from which the operand comes is identified as Y. A letter in parentheses associated with an operational register is used as a shorthand symbol for the contents of the operational register; however, this relationship is not quite true in the case of the letter Y since the **k** designator may distort the quantity which comes from address Y before it becomes the operand (Y). Summarizing the shorthand symbols,

Y	=	the storage location of the operand
(Y)	=	the operand
(A)	=	the contents of the accumulator
(Q)	=	the contents of the Q register
(B2)	=	the contents of B-register 2
(C5)	=	the contents of C-register 5

Table C1-1 is a list of the computer's repertoire of instructions; each instruction is listed by its function code number and name.

A. *FUNCTION CODE DESIGNATOR* — **f** (6 bits)

The highest order six bits of an instruction designate the function to be performed by that instruction. All values other than 00 and 77 are defined in the instruction list. The two codes 00 and 77 are fault conditions, which if executed will cause the computation to stop. The fault light will then be illuminated on the operation control panel.

B. *BRANCH CONDITION DESIGNATOR* — **j** (3 bits)

This octal designator normally specifies the condition under which the next sequential instruction in the program will be skipped. This provides for branching from a sequence without executing the jump instruction. Particular significance is associated with this designator in the case of repeated instructions. In such cases, the exit from the repeat mode is under the control of the **j** designator. A skip of the next sequential instruction is determined by the following rules except where otherwise noted in the instruction list.

j = 0: Do not skip the next instruction.
j = 1: Skip the next instruction.
j = 2: Skip the next instruction if (Q) is positive.
j = 3: Skip the next instruction if (Q) is negative.
j = 4: Skip the next instruction if (A) is zero.
j = 5: Skip the next instruction if (A) is nonzero.
j = 6: Skip the next instruction if (A) is positive.
j = 7: Skip the next instruction if (A) is negative.

C. *OPERAND INTERPRETATION DESIGNATOR* — **k** (3 bits)

The operand interpretation designator controls the transmission of the operand from the Z register to the X register and vice versa.

TABLE C1-1. INSTRUCTION REPERTOIRE - AN/USQ-17 COMPUTER

CODE	FUNCTION NAME	CODE	FUNCTION NAME
00	NOT USED	40	ENTER LOGICAL PRODUCT
01	RIGHT SHIFT Q	41	ADD LOGICAL PRODUCT
02	RIGHT SHIFT A	42	SUBTRACT LOGICAL PRODUCT
03	RIGHT SHIFT AQ	43	MASKED COMPARISON
04	COMPARE	44	REPLACE LOGICAL PRODUCT
05	LEFT SHIFT Q	45	REPLACE ADD LOGICAL PRODUCT
06	LEFT SHIFT A	46	REPLACE SUBTRACT LOGICAL PRODUCT
07	LEFT SHIFT AQ	47	STORE LOGICAL PRODUCT
10	ENTER Q REGISTER	50	SELECTIVE SET
11	ENTER ACCUMULATOR	51	SELECTIVE COMPLEMENT
12	ENTER B REGISTER	52	SELECTIVE CLEAR
13	ENTER C REGISTER	53	SUBSTITUTE
14	STORE Q REGISTER	54	REPLACE SELECTIVE SET
15	STORE ACCUMULATOR	55	REPLACE SELECTIVE COMPLEMENT
16	STORE B REGISTER	56	REPLACE SELECTIVE CLEAR
17	STORE C REGISTER	57	REPLACE SUBSTITUTE
20	ADD	60	ARITHMETIC JUMP
21	SUBTRACT	61	MANUAL JUMP
22	MULTIPLY	62	INPUT JUMP
23	DIVIDE	63	OUTPUT JUMP
24	ADD REPLACE	64	ARITHMETIC RETURN JUMP
25	SUBTRACT REPLACE	65	MANUAL RETURN JUMP
26	ADD Q	66	INPUT RETURN JUMP
27	SUBTRACT Q	67	OUTPUT RETURN JUMP
30	LOAD A, ADD Q	70	REPEAT
31	LOAD A, SUBTRACT Q	71	INDEX SKIP
32	ADD Q AND STORE	72	INDEX JUMP
33	SUBTRACT Q AND STORE	75	INITIATE INPUT BUFFER
34	REPLACE ADD Q	76	INITIATE OUTPUT BUFFER
35	REPLACE SUBTRACT Q	77	NOT USED
36	REPLACE ADD ONE		
37	REPLACE SUBTRACT ONE		

Those instructions which read an operand but do not replace it after the arithmetic is performed are designated *read* instructions. Those instructions that do not read an operand but store one are designated *store* instructions. Instructions which both read and write operands are known as *replace* instructions.

For *read* instructions, the **k** designator controls the transmission of the operand from the Z register to the X register. This transmission may select only half of the contents of the 30-bit Z register and may or may not extend the resulting upper bit throughout the higher order 15 bits of the X register. This provision allows a half word to be treated either as a 15-bit positive number or as a 14-bit number plus sign. In the case of **k** = 0, 4, or 7, the lower order 15 bits of the U register are not used to specify a storage address for the procurement of the operand. For **k** = 0 and **k** = 4, the lower order 15 bits of the U register are used directly as a 15-bit operand. In the case of **k** = 7, the contents of the accumulator are used as a 30-bit operand.

The effect of the various values of the **k** designator on the operand are listed below. This applies to all *read* instructions except where noted under the individual instruction.

For *read* instructions:

k = 0: Clear the X register. Then transmit the lower order 15 bits of (U) to the lower order 15 bits of X.

k = 1: Clear the X register. Then transmit the lower order 15 bits of (Z) to the lower order 15 bits of X.

k = 2: Clear the X register. Then transmit the higher order 15 bits of (Z) to the lower order 15 bits of X.

k = 3: Clear the X register. Then transmit the entire 30 bits of (Z) to X.

k = 4: Clear the X register. Then transmit the lower order 15 bits of (U) to the lower order 15 bits of X. If bit-position 14 of (X) is *one*, set each of the higher 15 bits of X to *one*.

k = 5: Clear the X register. Then transmit the lower order 15 bits of (Z) to the lower order 15 bits of X. If bit-position 14 of (X) is *one*, set each of the higher order 15 bits of X to *one*.

k = 6: Clear the X register. Then transmit the higher order 15 bits of (Z) to the lower order 15 bits of X. If bit-position 14 of (X) is *one*, set each of the higher order 15 bits of X to *one*.

k = 7: Clear the X register. Then transmit (A) to all 30 bits of the X register.

The **k** designator controls the transmission from the X register to the Z register on *write* operations. The effect of the **k** designator in *store* instructions is defined below except where noted under the individual instructions.

For *store* instructions:

k = 0: Clear the Q register. Then transmit (X) to the Q register.

k = 1: Replace the lower order 15 bits of the quantity stored at address Y with the lower order 15 bits of (X), leaving the higher order 15 bits undisturbed.

k = 2: Replace the higher order 15 bits of the quantity stored at the address Y with the lower order 15 bits of (X), leaving the lower order 15 bits undisturbed.

k = 3: Replace the quantity stored at address Y with (X).

k = 4: Clear the accumulator. Then add (X) to the accumulator.

k = 5: Replace the lower order 15 bits of the quantity stored at address Y with the complement of the lower order 15 bits of (X), leaving the higher order 15 bits undisturbed.

k = 6: Replace the higher order 15 bits of the quantity stored at address Y with the complement of the lower order 15 bits of (X), leaving the lower order 15 bits undisturbed.

k = 7: Replace the quantity stored at address Y with the complement of (X).

The *replace* instructions combine a reading operation with a writing operation. For the read portion, the **k** designator has the same meaning as the *read* instructions. For the write portions, a different interpretation is used.

For *replace* instructions:

k = 0: Not used.

k = 1: During the read portion, clear the X register. Then transmit the lower order

15 bits of (X) to the lower order 15 bits of X. During the write portion, replace the lower order 15 bits of the quantity stored at address Y with the lower order 15 bits of (X), leaving the higher order 15 bits undisturbed.

k = 2: During the read portion, clear the X register. Then transmit the higher order 15 bits of (Z) to the lower order 15 bits of X. During the write portion, replace the higher order 15 bits of the quantity stored at address Y with the lower order 15 bits of (X), leaving the lower order 15 bits undisturbed.

k = 3: During the read portions, clear the X register. Then transmit the entire 30 bits of (Z) to X. During the write portion, replace the quantity stored at address Y with (X).

k = 4: Not used.

k = 5: During the read portion, clear the X register. Then transmit the lower order 15 bits of (Z) to the lower order 15 bits of X. If bit-position 14 of (X) is *one*, set each of the higher order 15 bits of X to *one*. During the write portion, replace the lower order 15 bits of the quantity stored at address Y with the lower order 15 bits of (X), leaving the higher order 15 bits undisturbed.

k = 6: During the read portion, clear the X register. Then transmit the higher order 15 bits of (Z) to the lower order 15 bits of X. If bit-position 14 of (X) is *one*, set each of the higher order 15 bits of X to *one*. During the write portion, replace the higher order 15 bits of the quantity stored at address Y with the lower order 15 bits of (X), leaving the lower order 15 bits undisturbed.

k = 7: Not used.

D. *ADDRESS MODIFICATION DESIGNATOR*. — **b** (3 bits)

The address modification designator specifies which of the B registers, if any, will be used to modify the **y** designator before a storage reference is made. If a modification takes place, the contents of a B register are added to the lower order 15 bits of (U). This operation employs an additive accumulator; hence, the quantity consisting of all *zeros* cannot occur as a result unless both the **y** designator and the contents of the B register are all *zeros*. The effect of the various values of the **b** designator are

b = 0: Do not modify **y**.

b $\neq$ 0: Add the contents of selected B register to **y**.

E. *OPERAND ADDRESS DESIGNATOR* — **y** (15 bits)

The operand address designator specifies, subject to modification, which of the 32,768 storage locations will be referenced in the execution of an instruction. During execution, most instructions reference this storage location once. In some cases, however, the operand is read from storage, operated upon, and then returned to the same storage location. Such instructions are described as *replace* instructions in the listing. Other instructions, such as those with **k** = 0 or **k** = 4, make no storage reference for an operand but take the operand directly from the lower 15 bits of the instruction word itself.

4. LIST OF INSTRUCTIONS

01 *Right Shift Q*

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.

02 *Right Shift A*

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.

03 *Right Shift AQ*

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 bits of (Q). The higher order bits of (A) are replaced with the original sign bit as the word is shifted. Only the lowest order six bits of (Y) are recognized for this instruction.

04 *Compare*

This instruction compares (Y) with (A) and/or (Q). It does not alter either (A) or (Q). The branch condition designator is interpreted in a special way for this instruction.

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).

j = 3: Skip the next instruction if (Y) is greater than (Q).

j = 4: Skip the next instruction if (Q) is greater than, or equal to, (Y) and (Y) is greater than (A).

j = 5: Skip the next instruction if (Y) is greater than (Q) or if (Y) is less than, or equal to, (A).

j = 6: Skip the next instruction if (Y) is less than, or equal to, (A).

j = 7: Skip the next instruction if (Y) is greater than (A).

05 *Left Shift Q*

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.

06 *Left Shift A*

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.

07 *Left Shift AQ*

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 vice versa. Only the lower order six bits of (Y) are recognized by this instruction.

10 *Enter Q Register*

Clear the Q register. Then transmit (Y) to Q.

11 *Enter Accumulator*

Clear the accumulator. Then add (Y) to the accumulator.

12 *Enter B Register*

Clear the contents of B-register **j**. Then transmit the lower order 15 bits of (Y) to B-register **j**. The higher order 15 bits of (Y) are ignored in this instruction. The **j** designator is used to specify the selected B register for this instruction and is not available for its normal function.

13 *Enter C Register*

Clear the contents of C-register **j**. Then transmit (Y) to C-register **j**. If the selected C register is not a 30-bit register, the lower order portion of (Y) is transmitted and the higher order portion ignored. Finally, set the selected C register indicator bit to *one*. The **j** designator is used to specify the selected C register for this instruction and is not available for its normal function.

14 *Store Q Register*

Store (Q) at storage address Y as directed by the **k** designator. For **k** = 0, clear the Q register. Then transmit the complement of (X) to the Q register, which has the effect of complementing (Q).

15 *Store Accumulator*

Store (A) at storage address Y as directed by the **k** designator. For **k** = 4, clear the accumulator. Then transmit the complement of (X) to the accumulator, which has the effect of complementing (A).

16 *Store B Register*

Form a 30-bit quantity in the X register, the lower order 15 bits of which correspond to the contents of B-register **j** and the higher order 15 bits of which are *zero*. Then store (X) at storage address Y as directed by the **k** designator. The **j** designator is used to specify the selected B register for this instruction and is not available for its normal function.

17 *Store C Register*

Store the contents of C-register **j** at storage address Y as directed by the **k** designator. If the selected C register is not a 30-bit register, the higher order bits are interpreted as *zeros*. Clear the selected C register indicator bit but not the register. The **j** designator is used to specify the selected C register for this instruction and is not available for its normal function.

20 *Add*

Add (Y) to the previous contents of the accumulator.

21 *Subtract*

Subtract (Y) from the previous contents of the accumulator.

22 *Multiply*

Multiply (Q) times (Y) leaving the double-length product in AQ. If the factors are considered as integers, the product is an integer in AQ. The **j** designator is interpreted prior to end correction, thus, permitting overflow detection by sensing the sign of the accumulator.

23 *Divide*

Divide (AQ) by (Y) leaving the quotient in the Q register and the remainder in the accumulator. The remainder bears the same sign as the quotient. The **j** designator is interpreted prior to end correction, thus, permitting overflow detection by sensing the sign of the accumulator.

24 *Add Replace*

Add (Y) to the previous contents of the accumulator. Then store (A) at the storage address Y as directed by the **k** designator.

25 *Subtract Replace*

Subtract (Y) from the previous contents of the accumulator. Then store (A) at the storage address Y as directed by the **k** designator.

26 *Add Q*

Shift (A) and (Q) left 30 places as a single 60-bit register. Then add (Y) to the accumulator. Next, shift (A) and (Q) left 30 places as a single 60-bit register. The contents of the accumulator are undisturbed by this instruction. The **j** designator has special meaning in this instruction:

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.

27 *Subtract Q*

Shift (A) and (Q) left 30 places as a single 60-bit register. Then subtract (Y) from the accumulator. Next, shift (A) and (Q) left 30 places as a single 60-bit register. The contents of the accumulator are undisturbed by this instruction. The **j** designator has special meaning in this instruction as listed under 26, *Add Q*.

30 *Load A Add Q*

Clear the accumulator, add (Q) to the accumulator, and add (Y) to the accumulator.

31 *Load A Subtract Q*

Clear the accumulator, subtract (Q) from the accumulator, and add (Y) to the accumulator.

32 *Add Q and Store*

Add (Q) to the previous contents of the accumulator; store (A) at storage address Y as directed by the **k** designator. The **j** designator is not interpreted for values 0 and 4 of the **k** designator.

33 *Subtract Q and Store*

Subtract (Q) from the previous contents of the accumulator. Then store (A) at storage address Y as directed by the **k** designator. The **j** designator is not interpreted for values 0 and 4 of the **k** designator.

34 *Replace Add Q*

Clear the accumulator, add (Q) to the accumulator, add (Y) to the accumulator, and then store (A) at storage address Y as directed by the **k** designator.

35 *Replace Subtract Q*

Clear the accumulator. Subtract (Q) from the accumulator. Next, add (Y) to the accumulator. Then store (A) at storage address Y as directed by the **k** designator.

36 *Replace Add One*

Clear the accumulator. Add one to the accumulator. Next, add (Y) to the accumulator. Then store (A) at storage address Y as directed by the **k** designator.

37 *Replace Subtract One*

Clear the accumulator. Add minus one to the accumulator. Next, add (Y) to the accumulator. Then store (A) at storage address Y as directed by the **k** designator.

40 *Enter Logical Product*

Clear the accumulator. Form the bit-by-bit product of (Y) and (Q) in the X register. Then add (X) to the accumulator.

41 *Add Logical Product*

Form the bit-by-bit product of (Y) and (Q) in the X register. Then add (X) to the previous contents of the accumulator.

42 *Subtract Logical Product*

Form the bit-by-bit product of (Y) and (Q) in the X register. Then subtract (X) from the previous contents of the accumulator.

43 *Masked Comparison*

Form the bit-by-bit product of (Y) and (Q) in the X register. Next, subtract (X) from the previous contents of the accumulator. Then perform the branch point evaluation for skipping the next sequential instruction as directed by the **j** designator. Finally, add (X) to the accumulator.

This instruction results in no net change in the contents of any operational register. Through the branch condition designator, it provides a comparison of a portion of (Y) with the contents of the accumulator.

44 *Replace Logical Product*

Clear the accumulator. Form the bit-by-bit product of (Y) and (Q) in the X register. Next, add (X) to the accumulator. Then store (A) at storage address Y as directed by the **k** designator.

45 *Replace Add Logical Product*

Form the bit-by-bit product of (Y) and (Q) in the X register. Next, add (X) to the previous contents of the accumulator. Then store (A) at storage address Y as directed by the **k** designator.

46 *Replace Subtract Logical Product*

Form the bit-by-bit product of (Y) and (Q) in the X register. Next, subtract (X) from the previous contents of the accumulator. Then store (A) at storage address Y as directed by the **k** designator.

47 *Store Logical Product*

Form the bit-by-bit product of (A) and (Q) in the X register. Then store (X) at storage address Y as directed by the **k** designator. The **j** designator is not interpreted for values 0 and 4 of the **k** designator.

50 *Selective Set*

Set individual bits of (A) to *one* corresponding to *ones* in (Y), leaving the remaining bits of (A) unaltered.

51 *Selective Complement*

Complement individual bits of (A) corresponding to *ones* in (Y) leaving the remaining bits of (A) unaltered.

52 *Selective Clear*

Clear individual bits of (A) corresponding to *ones* in (Y) leaving the remaining bits of (A) unaltered.

53 *Substitute*

Clear individual bits of (A) corresponding to *ones* in (Q) leaving the remaining bits of (A) unaltered. Next, form the bit-by-bit product of (Y) and (Q) in the X register. Then set individual bits of (A) to *one,* corresponding to *ones* in (X), leaving the remaining bits of (A) unaltered.

This instruction has the effect of replacing bits of (A) with bits of (Y) corresponding to *ones* in (Q).

54 *Replace Selective Set*

Set 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 as directed by the **k** designator.

55 *Replace Selective Complement*

Complement individual bits of (A) corresponding to *ones* in (Y) leaving the remaining bits of (A) unaltered. Then store (A) at storage address Y as directed by the **k** designator.

56 *Replace Selective Clear*

Clear individual bits of (A) corresponding to *ones* in (Y) leaving the remaining bits of (A) unaltered. Then store (A) at storage address Y as directed by the **k** designator.

57 *Replace Substitute*

Clear individual bits of (A) corresponding to *ones* in (Q) leaving the remaining bits of (A) unaltered. Form the bit-by-bit product of (Y) and (Q) in the X register. Next, set individual bits of (A) to *one,*corresponding to *ones* in (X), leaving the remaining bits of (A) unaltered. Then store (A) at storage address Y as directed by the **k** designator.

This instruction has the effect of replacing bits of (Y) with bits of (A) corresponding to *zeros* in (Q).

60 *Arithmetic Jump*

This instruction clears the program address register, P, and enters a new program address for certain conditions of the contents of arithmetic registers. The **j** designator is interpreted in a special way for this instruction, determining the conditions under which a jump in a program address occurs.

j = 0: No action. Continue with the current program sequence.

j = 1: Execute jump.

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.

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, then 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.

61 *Manual Jump*

This instruction clears the program address register P and enters a new program address for certain conditions of manual key selections. The **j** designator is interpreted in a special way for this instruction, determining the conditions under which a jump in a program address occurs.

j = 0: Execute jump regardless of key selections.

j = 1: Execute jump if Key 1 is selected.

j = 2: Execute jump if Key 2 is selected.

j = 3: Execute jump if Key 3 is selected.

j = 4: Execute jump. Then stop computation.

j = 5: Execute jump. Then stop computation if Key 5 is selected.

j = 6: Execute jump. Then stop computation if Key 6 is selected.

j = 7: Execute jump. Then stop computation if Key 7 is selected.

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, then 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.

The program may be stopped by certain key selections upon the execution of this instruction. The **j** designator specifies which key selections are effective.

62 *Input Jump*

This instruction clears the program address register, P, and enters a new program address for certain conditions determined by the status of the C register. The **j** designator is interpreted in a special way for this instruction to designate which C-register status will be examined. If the C-register **j** indicator bit is in the *one* state, indicating information in the register, the jump condition is satisfied. If the selected indicator bit

is in the *zero* state, the jump condition is not satisifed. 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, 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.

63 *Output Jump*

This instruction clears the program address register, P, and enters a new program address for certain conditions determined by the status of the C register. The **j** designator is interpreted in a special way for this instruction to designate which C-register status will be examined. If the C-register **j** indicator bit is in the *zero* state, indicating no information in the register, the jump condition is satisfied. If the selected indicator bit is in the *one* state, the jump condition is not satisfied. 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, 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.

64 *Arithmetic Return Jump*

This instruction executes a return jump sequence for certain conditions determined by the contents of the arithmetic register. The **j** designator is interpreted in a special way for this instruction, determining the conditions under which the return jump sequence is executed.

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.

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 (shown above), the following sequence is performed:

Clear the X register and transmit (P) to the lower order 15 bits of X. Clear the P register and transmit the lower order 15 bits of (Y) to P. Then store the lower order 15 bits of (X) in the lower order 15 bits of storage address (P) leaving the higher order 15 bits undisturbed. Finally, increase (P) by one for the program address of the next instruction.

65 *Manual Return Jump*

This instruction executes a return jump sequence for conditions determined by manual key selections. The j designator is interpreted in a special way for this instruction, determining the conditions under which the return jump sequence is executed.

j = 0: Execute return jump regardless of key selections.

j = 1: Execute return jump if Key 1 is selected.

j = 2: Execute return jump if Key 2 is selected.

j = 3: Execute return jump if Key 3 is selected.

j = 4: Execute return jump. Then stop computation.

j = 5: Execute return jump. Then stop computation if Key 5 is selected.

j = 6: Execute return jump. Then stop computation if Key 6 is selected.

j = 7: Execute return jump. Then stop computation if Key 7 is selected.

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, the following sequence is performed.

Clear the X register and transmit (P) to the lower order 15 bits of X. Clear the P register and transmit the lower order 15 bits of (Y) to P. Store the lower order 15 bits of (X) in the lower order 15 bits of storage address (P) leaving the higher order 15 bits undisturbed. Then increase (P) by one for the program address of the next instruction.

66 *Input Return Jump*

This instruction executes a return jump sequence for certain conditions determined by the status of the C register. The j designator is interpreted in a special way for this in-

struction to designate which C register status will be examined. If the C-register j indicator bit is in the *one* state, indicating information in the register, the return jump condition is satisfied. If the selected indicator bit is in the *zero* state, the return jump condition is not satisfied. 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, the following sequence is performed:

Clear the X register and transmit (P) to the lower order 15 bits of X. Clear the P register and transmit the lower order 15 bits of (Y) to P. Then store the lower order 15 bits of (X) in the lower order 15 bits of storage address (P) leaving the higher order 15 bits undisturbed. Finally, increase (P) by one for the program address of the next instruction.

67 *Output Return Jump*

This instruction executes a return jump sequence for certain conditions determined by the status of the C register. The j designator is interpreted in a special way for this instruction to designate which C register will be examined. If the C-register j indicator bit is in the *zero* state, indicating no information in the register, the return jump condition is satisfied.

If the selected indicator bit is in the *one* state, the return jump condition is not satisfied. 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, the following sequence is performed:

Clear the X register and transmit (P) to the lower order 15 bits of X. Clear the P register and transmit the lower order 15 bits of (Y) to P. Next, store the lower order 15 bits of (X) in the lower order 15 bits of storage address (P) leaving the higher order 15 bits undisturbed. Then increase (P) by one for the program address of the next instruction.

70 *Repeat*

Initiate a repeat mode of operation which will cause the next sequential instruction to be repeated (Y) times or until a branch condition is satisfied, whichever occurs first. Maintain the number of remaining executions in B-register 7. Only the lower order 15 bits of (Y) are used in this instruction. The j designator is interpreted in a special way for this instruction. The mode of modification of the repeated instruction is specified by

the j designator as

j = 0 or 4: Do not modify the repeated instruction after each individual execution.

j = 1 or 5: Increase Y by one after each execution of the repeated instruction.

j = 2 or 6: Decrease Y by one after each execution of the repeated instruction.

j = 3 or 7: Repeat the initial B-register modification of the repeated instruction before each execution.

This instruction is implemented by the following sequence:

Clear B-register 7. Transmit the lower order 15 bits of (Y) to B-register 7. Record the lower two bits of the j designator for interpretation during the repeat mode. Establish the repeat mode of operation and read the next sequential instruction.

71 *Index Skip*

If the contents of B-register j are equal to (Y), skip the next instruction in the current sequence, proceed to the following instruction, and clear B-register j. If the contents of B-register j are not equal to (Y), proceed to the next instruction in the sequence in a normal manner. Increase the contents of B-register j by one. The j designator 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 with B-register j.

72 *Index Jump*

If the content of B-register j is nonzero, execute a jump in the program address to address (Y). Reduce the contents of B-register j by one. If the contents of B-register j are zero, proceed to the next instruction in a normal manner. Do not alter the contents of B-register j.

The j designator 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.

75 *Initiate Input Buffer*

This instruction establishes an input buffer via C-register j to magnetic-core storage with an initial storage address (Y). Subsequent to this instruction, 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 current next address will be maintained throughout the buffer process in the lower order 15 bits of magnetic-core storage address **j**. This mode will continue until it is superseded by a subsequent initiation of an input or output buffer via the same C register or until the higher order half and the lower order half of storage address **j** contain equal quantities, whichever occurs first.

This instruction is implemented as follows:

If **k** = 3, store (Y) in storage location **j**. If **k** $\neq$ 3, store the lower order 15 bits of (Y) in the lower order half of storage location **j** leaving the higher order half undisturbed. In either case, set the C-register **j** status designator to input mode active. Proceed to the next instruction.

The **j** designator is used to specify the selected C register for this instruction and is not available for its normal function.

76 *Initiate Output Buffer*

This instruction establishes an output buffer via C-register **j** from initial storage address (Y) in magnetic-core storage. Subsequent to this instruction, the individual transfers will be executed at a rate usually determined by external devices. The storage address initially established by this instruction will be advanced by one following each individual transfer. The current next address will be maintained throughout the buffer process in the lower order 15 bits of magnetic-core storage-address **j**. This mode will continue until it is superseded by a subsequent initiation of an input or output buffer via the same C register or until the higher order half and the lower order half of storage-address **j** contain equal quantities, whichever occurs first.

This instruction is implemented as follows:

If **k** = 3, store (Y) in storage location **j**. If **k** $\neq$ 3, store the lower order 15 bits of (Y) in the lower order half of storage-location **j** leaving the higher order half undisturbed. In either case, set the C-register **j** status designator to output mode active. Proceed to the next instruction.

The **j** designator is used to specify the selected C register for this instruction and is not available for its normal function.

5. CONTROL SEQUENCES

The execution of an instruction is divided into a number of lesser operations called *sequences.* Each performs a logical portion of an instruction. Sequences are further subdivided into steps, each of which performs an elementary manipulation on bits of data or makes an elementary decision on the basis of a bit of data. A sequence is the smallest portion of a computer instruction which may be executed as an entity. A manual selection on the operation control panel permits an operator manually to step through an instruction one sequence at a time.

Each of the seven sequences provided in the AN/USQ-17 (Serials 1 - 5) is identified by a letter. The first four sequences to be described form the execution of a normal instruction. The last three sequences execute a buffer operation on request of external synchronizing signals.

"*A*" *Sequence—Read Next Instruction*

This sequence reads the next instruction from storage and modifies the operand address as directed by the **b** designator. The address for the next instruction always resides in the P register at the beginning of the "A" sequence. As soon as the storage reference has been initiated, (P) is increased by one in anticipation of the next sequential instruction in the current program. The content of a B register, as specified by the **b** designator is then added to the **y** designator. This addition is accomplished by transferring the contents of the selected B register to the R register and adding (R) to **y**. If **b**=0, the cleared R register content is added to the **y** designator.

The "A" sequence for an instruction execution in the repeat mode represents a significant departure from the normal operation described above. During a repeat mode, the previous instruction remains in the U register and no storage reference is made. An increment is entered in the R register, as directed by the repeat status designator, and (R) is added to the **y** designator as in a normal execution. The contents of B-register 7 are decreased by one in the repeat mode. If the resultant value of (B7) is *zero,* the repeat mode is terminated. The content of the P register is not altered in a repeat mode.

"*B*" *Sequence—Read Operand*

This sequence reads the operand from storage and performs any preliminary arithmetic appropriate for the particular instruction being implemented. The **k** designator plays a major role in the execution of this sequence. If **k**=0, 4, or 7, the operand is not obtained

from storage but is transferred from the U or A registers to the X register. If a storage reference is made, the operand is transferred from the Z register to the X register as described in the characterisitcs of the **k** designator. If arithmetic operations are required prior to procuring the operand, these operations are performed before the operand is transferred to the X register.

"*C*" *Sequence—Arithmetic Manipulation*

This sequence completes all arithmetic processes required for the execution of the instruction. Add, subtract, multiply, divide, and shift operations are all performed in this sequence. No storage references are made. If a jump in the program address is required, the contents of the P register and the X register are interchanged during this sequence. If a *store* operation is required, this sequence is followed by the "D" sequence. If no *store* operation is required, this sequence is followed by the "A" sequence.

"*D*" *Sequence—Store Operand*

This sequence is executed only on those instructions which require a result to be written into storage. In the case of a return jump instruction, the storage location comes from the P register. In the case of all other instructions, the storage address comes from the U register. In any case, the quantity contained in the X register at the beginning of the "D" sequence is stored at the specified storage location with the **k** designator controlling the transfer from the X register to the Z register.

"*E*" *Sequence—Read Buffer Address*

This sequence is enabled by an external request for the transfer of a word of data between the storage section and the input/output section of the computer. This request may occur at any time with respect to the main program sequences; hence, the "E" sequence may be executed between any two of the normal sequences. Once this sequence has been initiated, the normal sequences are delayed until the entire buffer operation has been completed. If a buffer operation is required, this sequence reads an address from a special storage location and enters it in the R register. This sequence then enables the "F" sequence. If a transfer operation is required, the "E" sequence reads an address from B-register 6, performs the transfer between the required C register and storage, and then increases the address by one and returns it to B-register 6. In this case, no other buffer sequence is required.

"F" Sequence—Transfer Buffer Word

This sequence buffers a word of input or output data between a selected C register and the storage address contained in the R register. The "G" sequence is then enabled.

"G" Sequence—Store Buffer Address

This sequence increases the buffer address contained in the R register by one and stores the result in the special storage location from which it originally came. During this storage reference, the new buffer address being recorded is compared with a terminal address contained in the upper half of the same storage location. If the two addresses are equal, the buffer mode is terminated. In either case, the buffer operation for this word of input or output data is completed, and the main program sequences resume.

6. SPECIAL MODES OF OPERATION

The AN/USQ-17 (Serials 1 - 5) Computer performs a number of complex operations which may be identified as *special modes* of operation. Several of these special modes require the existence of magnetic-core storage locations as implicit addresses for control information associated with the functioning of the special mode. These special storage locations are assigned to beginning values of the address range as follows:

00000	initial starting address from a master clear
00001	buffer control word for C-register 1
00002	buffer control word for C-register 2
00003	buffer control word for C-register 3
00004	buffer control word for C-register 4
00005	buffer control word for C-register 5
00006	buffer control word for C-register 6
00007	buffer control word for C-register 7
00010	real-time clock
00011	interrupt instruction
00012	interrupt exit
00013	interrupt entrance

A. *REPEAT MODE OF OPERATION*

A repeat mode of operation is initiated by the execution of an initiate repeat (70) instruction The purpose of the repeat mode is to perform multiple executions of an instruction with minor variations in the address of the operand. This instruction is used to process a list of data words with a minimum of storage references for control purposes. Such operations as search-

ing a list of words for coincidence with a reference word, or adding a column of numbers, may be performed much more rapidly with this mode than with an equivalent subroutine. The **j** designator plays a major role in this mode. It permits the repeat mode to be terminated when a particular execution satisfies the branch condition. For example, a search for coincidence with a reference word could utilize the branch condition designator in the repeated instruction. When, and if, the coincident item is located, the repeat mode is terminated. The address of the selected word can then be obtained by subtracting the number of incompleted executions, as indicated in B-register 7, from the terminal address of the search.

The repeat mode is implemented through a control designator which modifies the execution of the "A" sequence when the repeat mode is active. A nonzero value of this designator suppresses the procurement of the next sequential instruction from storage. The operand address of the repeated instruction is modified during the "A" sequence of each execution of the instruction to advance the address of the next item in the list being processed. The number of incompleted executions is maintained in B-register 7 throughout the repeat mode.

If a repeat mode is terminated by a branch condition, the instruction in the storage location following the repeated instruction is skipped. If the repeat mode is terminated by exhausting the repeat count in B-register 7, the instruction in the storage location following the repeated instruction is executed.

B. *BUFFER MODE OF OPERATION*

A buffer mode of operation is initiated by the execution of an initiate input buffer (75) or initiate output buffer (76) instruction. The purpose of the buffer mode is to provide input and output communication between the magnetic-core storage and external equipment via the communication registers. This operation is independent of, and in parallel with, the main computer program. Provisions are made in the computer for up to seven such independent buffering operations at one time; one through each of the seven communication registers.

Each buffer control utilizes a special magnetic-core storage location for control information. Two 15-bit addresses are retained in this storage location and are referenced and modified as the buffer operation is performed. The higher order 15 bits of a buffer control word represent the terminal address plus one for the buffered data. Upon initiation of the buffer, the lower order 15 bits of a control word represent the initial storage address of the buffered data. As each item is transferred, the lower portion of the buffer control word is increased by one until this portion is identical with the upper portion of the control word. A control signal is generated by the equality of the upper and lower halves of a control word which ter-

minates the buffer mode for the C register involved.

All data transferred into or out of the magnetic-core storage via a buffer mode are treated as 30-bit words. If the C register involved in the buffer mode provides less than 30 bits, the higher order bits are always interpreted as *zero*. The initial buffer address is included in and the terminal address excluded from the storage addresses used in a buffer mode.

The individual steps in a buffer mode of operation are initiated by the entry, or removal, of a data word in the associated C register. The main program sequences are suspended by these events, and the three-sequence operation shown below is performed to complete the buffer process.

- The "*E*" *Sequence* reads the control word associated with the active C register and enters the lower order 15 bits into the R register.
- The "*F*" *Sequence* uses the address in the R register to transfer the data word in the C register to or from the magnetic-core storage.
- The "*G*" *Sequence* increases the address in the R register by one and records the result in the lower order 15 bits of the special storage location. If the upper and lower halves of this storage location are equal, the buffer mode is terminated.

If several requests for buffer action occur at one time, they are processed in the order of the C-register number with C-register 7 having the top priority. A designator holds the selected C-register number throughout the three buffer sequences. As soon as one request has been satisfied, and the designator has been cleared, a second request may become active and the corresponding C-register number entered in the designator. The main program sequences resume when the designator remains cleared on completion of a buffer sequence.

C. *ADVANCE CLOCK MODE OF OPERATION*

Magnetic-core storage address 00010 functions as a 30-bit counter, advancing one count each millisecond. This provision is made so that real-time measurements may be included in the computation. The advancing of this counter is accomplished by a buffer type operation under the control of a 1-kilocycle crystal oscillator. Once each millisecond, a control designator is set to indicate the need for advancing the clock. The main program "E" and "G" sequences are executed as if a buffer operation were being performed. These sequences advance the lower order 15 bits of (00010). If at the end of the "G" sequence (R)=0, a special "F" sequence is enabled which advances the higher order 15 bits of (00010) by one.

D. *INTERRUPT MODE OF OPERATION*

The interrupt mode of operation permits a computer program to be interrupted by an unexpected request signal from external equipment. Upon interruption by such a signal, the computer executes the instruction stored at a special storage location (00011). If this instruction were not a jump instruction, the program would then continue as if it had not been interrupted. A more usual instruction stored at address 00011 would be a return jump to address 00012. This would record the current program address in the lower order 15 bits of (00012) and commence a routine beginning at address 00013. This routine would evaluate the reason for the interruption via the C register communicating with external equipment. When completed, the routine would execute a jump instruction at address 00012 and return to the main program.

An interruption of a program and the resulting execution of the instruction at address 00011 cause the interrupt designator to be set. This disables the interrupt-request line from the external equipment and prevents a second interruption from superseding the first. Upon completion of the interrupt routine, an instruction is normally read from address 00012 to return to the main program. The process of reading an instruction, not an operand, from address 00012 causes the interrupt-request line to be enabled for further interruptions.

7. COMMUNICATION CABLES

The AN/USQ-17 (Serials 1 - 5) Computer communicates with other equipment via a number of communication lines. These communication lines are physically grouped into cables containing 20 twisted pairs of wires and terminating in 37-pin AN connectors. There is a total of 24 of these cables functionally grouped into three categories. These categories are described below as input, output, and function cables. Each wire in a cable has a function which is identical to a similar wire in any other cable of the same category; hence, any external equipment can be connected to any C register as long as the cable categories are observed. Communication registers with fewer than 30 bits transmit and receive zeros in the higher order digits. A potential of -10 volts is interpreted as a *zero* signal. A voltage potential of zero is interpreted as a *one* signal.

A. *INPUT CABLES*

Twelve input cables are provided in this computer. Two cables are associated with each of the communication registers except C-register 3 which is the function register. Each cable contains 30 information lines and 2 control lines. The information lines are twisted with one another, and the control lines are twisted with ground wires. One control line, identified as the *ready line,* signals the external equipment that the C register is ready for an input. The

other control line, identified as the *resume line,* signals the computer that the input information lines are to be sampled and the results entered in the C register.

B. *OUTPUT CABLES*

Six output cables are provided. One output cable is associated with each of the communication registers except C-register 3. Each cable contains 30 information lines and two control lines. The information lines are twisted with one another, and the control lines are twisted with ground wires. One control line, identified as the *ready line,* signals the external equipment that the C register has been filled and that the output information lines are ready to be sampled. The other control line, identified as the *resume line,* signals the computer that the information has been sampled and that it is no longer needed.

C. *FUNCTION CABLES*

Six function cables are provided. Each cable contains outputs from C-register 3 and inputs to C-register 3. The cables are numbered 1 through 6, corresponding to values of the output-channel designator and the input-channel designator which activate them. Function channels 1 and 2 have fifteen active input lines and twelve active output lines. Function channels 3, 4, 5, and 6 have six active input lines and six active output lines. Information lines are twisted with one another, and the control lines are twisted with ground wires.

Each cable contains 15 input information lines, 12 output information lines, and 2 control lines. One control line, identified as the *ready line,* signals the external equipment that C-register 3 has been filled and that the output lines are ready to be translated. The other control line, identified as the *interrupt-request line*, may be activated at any time by external equipment to interrupt the computer program. Inputs to C-register 3 are sampled at the time the input-channel selection is made.

PART 2

INPUT/OUTPUT SPECIFICATION FOR THE AN/USQ-17 COMPUTER SET AND ASSOCIATED EQUIPMENT

1. BASIC INFORMATION

The NTDS Unit Computer (AN/USQ-17) has seven input/output registers, called C registers: C^1 is a 6-bit register; C^2 is a 15-bit register; C^4, C^5, C^6, and C^7 are each 30-bit registers. The C^3 register is a special Function Register of 15 bits which is used primarily to establish switching in external equipment. All C registers communicate with external equipment in the parallel mode. Figure C1-1 illustrates the input/output section of the Unit Computer in block diagram form.

1.1 DATA CHANNELS

Each C register except C^3 has two logically independent input channels and one output channel. Each channel size equals the register size, i.e., 30 bits for C^4, C^5, C^6, and C^7; 15 bits for C^2; and 6 bits for C^1. In addition to the data lines, each channel contains two control lines called the READY line and the RESUME line. Ready signals originate only in the computer, while Resume signals always originate in the external equipment. Ready always means that the computer is ready. An input Ready means that the computer is ready to receive an input word via the selected channel; an output Ready means that the computer is ready with an output word in its selected output channel.

1.2 FUNCTION CHANNELS

The C^3 Function Register is used to select particular external equipments and their modes of operation. The C^3 register has six input and six output channels, all logically isolated. Each input channel has two control lines — the READY line and the INTERRUPT line. No RESUME lines exist for C^3. One input and one output channel, together with the control lines for these channels, appear on each connector associated with C^3.

1.3 CONTROL COMMUNICATION

The NTDS Unit Computer is designed to use a d-c level input/output system. Signals are d-c

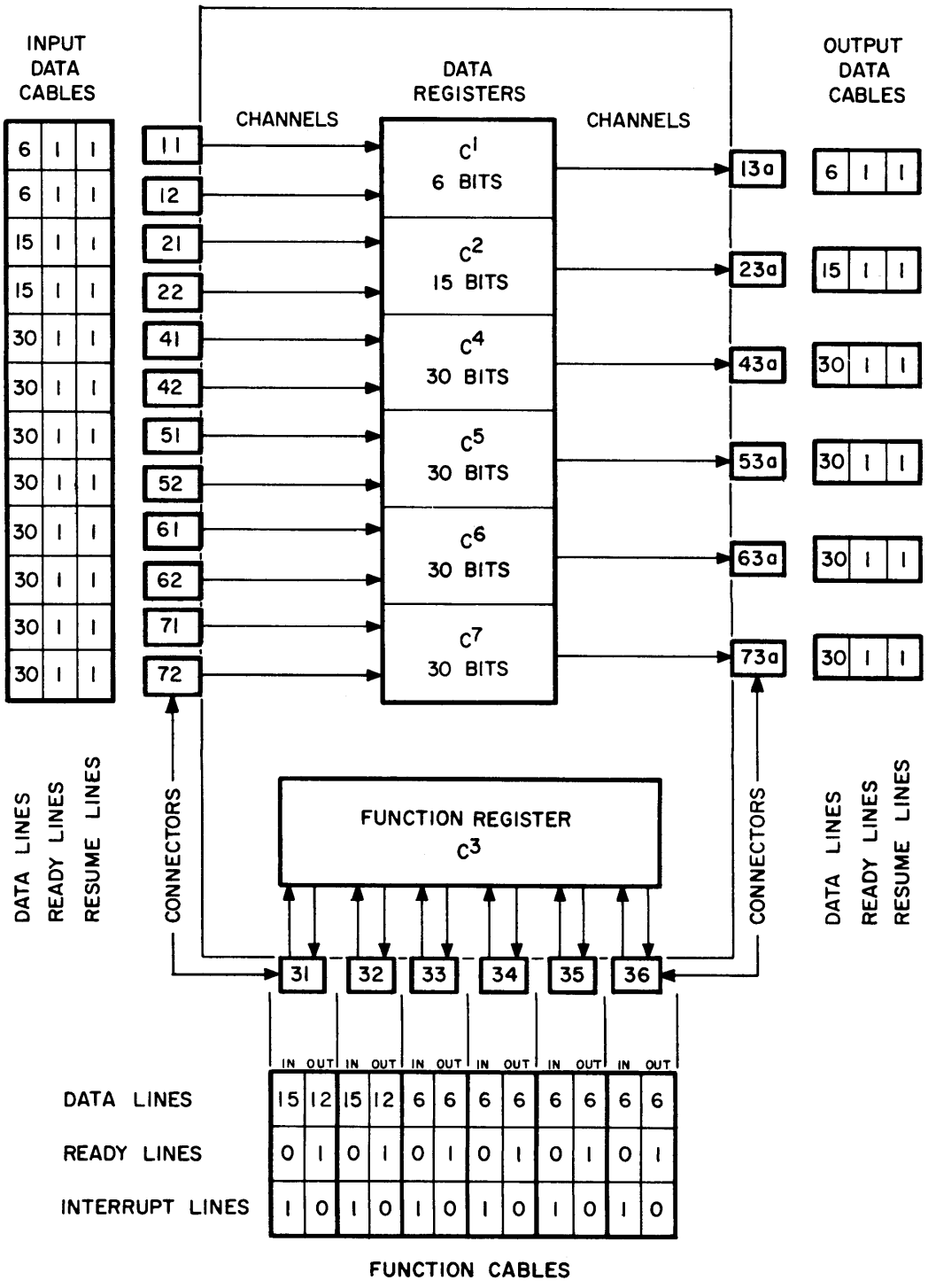


Figure C1-1. Block Diagram - Input/Output Section

levels which may be changed upon interchange of control information. Signals may exist for microseconds or days, depending on the nature of the particular task.

Note that the input/output circuits for the control lines are no different from those of the data lines. Hence, delay times, rise and fall times, and storage times are similar.

1.3.1 Data Transfer from Input Equipment to Unit Computer

Figure C1-2 is a block diagram of a computer receiving data from an input equipment.

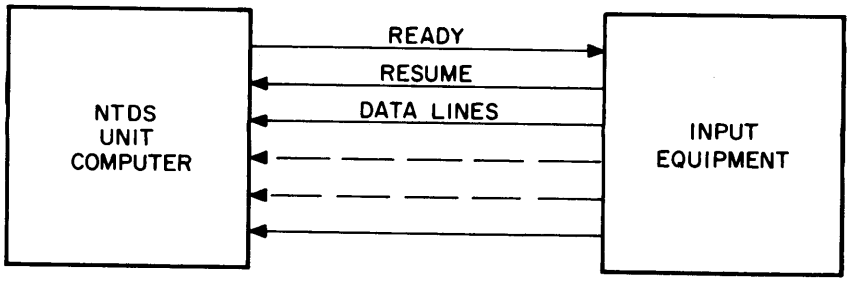


Figure C1-2. Data from Input Equipment to Computer

The computer accepts data from the input equipment as follows.

1) The computer sends a Ready signal to the equipment, signifying that it is prepared to receive a word.

2) The equipment responds by placing signals on all data lines of the cable simultaneously.

3) The Resume signal is then sent to the computer signifying that the data are available at the input terminals of the computer.

4) The computer, sensing the Resume signal, turns off its Ready signal and accepts the incoming word.

5) The external equipment, upon sensing termination of the Ready, drops its Resume.

6) When the computer (under program control) has recognized the word, the Ready line may again be energized, signifying that the computer is prepared to receive another word from the equipment.

(The same information is shown in a timing diagram in Figure C1-4.)

1.3.2 Data Transfer from Unit Computer to Output Equipment

The following illustration (Figure C1-3) is a block diagram of the computer transferring data to an output equipment.

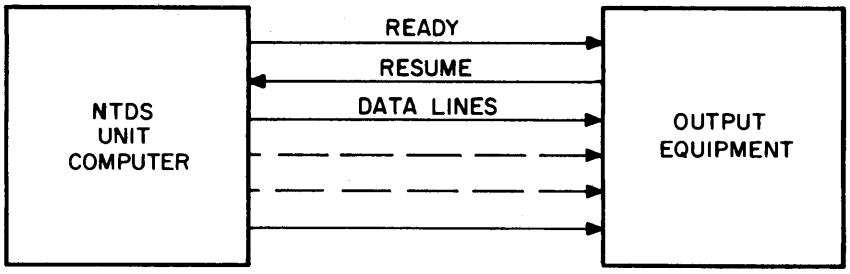


Figure C1-3. Data from Computer to Output Equipment

In transferring data from the computer, the following sequence is used.

1) The computer program places an output word in a C register. All information lines of the C register are energized simultaneously.

2) The computer then sends a Ready signal, which means that the output data are available on the data lines.

3) The output equipment accepts the incoming word at its own recognition rate and sends a Resume signal to the computer.

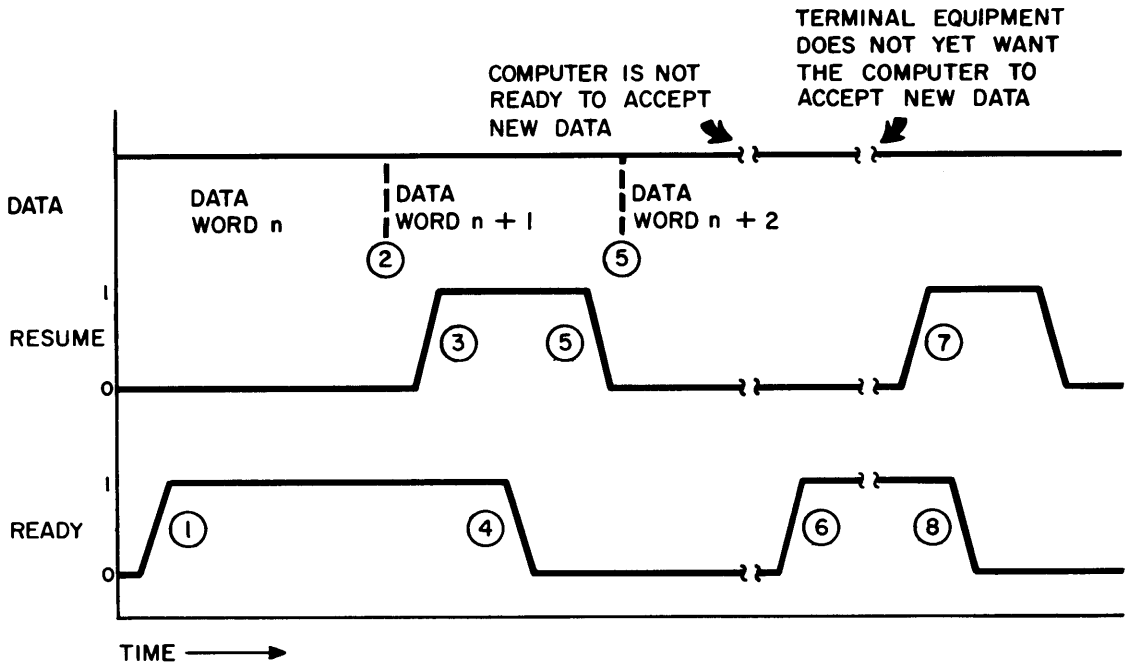
4) The computer senses the Resume signal and turns off its Ready signal.

5) Upon termination of the Ready signal, the output equipment terminates its Resume signal.

(A timing diagram showing how the Unit Computer communicates with an output equipment is shown in Figure C1-5.)

1.3.3 Interrupt

Input channels to C^3 contain Interrupt control lines. When an Interrupt line is energized, the



(1) Computer energizes Input Ready line.

(2) Terminal equipment changes data.

(3) Computer senses that the Resume signal has gone from *zero* to *one* and gates data to input register.

(4) Computer drops Ready signal 2.5 μsec after sensing Resume signal.

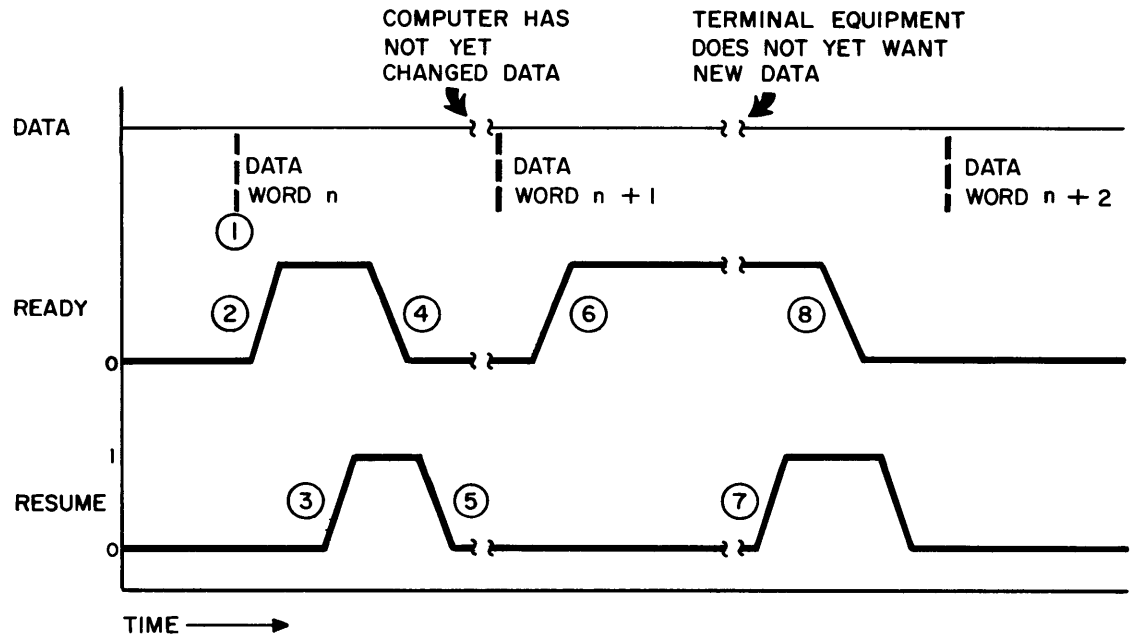
(5) Terminal equipment senses that the Ready signal has gone from *one* to *zero* and drops the Resume signal. Terminal equipment may also change data at this time.

(6) Computer has gated data out of input register and is ready to accept data as in 1.

(7) Same as 3.

(8) Same as 4.

Figure C1-4. Input Sequence



① Computer changes data.

② Computer energizes its Output Ready line.

③ Terminal equipment has sampled data and sends Resume signal to the computer.

④ Computer senses that Resume signal has gone from *zero* to *one* and drops the Ready signal in 2.5 μsec.

⑤ Terminal equipment drops Resume signal after computer drops the Ready signal.

⑥ Same as 2.

⑦ Same as 3.

⑧ Same as 4.

Figure C1-5. Output Sequence

computer program is interrupted, and a special subroutine is entered to determine the interrupting channel and the cause for interruption, to take action, and to return to the main program.

1.3.4 Timing

When transmitting data from the computer to an external equipment, the data lines must be stable before being sampled. A fixed time delay of 1.5 microseconds exists between the instant the computer loads an output register and the instant the output Ready signal is energized. This delay is insufficient to insure that the data lines are stabilized. It is recommended that additional delay (of at least 3 microseconds where 030B output amplifiers are used in the computer output channel or 10 microseconds where 030 output amplifiers are used, see Section 4.2) be inserted in external equipment between detection of the output Ready and sampling of the data lines. An alternate method is to select output amplifiers such that the amplifier used for the Ready is slower than those used for any of the data bits.

An analogous situation exists for computer input. A fixed 3-microsecond delay exists between the time the computer receives the Resume signal and the time the computer samples the input data. This delay is not sufficient to ensure that the data lines are stable before being sampled by the computer if the external equipment resumes simultaneously with placing data on the line. For this reason, it is necessary to incorporate in each input equipment a delay of at least 4.5 microseconds between the time data are placed on the lines and the time the Resume signal is initiated, and to be sure that variations in output amplifier response do not reduce this time difference.

2. DESCRIPTION OF INPUT/OUTPUT SIGNALS

2.1 GENERAL

2.1.1 This description applies to all devices which connect with the Unit Computer.

2.1.2 This description is written to allow minimum data transfer time consistent with good engineering practice and moderate hardware requirements.

2.1.3 The binary *zero* and *one* voltage levels shall be measured at computer and peripheral equipment output terminals with currents specified below.

2.1.4 The d-c resistance of the ground return for a cable shall not exceed 0.5 ohm.

2.1.5 For the purpose of this description, an output circuit shall be any circuit that applies data or control information (i.e., Ready, Resume, Function Code, etc.) to an inter-communication cable.

An input circuit shall be any circuit that receives data or control information from an inter-communication cable.

An output circuit located in the Unit Computer will be connected to an input circuit located in the peripheral equipment. An input circuit located in the Unit Computer will be connected to an output circuit located in the peripheral equipment.

2.2 OUTPUT CIRCUITS

2.2.1 The binary *one* state of an output amplifier shall be 0v ± 1.5v at the terminals of the equipment under all conditions.

2.2.2 The binary *zero* state of an output amplifier shall be -12.5v ± 2.5v at the terminals of the equipment under all conditions.

2.2.3 In the binary *one* or *zero* state, an output circuit (located in the Unit Computer or peripheral equipment) need supply no more than 1 ma to any one input circuit in another equipment.

2.2.4 An output circuit must be capable of driving up to ten input circuits in parallel.

2.2.5 The total wiring capacitance an output circuit must drive may vary from 0 to 9000 micromicrofarads.

2.3 CHARACTERISTICS OF OUTPUT CIRCUITS

2.3.1 Data applied to a cable may not be changed once a computer output Ready signal or a computer input Resume signal has been set until an appropriate control signal has been received.

2.3.2 A computer output Ready signal indicates to the peripheral equipment that data have already been placed on the lines. Computer output Ready shall be delayed so that 10 per cent amplitude of the computer output Ready occurs 1.5 microseconds after 10 per cent amplitude of the earliest data signal.

2.3.3 An input Resume indicates to the computer that the external equipment has placed data on the lines and that sufficient time has elapsed for the data lines to become stabilized. Input

Resume signals shall be delayed so that 10 per cent amplitude of the input Resume occurs at least 4.5 microseconds after 10 per cent amplitude of the earliest data signal. This time relationship shall be maintained at the computer input terminals.

2.3.4 The input Resume and output Resume signals associated with the Unit Computer C-registers 1, 2, 4, 5, 6, and 7 must have the following characteristics: (Note that C^3 has output Readys only and has no Resumes.)

2.3.4.1 A Resume signal, once having been changed from *zero* to *one* shall be returned to *zero* within 20 microseconds from the time the Ready signal in the same cable changes from *one* to *zero*.

2.3.4.2 A Resume signal, once having been changed from *one* to *zero* shall not be returned to *one* until the Ready signal in the same cable changes from *zero* to *one*.

2.3.5 An Interrupt signal originates in the peripheral equipment as a signal on the Interrupt line of a function cable. The Interrupt signal commands the computer to enter a specific Interrupt routine. An Interrupt signal, once having been changed from *zero* to *one* shall not be returned to *zero* until the computer, as part of its Interrupt routine, sends a function word via the C^3 register to the interrupting equipment commanding the equipment to terminate the Interrupt signal.

2.4 INPUT CIRCUITS

2.4.1 The maximum steady-state current drawn from a line by an input circuit shall not exceed 1 ma.

2.4.2 The input circuit shall be such that if the input wire is disconnected, the effect will be as though a *zero* were present at the input.

2.4.3 The threshold level distinguishing the *one* state from the *zero* state shall be -6v $\pm$ 1v at the input terminals.

2.4.4 The input circuit shall provide an integration delay of 1.5 microseconds $\pm$ 0.5 microseconds to a step function of 15v applied to the input.

2.4.5 Phase relations between data signals and between data and computer output Ready and input Resume signals shall be preserved through input circuits connected to the same cable except as they are affected by the tolerance allowed for integration delay.

2.4.6 All external equipments must adequately resynchronize all control signals, i.e., signals other than data.

2.5 SYSTEM REQUIREMENTS

2.5.1 The sum of the lengths of all cables connected to an output circuit shall not exceed 300 feet.

2.5.2 Voltages on data lines must be stable before an input circuit recognizes the associated computer output Ready or input Resume signal.

2.5.3 Only an output Ready line is available from each output channel of C^3 in addition to data lines. Other Ready and Resume information must be obtained from other sources for function requests and interrupts.

3. CONNECTOR AND TABLE OF CONNECTIONS

3.1 DESCRIPTION OF CONNECTOR

A 37-pin connector is to be used and has the following "AN" identification:

PLUG - AN 3102 A-28-21P (C)
SOCKET - AN 3108 B-28-21S (C)

3.2 PIN AND CONDUCTOR IDENTIFICATION

Table C1-2 identifies the relationships among conductors, connector pins, and function performed. Each conductor is identified by a color-coding scheme. The colors are identified by a two-digit number in the first column. The table also shows the proper connections for computer data output cables, computer data input cables, and computer function cables.

4. CIRCUIT DIAGRAMS

4.1 INPUT CIRCUIT

Figure C1-6 is a circuit diagram of the input amplifier used in the Unit Computer. Resistor R1 and the -15v supply produce the effect of a *zero* at the input when the input line is disconnected. The combination of R2 and C1 forms an integrating delay which prevents the input circuit from recognizing a short duration pulse, such as may be experienced from noise or crosstalk.

TABLE C1-2. PIN AND CONDUCTOR IDENTIFICATION

CONDUCTOR COLOR IDENTIFICATION	CONNECTOR PIN	COMPUTER DATA OUTPUT CABLE	COMPUTER DATA INPUT CABLE	COMPUTER FUNCTION CABLE
01 02	A B	Resume Resume Shield	Ready Ready Shield	Interrupt Int. Shield
03 04	C D	Ready Ready Shield	Resume Resume Shield	Ready Ready Shield
05 06	E F	Bit 0 1	Bit 0 1	In Bit 0 In Bit 1
07 08	G H	2 3	2 3	In Bit 2 In Bit 3
09 10	J K	4 5	4 5	In Bit 4 In Bit 5
11 12	L M	6 7	6 7	In Bit 6 In Bit 7
13 14	N P	8 9	8 9	In Bit 8 In Bit 9
15 16	R S	10 11	10 11	In Bit 10 In Bit 11
17 18	T U	12 13	12 13	In Bit 12 In Bit 13
19 20	V W	14 15	14 15	In Bit 14 Out Bit 0
21 22	X Z	16 17	16 17	Out Bit 1 Out Bit 2
23 24	a b	18 19	18 19	Out Bit 3 Out Bit 4
25 26	c d	20 21	20 21	Out Bit 5 Out Bit 6
27 28	e f	22 23	22 23	Out Bit 7 Out Bit 8
29 30	g h	24 25	24 25	Out Bit 9 Out Bit 10
31 32	j k	26 27	26 27	Out Bit 11 Not Used
33 34	l n	28 29	28 29	Not Used Not Used
	p r			
	s	Cable Shield	Cable Shield	Cable Shield

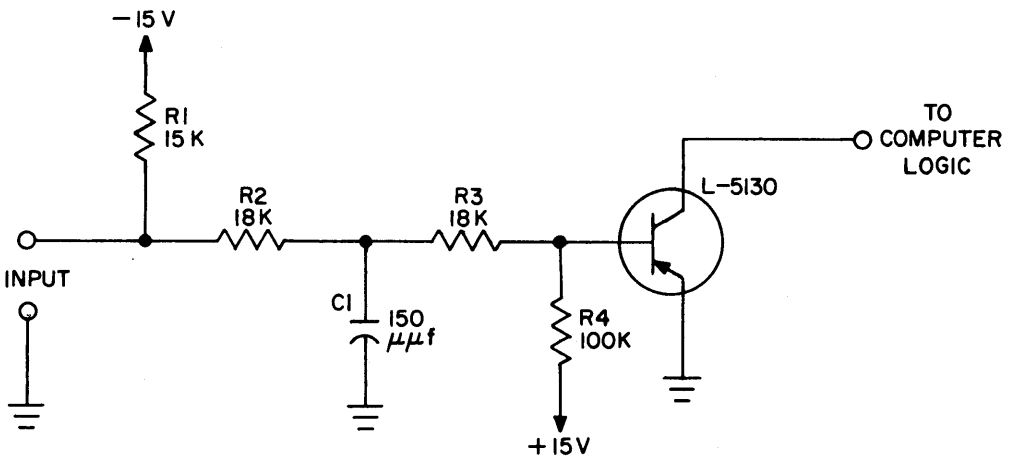


Figure C1-6. Circuit Diagram - Input Amplifier

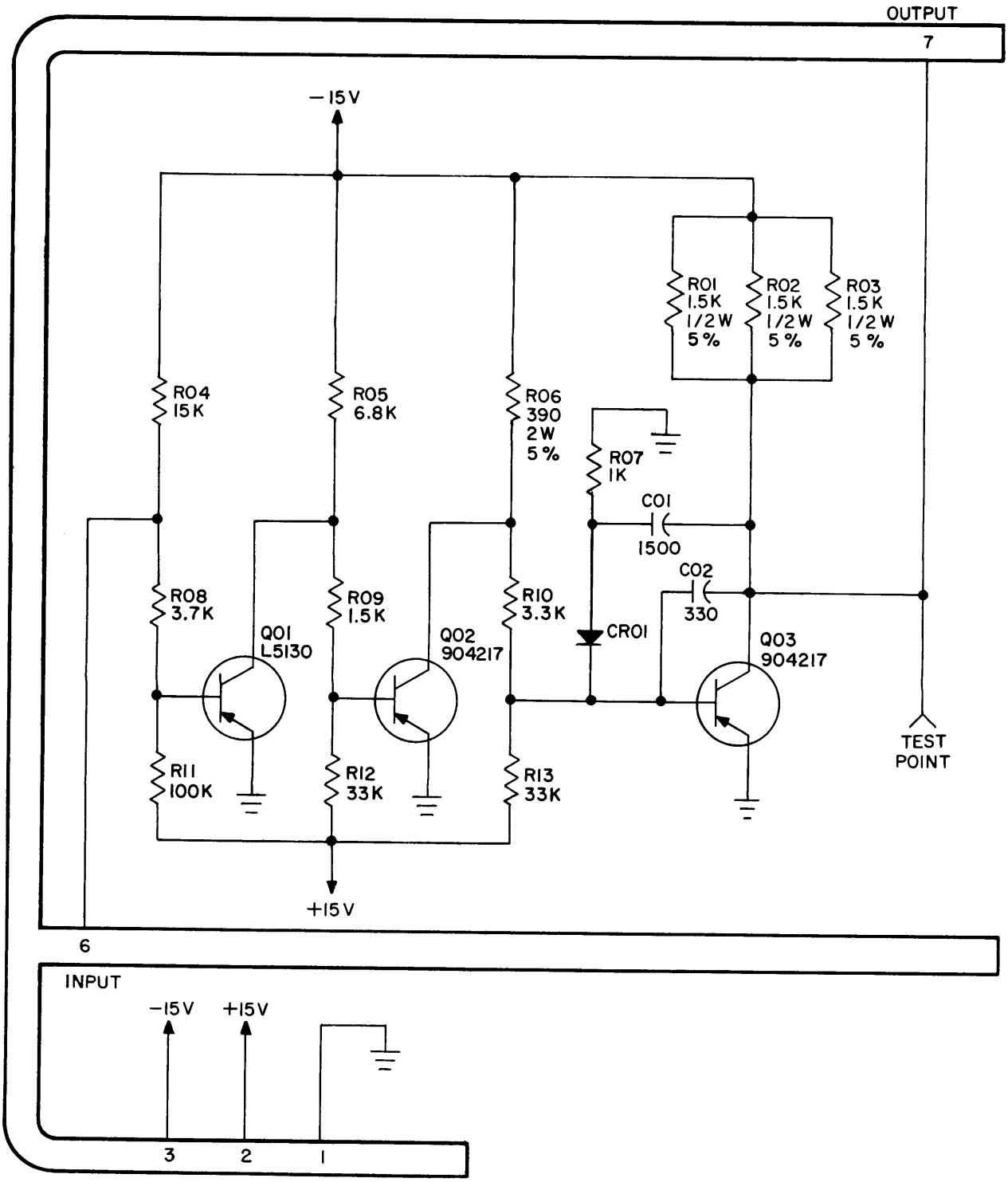
The transistor collector connects directly to an input of a Unit Computer logic circuit.

4.2 OUTPUT CIRCUITS

Three types of output amplifiers may be encountered in the experimental model Unit Computers. The inputs to all types are the same (the collector of an L-5130 transistor acts essentially as a switch giving open circuit and short circuit conditions) and the output voltage levels are the same, but the circuits are fundamentally different.

Type 030 output card (Figure C1-7) was the original output amplifier and is a simple transistor switch. It was intended for use with low data rate devices (50 - 100 microsecond region). Considerable variations in response speed will be found from card to card. Exact specifications for rise and fall times do not exist for this card. Considerable extra output Ready delay (10 μsec) must be used in peripheral equipments operated from this card to allow for variations in response speed. For these reasons, the 030 card is unsuitable for use with high data rate equipments.

Type 030A output card (Figure C1-8) was an interim attempt to overcome some of the difficulties of the 030. The 030A card uses complementary symmetry output transistors and has a very fast switching time, substantially less than one microsecond. However, the high frequencies generated by rapid transition will give crosstalk problems on the output lines if cables

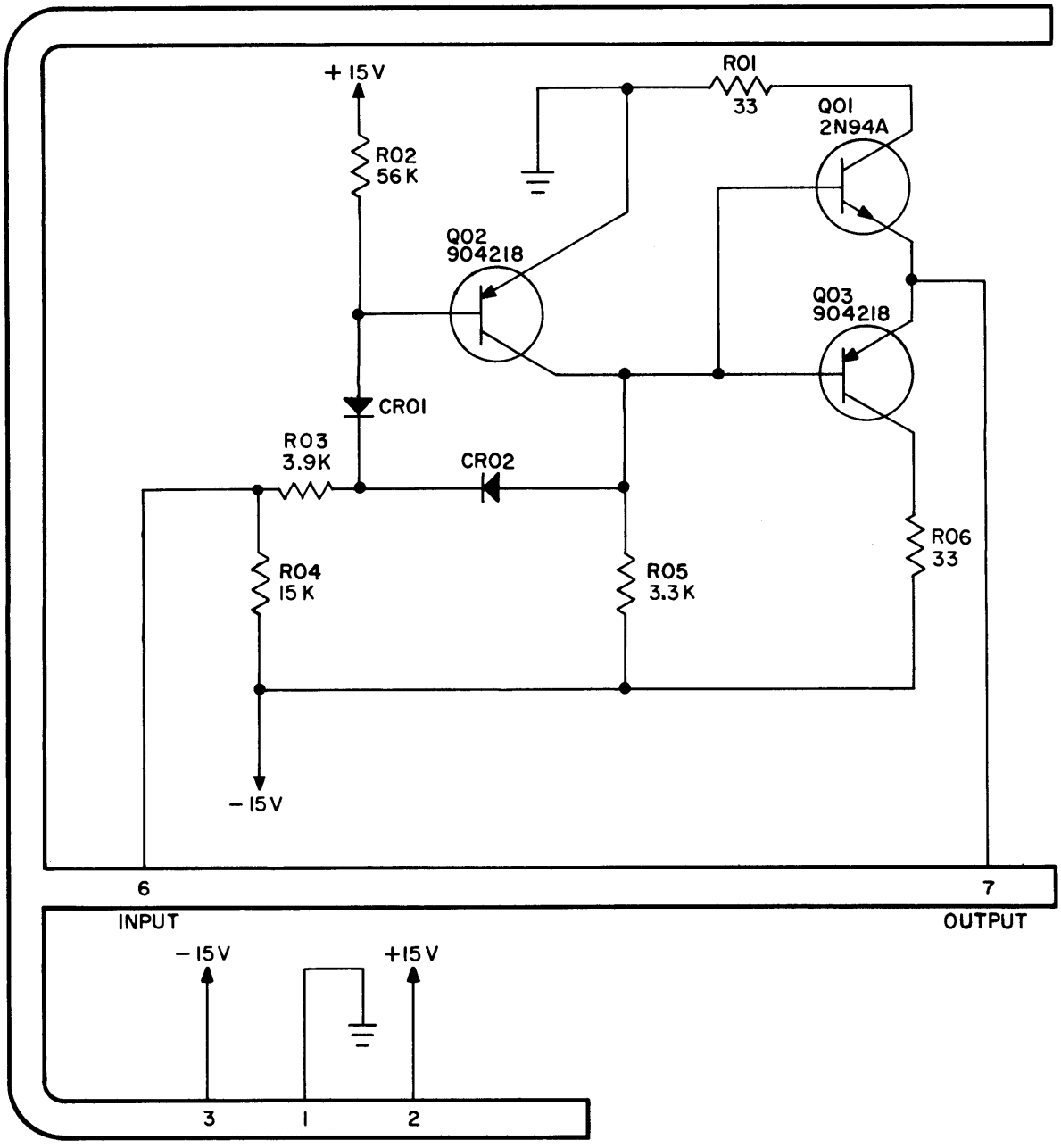


NOTE:

UNLESS OTHERWISE INDICATED, ALL DIODES ARE REMINGTON RAND UNIVAC PART NUMBER 907801 (—▶|—)

CATHODE

Figure C1-7. Schematic Diagram - Output Amplifier Type 030 Card



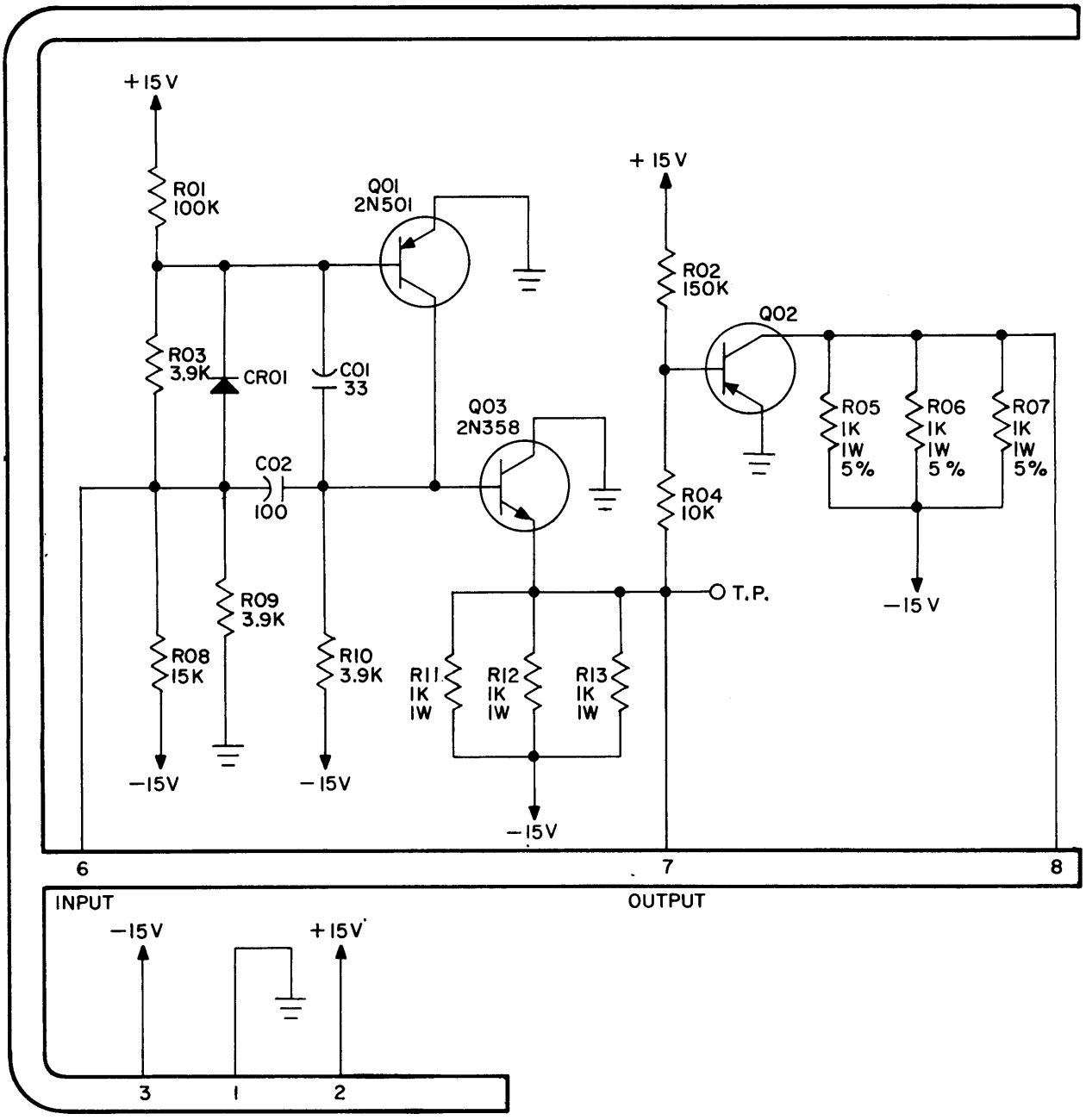
NOTE:
UNLESS OTHERWISE INDICATED, ALL DIODES ARE REMINGTON RAND UNIVAC PART NUMBER 907801 (—▶|—)
CATHODE

Figure C1-8. Schematic Diagram - Output Amplifier Type 030A Card

longer than 30 feet are used. If difficulty is experienced using 030A cards, it is recommended that the following check be performed.

> Perform an output operation with a word of all *ones* and monitor the output Ready line with an oscilloscope. A noise pulse will be seen on the output Ready as the 30 ones hit the data lines. If this pulse approaches the triggering voltage, there is danger that a false early output Ready will be received.

Type 030B output card (Figure C1-9) was the first attempt to design a card to communicate over long cables (up to 300 feet) at high data rates. The first stage (associated with transistor Q01) employs two capacitors connected between collector and base to give negative feedback and control the transition time. Switching time is held between 3 and 6 microseconds with capacitive loading varying from 0 to 9000 $\mu\mu$f. The output stage (associated with Q03) uses a grounded collector NPN transistor. The purpose of the circuit associated with Q02 is to provide constant power supply loading regardless of the output condition. Tests have demonstrated that the 030B card, when used with the proper output Ready delay (see section 1.3.4), is capable of communication over cables up to 300 feet in length. However, rigorous evaluation shows that operation is marginal with maximum length of the commonly used cable. It is recommended that cables not over 150 feet be used if high data rates are desired.



NOTE:
UNLESS OTHERWISE INDICATED, ALL DIODES ARE REMINGTON RAND UNIVAC PART NUMBER 907801 (—▶|—)
CATHODE

Figure C1-9. Schematic Diagram - Output Amplifier Type 030B Card

SECTION C2

CHARACTERISTICS OF THE

AN/USQ-17 (SERIAL 6) UNIT COMPUTER

CHARACTERISTICS OF THE AN/USQ-17 (SERIAL 6) UNIT COMPUTER

1. BASIC INFORMATION

The instruction repertoire for the AN/USQ-17, Serial 6 Unit Computer is similar to that of the AN/USQ-17, Serials 1 through 5; therefore, this section deals only with instructions and registers which differ from those described in section C1. Basic changes exist in C-registers 1, 2, and 3 and External Function codes 73 and 74. Special addresses ranging from 00000 and 00031 have new assignments. Interrupts, both internal and external, require special attention when coding the Serial 6.

The following pages present significant changes of the Serial 6 from Serials 1 through 5. The programmer must thoroughly familiarize himself with deviations given herein before coding the Serial 6 computer. Figure C2-1 shows the entire panel of the maintenance console for the Serial 6 computer; Figure C2-2 is a larger view of the left-hand section of the panel.

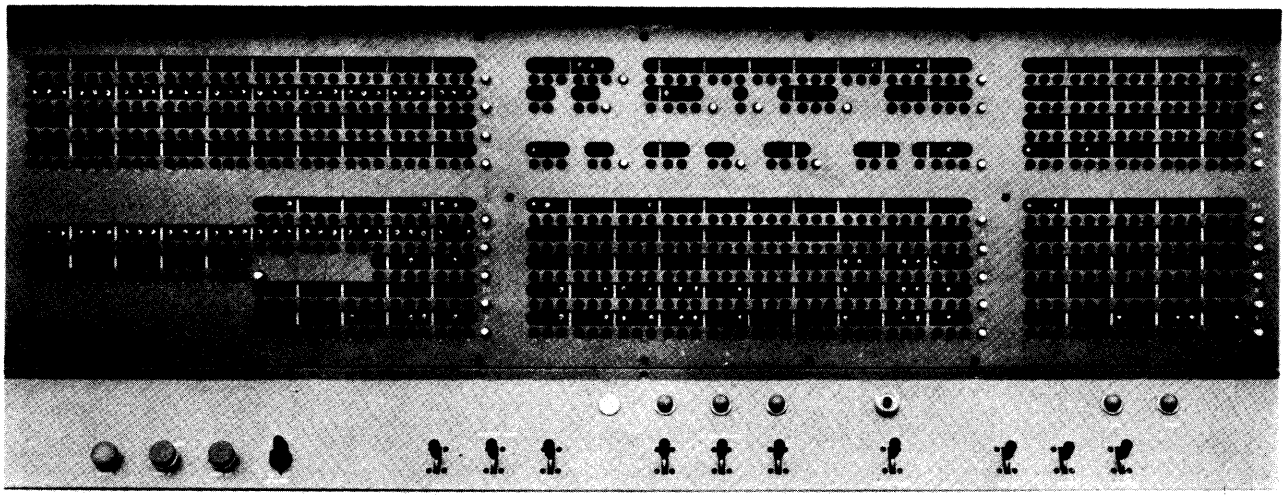


Figure C2-1. Maintenance-Console Panel for Serial 6

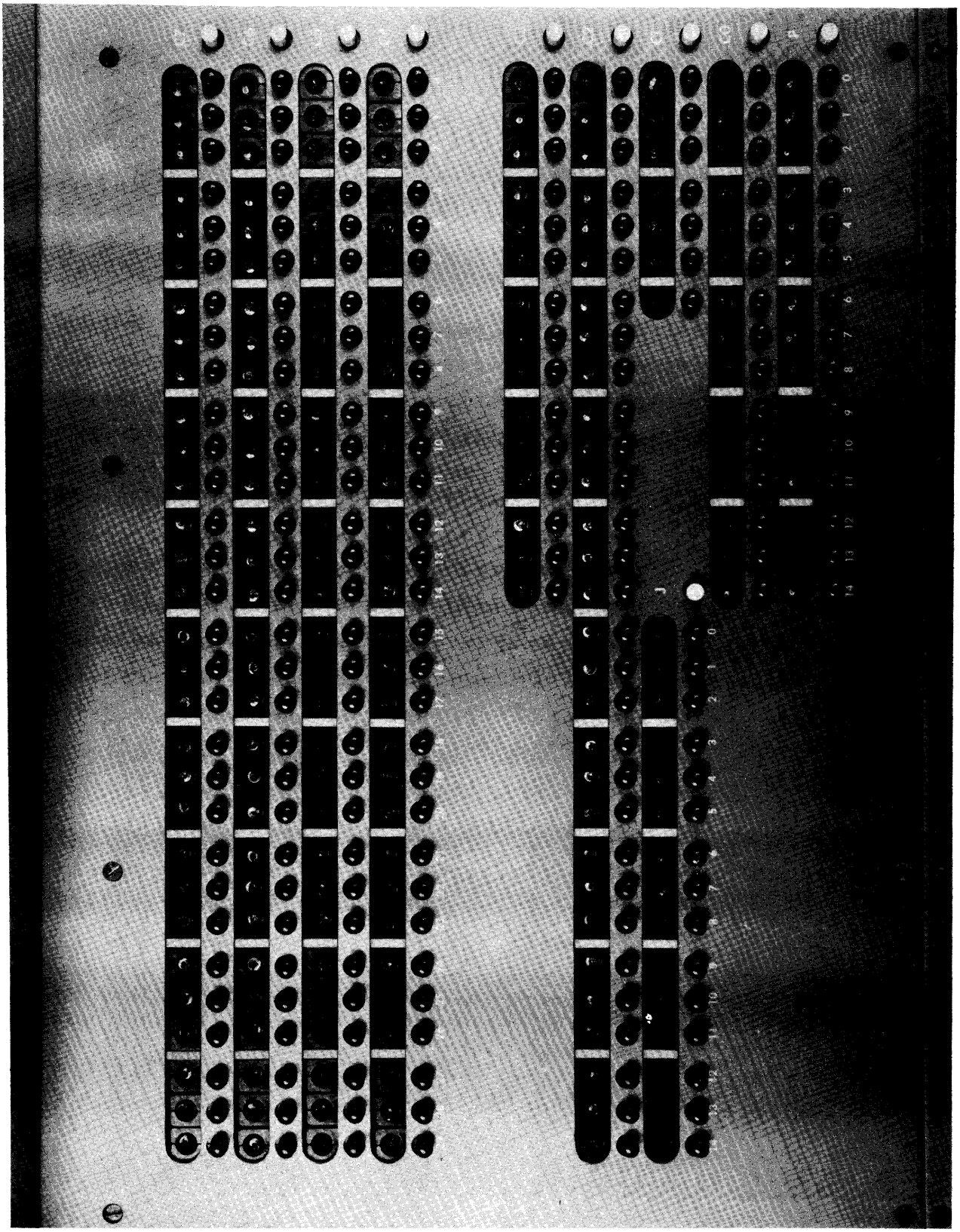


Figure C2-2. Left-Hand Section of Maintenance Console Panel

2. DESIGN CHANGES IN AN/USQ-17 SERIAL 6 UNIT COMPUTER

A. *C REGISTERS AND INSTRUCTIONS*

(1) *C-Register* 1

The C^1 communication register now consists of seven bits. This allows C^1 to be used with external equipment utilizing 7-level media.

(2) *C-Register* 2

The C^2 communication register now consists of 30 bits. There are now five C registers capable of transferring full-length words.

(3) *C-Register* 3

The function register remains the same in all respects except the interrupt is now called the External-Interrupt request. Each input function channel has a separate *interrupt-request* line. Requests are honored according to a priority system (Channel 6 down to Channel 1). Each channel's interrupt-request line is assigned a unique fixed address - 00011 for channel 1, 00012 for channel 2, etc. The exit address is 00017. The interrupt feature is described in detail later.

(4) *Instruction* 73

Function code value 73, vacated by deletion of the transfer mode, has been assigned to a new instruction: Select Channel. This instruction sets the $\mathbf{c}^j$ designator in C^0 to the value of the lowest two or three bits of the operand (Y). Because the **j** designator specifies which channel-selection designator is to be set, it is not available for the normal skip function.

The value **j** = 0 has no effect within the computer other than to take time.

Instructions 13 and 17 with **j** = 0 (Enter C^0 and Store C^0) are not changed.

(5) *Instruction* 74

Function code value 74, vacated by deletion of the transfer mode, has been assigned to a new instruction: Initiate Buffer Monitor. This instruction establishes a monitor of the buffer mode on C-register **j**. This is the only step directly taken by this instruction. It is implemented by setting the buffer-monitor designator, $\mathbf{a}^j$, for the desired C register. Hence the **j** designator does not have its normal skip function. The value **j** = 3 is without effect since there is no $\mathbf{a}^3$ designator. The **k**-, **b**-, and **y**-designator values are irrelevant in this instruction.

The monitor established by Instruction 74 has no effect as long as a buffer mode is active on the specified C register. As soon as the buffer is terminated, an Internal-Interrupt request is generated. If the C register is not in the buffer mode when Instruction 74 is executed, the Internal-Interrupt request is generated during the next sequential instruction. An Internal-Interrupt request acts similarly to the External-Interrupt request. Both types are further discussed below. A given buffer-monitor designator is cleared only after the resulting Internal-Interrupt request has been acknowledged by the computer.

B. *DESIGNATORS*

The **z** and **t** designators were deleted in the redesign of the buffer mode and elimination of the transfer mode. The **i** designator was removed in the redesign of the interrupt mode. Several new designators have been added to the computer console. These are described briefly in the following paragraphs.

(1) *Buffer Monitor Designators* - $\mathbf{a}^1$, $\mathbf{a}^2$, $\mathbf{a}^4$, $\mathbf{a}^5$, $\mathbf{a}^6$, and $\mathbf{a}^7$ (one bit each)

Instruction 74 sets $\mathbf{a}^j$ to *one* when it is desired to monitor a buffer mode via C-register j. When $\mathbf{a}^j$ is *one*, termination of a buffer mode via C-register j produces an Internal-Interrupt request that initiates an interrupt program beginning at address 0002j. The values of each buffer-monitor designator are interpreted as follows:

a = 0 Buffer monitor inactive

a = 1 Buffer monitor active

(2) *E Sequence Status Designators* - $\mathbf{e}^1$ and $\mathbf{e}^2$ (one bit each)

The E sequence, which performs an entire buffer or advance-clock operation, is divided into two parts, E^1 and E^2. Each part has a corresponding status designator. The interpretation of the values of these designators is as follows:

$\mathbf{e}^1 = 0$ E^1 sequence inactive

$\mathbf{e}^1 = 1$ E^1 sequence active

$\mathbf{e}^2 = 0$ E^2 sequence inactive

$\mathbf{e}^2 = 1$ E^2 sequence active

(3) *Interrupt Status Designator* - **v** (two bits)

This designator is used to indicate whether an interrupt request has been acknowledged (the request may be internal or external) and whether the interrupt mode is active. Interpretation

of the values of **v** is as follows:

v = 0 No request and interrupt mode inactive if **w** = 0

External request acknowledged and interrupt mode inactive if **w** ≠ 0

v = 1 Internal request acknowledged and interrupt mode inactive

v = 2 Interrupt mode active

v = 3 Not used

(4) *Interrupt Reference Designator* - **w** (three bits)

This designator is used to indicate the source of an interrupt request. Interpretation of the values of **w** is as follows:

w = 0 No interrupt requests

w = 1 External request from channel 1 or internal request for C^1

w = 2 External request from channel 2 or internal request for C^2

w = 3 External request from channel 3

w = 4 External request from channel 4 or internal request for C^4

w = 5 External request from channel 5 or internal request for C^5

w = 6 External request from channel 6 or internal request for C^6

w = 7 Internal request for C^7

C. *INTERRUPT MODE*

The interrupt mode permits the main program to be interrupted by either a request signal from external equipment or by a request generated within the computer itself. The former is called *External-Interrupt request* and the latter is called *Internal-Interrupt request.* An Internal-Interrupt request is generated when 1) a monitor for a buffer mode via C-register j was established by execution of instruction 74 and 2) the buffer mode via C-register is inactive.

Neither an External nor an Internal-Interrupt request is acted upon when any of the following conditions exist: f = 70, r ≠ 0, or g = 1. An Internal-Interrupt request is acted upon only when no External-Interrupt request is present.

There are six External-Interrupt-request channels numbered 1 through 6. An Internal-Interrupt request is associated with C-registers 1, 2, 4, 5, 6, and 7. Table C2-1 shows that a

TABLE C2-1. ALLOCATION OF SPECIAL ADDRESSES

Address	Special Storage Function
00000	Initial Starting Address from Master Clear
00001	Buffer Control Register for C^1
00002	Buffer Control Register for C^2
00003	Buffer Control Register for C^3
00004	Buffer Control Register for C^4
00005	Buffer Control Register for C^5
00006	Buffer Control Register for C^6
00007	Buffer Control Register for C^7
00010	Real-Time Clock Register
00011	External-Interrupt Entrance Register for Channel 1
00012	External-Interrupt Entrance Register for Channel 2
00013	External-Interrupt Entrance Register for Channel 3
00014	External-Interrupt Entrance Register for Channel 4
00015	External-Interrupt Entrance Register for Channel 5
00016	External-Interrupt Entrance Register for Channel 6
00017	External-Interrupt Exit Register
00020	External-Interrupt Nullification Register
00021	Internal-Interrupt Entrance Register for C^1
00022	Internal-Interrupt Entrance Register for C^2
00024	Internal-Interrupt Entrance Register for C^4
00025	Internal-Interrupt Entrance Register for C^5
00026	Internal-Interrupt Entrance Register for C^6
00027	Internal-Interrupt Entrance Register for C^7
00030	Internal-Interrupt Exit Register
00031	Internal-Interrupt Nullification Register

separate entrance address is reserved for each *External-Interrupt* channel. Likewise, a separate entrance address is reserved for Internal-Interrupt requests associated with each of the C registers mentioned above. Reservation of a separate entrance address for each source of an interrupt request provides the interrupt program with an unambiguous indication of the source of the request.

A priority system exists among the six External-Interrupt-request channels. Channel 6 has the highest priority and channel 1 has the lowest. This system operates in such a manner that, for example, a request on channel 4 is recognized and acted upon only when no request is present on channels 5 or 6. As a consequence, it is necessary for an interrupt-request signal to be held on the lines until it is recognized.

As indicated in Table C2-1, each external channel is associated with a special address in memory; thus channel 4 is associated with address 00014. When an interrupt request on channel 4 is recognized, (S) is set to 00014. If this request is not to be ignored, address 00014 contains a return jump to address N. Execution of this instruction stores the current (P) which is the address of the next instruction in the main program at address N. At address N+1 the routine for handling requests from channel 4 commences. The initial instructions of this routine will probably store (A), enter the content of the lower half of address N in A, and finally store (A) in the lower half of address 00017. This action prepares for the exit from the External-Interrupt mode. Address 00017 is the exit address for all External-Interrupt-response routines. It is only upon execution of an instruction at this address that the External-Interrupt mode is completed and additional requests can be recognized. If it is desired to ignore an interrupt request on a given channel, for example, channel 4, then address 00014 is simply loaded with a return jump to address 00017 which should always contain an unconditional jump. As a consequence, the lower half of address 00017 is loaded with the address of the next instruction in the main program. The instruction at special address 00020, which normally contains an unconditional jump to address 00017, is now executed. Subsequent execution of the unconditional jump instruction at address 00017 causes resumption of the main program and terminates the External-Interrupt mode.

There are six sources of Internal-Interrupt requests, one associated with each of the C registers which can be usefully employed in the buffer mode. Thus, no Internal-Interrupt request is associated with C^3. As given in Table C2-1, each source of an Internal-Interrupt request is associated with a special address in memory. Operation of the Internal-Interrupt mode will be examined by means of an example. Suppose that a buffer mode is active on C-

register 2 and that Instruction 74 (see description above) has been executed with $j = 2$ to establish a monitor for this buffer mode. When the buffer mode via C-register 2 is terminated, an Internal-Interrupt request is produced.

A priority system governs recognition of Internal-Interrupt requests. Requests associated with C-register 7 have highest priority, those associated with C-register 6 next highest, etc; thus, the request for C-register 2 is acted upon only if there are no External-Interrupt requests present and if no Internal-Interrupt requests are present because of a completed buffer mode for registers C^7, C^6, C^5, and C^4.

When conditions for recognition of this Internal-Interrupt request for register C^2 are satisfied, (S) is set to 00022. If this request is not to be ignored, then address 00022 contains a return jump instruction to address F which begins the routine for handling requests for C-register 2. Initial instructions of this routine will probably store (A), enter the content of the lower half of address F in A, and finally store (A) in the lower half of address 00030. Address 00030 is the exit address for all Internal-Interrupt-response routines. An instruction must be read from this address in order to terminate the Internal-Interrupt mode and to allow additional interrupt requests to be recognized. If it is desired to ignore an Internal-Interrupt request, then address 00022, in the case of this example, is loaded with a return jump to address 00030. Effect of this is to place the address of the next main program instruction in the lower half of address 00030. The instruction at special address 00031 is then executed. Address 00031 normally contains an unconditional jump to address 00030. Subsequent execution of the unconditional jump instruction at address 00030 causes resumption of the main program and terminates the interrupt mode.

SECTION C3

~~AN/USQ-20~~ AN/CP-642

UNIT COMPUTER CHARACTERISTICS

AN/USQ-20 UNIT COMPUTER CHARACTERISTICS

PART 1

GENERAL DESCRIPTION

1. BASIC INFORMATION

The AN/USQ-20 Unit Computer, brain of the Naval Tactical Data System, is a general-purpose, stored-program machine capable of processing very rapidly a large quantity of complex data. Major features of the AN/USQ-20 Unit Computer include the following:

1) *Internal high-speed storage with a cycle time of 8 microseconds and a capacity of 32,768 words (16,384 optional);*

2) *Repertoire of 62 instructions, most of which provide for conditional program branches;*

3) *Average instruction execution time of 13 microseconds;*

4) *30-bit word length;*

5) *Optional operation with 15-bit half-words;*

6) *Internally stored program;*

7) *Programmed checking of data parity;*

8) *Parallel, ones' complement, subtractive arithmetic;*

9) *Single-address instructions with provision for address modification via seven index registers;*

10) *Internal 7-day real-time clock for initiating operations at desired times;*

11) *12 input and 12 output channels for rapid data exchanges with external equipment without program attention;*

12) *2 input and 2 output channels for intercomputer data transfer;*

13) *16-word wired auxiliary memory, for storage of critical instructions and constants, which provides facility for Automatic Recovery in event of program failure and for automatic initial loading of programs.*

The AN/USQ-20 Unit Computer emphasizes rapid communication with external devices and large, randomly accessible internal storage.

Single-address instructions are employed and have an average execution time of 13 microseconds. Instruction words are 30 bits; data words can be either 15 or 30 bits.

Internal storage of the Unit Computer consists of a 32,768-word ferrite core memory. Each word may be interpreted as a single 30-bit word, or as two 15-bit words individually addressed. Control, Arithmetic, and Input/Output sections of the computer each have access to the Storage section. A complete cycle for storage of a 30-bit word received from one of the other sections requires eight microseconds.

Arithmetic and logical operations are performed in the parallel binary mode. In most instances, the result of an operation appears in a 30-bit accumulator register. Arithmetic is ones' complement subtractive with a modulus $(2^{30}-1)$.

Computer operation is controlled by a stored program capable of self-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. The next sequential program step may be skipped; it is under control of the content of the Accumulator or the Q register.

Communication between the AN/USQ-20 Unit Computer and its associated external equipment is normally handled by a block transfer of data, with timing under control of the external device. 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 response 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 channel consists of 30 parallel lines. Two input and two output (special) channels are reserved for communication 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 the function desired of the external device. An External Function Word to a tape control unit, for example, may specify *Rewind Tape Unit 3.*

The computer is housed in a single cabinet, 33 inches deep, 37 inches wide, 65 inches high. Thirteen trays, eight trays of logic modules and five trays of memory modules, are horizontally arrayed within the cabinet (Figure C3-1). Logic modules consist of encapsulated printed-circuit cards which plug into the trays. Maintenance test points are readily accessible at the front of the trays.

Computer cabinet doors, which are closed during normal operation, can be opened for maintenance. Inner surfaces of the doors contain maintenance control panels with register indicators, set and clear pushbuttons, and operating switches. A separate operating and maintenance console would be supplied for special applications, as in a multicomputer installation.

Primary power is provided to the computer from a 60-cycle input, 400-cycle output motor-alternator which, in addition to converting frequency, serves to isolate the computer from the main power source. Total power consumption is 2400 watts. For the installation planned for the AN/USQ-20 Unit Computer forced-air cooling is used; supplementary need for a heat exchanger is dictated by environment. Inter-equipment cabling enters the computer at the top of the cabinet and is run through a false floor.

The computer is designed and constructed to withstand severe shock and vibration. It may easily be installed aboard ship or in a trailer without special modification.

2. OPERATION

A simplified block diagram of the AN/USQ-20 Unit Computer appears in Figure C3-2. For explanatory purposes, the computer may be considered as comprised of four major sections: *Input/Output, Storage, Arithmetic,* and *Control.* Abbreviations on the diagram are explained as operation of the various sections is discussed.

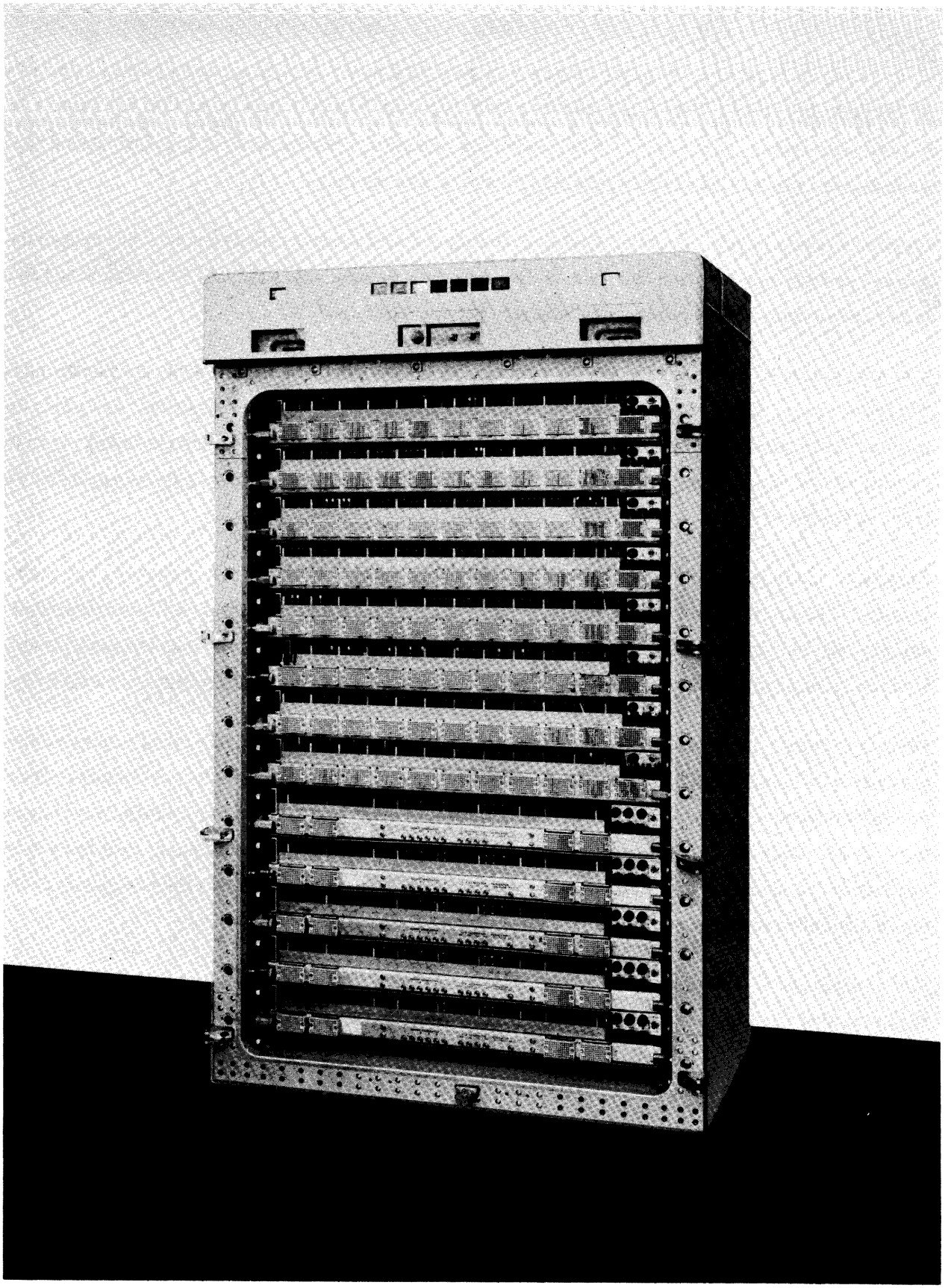


Figure C3-1. Computer Cabinet Interior

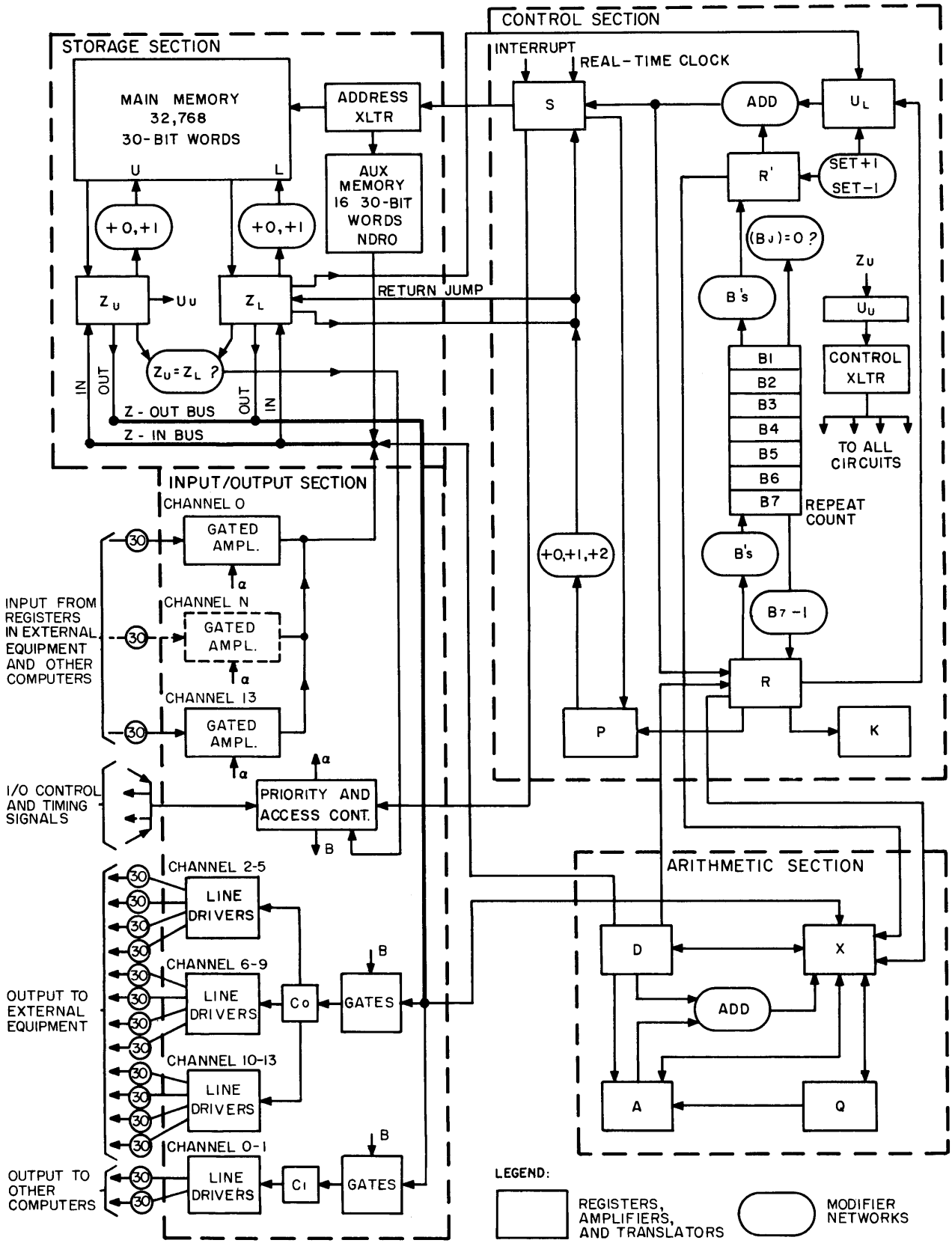


Figure C3-2. Simplified Block Diagram, AN/USQ-20 Unit Computer

A. *INPUT/OUTPUT SECTION*

The *Input/Output section* includes those data paths and control circuits used by the computer for communicating with external equipment. Main parts of the Input/Output section are 1) two output registers (C0 and C1) and their associated line drivers, 2) 14 sets of gated input amplifiers, and 3) priority and access control circuits.

Output Registers

The *C0 register* is used for transmissions to all external devices except other computers. As illustrated in Figure C3-2, C0 receives its input directly from the Storage section via gates controlled by the priority and access circuits. Three sets of 30 line drivers branch from the output of C0; each set drives four output channels. Gated registers located in the external devices determine which channel is active during any particular transmission.

The *C1 register* handles transmissions over the two special output channels — those used to transmit data to other computers. Operation of C1 is similar to that of C0: Words enter C1 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.

Note that 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.

Input Amplifiers

A set of *30 gated amplifiers* is provided for each of the 14 input channels. Gates are controlled by the priority and access circuits, with the channel having the highest number being given priority if two or more inputs are simultaneously requested. As in the case of the output channels, Channels 0 and 1 are used for intercomputer communication, which consequently receives the lowest priority.

This method of treating input data eliminates the need for input buffer registers and gives external equipment direct access to the computer's internal memory.

Priority and Access Circuits

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,

these circuits accept and transmit the control and timing signals which must be exchanged between the computer and equipment with which it communicates.

The 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 a channel already in use.

Main memory addresses referenced during a particular input or output transfer are determined by a special I/O control word. One such word is assigned to each channel. It is sufficient at this point to note that a signal generated by 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.

B. *STORAGE SECTION*

The *Storage section* consists of main 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, is coincident-current driven, and is addressed via the address translator. Content of the referenced address is read into the 30-bit Z register. Because of optional use of 15-bit half-words, Z is split into two 15-bit sections termed Z-upper (Z_U) and Z-lower (Z_L).

The memory operates in the destructive read-out mode. Time required for the read/restore cycle is eight microseconds.

During the restore portion of the cycle, the content of Z_L or Z_U may be increased by one, as indicated by the +0, +1 modifier boxes. This provision allows for automatically increasing the I/O control words, with the result that addresses referenced during a block transfer of data are automatically advanced.

The comparator, $Z_U = Z_L ?$, is used to detect coincidence between the two halves of the I/O control word. When coincidence occurs, a signal is generated to terminate the I/O transfer.

C. *ARITHMETIC SECTION*

The *Arithmetic section* is that part of the computer which performs numeric and logical

calculations. Though greatly simplified, Figure C3-1 shows the important components of the Arithmetic section: the A, D, Q, and X registers and the add network.

The *A register* (30 bits) may be thought of for programming purposes as a conventional accumulator. Because of the logic employed, however, the A register is actually only the main rank of the accumulator; the *D register* serves as a second rank. This configuration, while different from former and usual arrangements, permits the use of a much more reliable building-block circuit.

The Add operation is typical of the relationship between the A and D registers: The augend and addend are initially contained in A and D. As addition is performed, the sum is formed in parallel by the add network and placed in the *X register*. From X, the sum is transmitted to A.

The *Q register* (30 bits) is used principally during multiply and divide operations. The contents of both A and Q may be shifted left or right, individually or as one double-length 60-bit word.

D. *CONTROL SECTION*

The *Control section* consists of those registers and circuits necessary to procure, modify, and execute instructions of the program.

The *U register* (30 bits) is the program-control register. It holds the instruction word during execution of an instruction. The function code and the various execution modifiers are translated from appropriate sections of the register. The lower-order 15 bits of the U register have addition properties, modulus $2^{15}-1$. If an address modification is required before execution, contents of the appropriate *B register* are added to contents of the lower-order 15 bits of the U register before execution.

The *R register* (15 bits) functions as a communication register for the B registers. All internal transmissions to or from B registers pass through the R register. It also holds the quantity used as increment during address modification. This register has counting provisions for increasing contents of the B register.

The K *register* (6 bits) functions as a shift counter for all arithmetic operations that involve shifts. The maximum shift count is 63. Multiply and divide operations are controlled by pre-setting the K register to ZERO and counting operational steps.

The *S register* (15 bits) holds the storage address during memory references. At the beginning of a storage access period, the address is transferred to the S register. The contents of the S register are then translated to activate the storage selection system.

3. CONSOLE CONTROL

Both the indicator/control panel in the doors (referred to as the *in-door console*) and the separate (optional) operating console include 1) indicator lamps that display a detailed report of the internal status of the computer, and 2) controls that allow varied manually governed operations. It is not necessary to monitor the consoles during normal operation.

A. *REGISTERS*

Each register is represented on a console by 1) a row of display lamps, each of which indicates the content of a corresponding register stage; 2) a row of SET buttons, each of which can be used to manually enter a *one* into the corresponding stage; and 3) a CLEAR button, which can be used to manually enter zeros into all stages of the register. Many of the registers are involved in the mechanics of executing instructions, and are not directly accessible to the program. (These registers are not discussed in this publication.)

B. *SPECIAL MODES*

Both the *in-door console* and the separate console are provided with manual controls that permit the following special modes of operation:

1) Execution of one program instruction of a sequence for each depression of a switch.

2) Execution of consecutive program instructions at a low rate governed by a console frequency control.

3) Execution of one master clock phase for each depression of a switch.

4) Execution of consecutive master clock phases at a low rate governed by a console frequency control.

5) Operation that is normal except that the computer does not stop when it executes a programmed *Stop* instruction. (Such operation is called abnormal high-speed operation.)

The consoles are also provided with a manual control that may be used to disable the real-

time clock. This option enables the operator to suspend normal operation temporarily without affecting such operation when it is subsequently resumed. Such suspensions could include stopping the computer or operating temporarily in one of the special modes listed previously.

C. *PROGRAMMED STOPS*

The consoles include a set of six JUMP and STOP switches that can, in normal computer operation, govern the execution of manual-jump instructions. If *stop* conditions are satisfied, the computer stops and an indicator is lit to show the value of j in the manual-jump or manual return-jump instruction that stopped the computer. In abnormal high-speed operation, the indicator is lit but the computer does not stop.

D. *MASTER CLEAR*

The consoles include a spring-return key that is used to clear all registers and control designators to *zero*.

4. OTHER FEATURES

The AN/USQ-20 Unit Computer is especially well suited to real-time control, data processing, and data reduction. Most basic features of the computer are described in previous sections. A detailed account of how these features can be used is beyond the scope of this section. This subsection, however, suggests uses of certain computer features in a few areas. Features considered here are: 1) automatic programming, 2) floating-point arithmetic, 3) the real-time clock, 4) masked comparison, 5) inclusion of operands in instruction words, 6) detection of special faults, 7) program branching, 8) search operations, 9) half-word and full-word logic, 10) external and internal program interrupts, and 11) 16-word wired auxiliary memory.

A. *REAL-TIME 7-DAY CLOCK*

Among the features that suit the computer to real-time problems is the 7-day clock which contains an accurate record of time. The clock may be used to log the receipt times of a periodic real-time input. Each message and its receipt time may be recorded together. Another use of the clock is to initiate periodic programmed operations without requiring more

than occasional attention of the main program. Since the clock recycles only once in 7 days, it is suitable for use where the computer is used on an around-the-clock basis.

B. *MASKED COMPARISON*

Masked Comparison is used to compare all or any part of a word with the contents of the accumulator. It also tests for *equality*, and *non-equality*, *greater than*, or *less than* conditions. In all cases, the original content of the Accumulator is left unchanged.

C. *INCLUSION OF OPERANDS IN INSTRUCTION WORDS*

The lower half (15 bits) of an instruction word is commonly used as an operand address. Where 15-bit operands are acceptable, the lower half of the instruction word may itself serve as an operand. This option of storing the operand as part of the instruction word is particularly advantageous for certain applications. It reduces computation by eliminating a memory reference and provides twice the storage capacity, if 15 bits are adequate for the precision required.

D. *DETECTION OF SPECIAL FAULTS*

Fault conditions may arise in the execution of a *divide* instruction: the divisor may be zero or the quotient may exceed 30 bits (including sign). Either *fault* condition is detected by programming the *divide* instruction with j = 3 (skip if Q is negative). When the instruction is executed, a fault produces a skip. The instruction that follows the one that is skipped contains a jump to a remedial subroutine. The instruction that immediately follows *divide* is programmed with j = 1 (unconditional skip) so that the remedial subroutine is avoided if no fault is detected. The computer never stops because of a divide fault.

When a *multiply* instruction is performed, the product is formed in AQ. If it does not exceed 30 bits (including sign bit), it is contained entirely in Q. If it exceeds 30 bits, it extends into A and is called a double-length product. A method of detecting a double-length product depends on these two facts:

1) During execution of a *multiply* instruction, both factors are arbitrarily represented as positive numbers (sign bit = 0); correction of the sign bit of the product occurs late in the program step.

2) Skip-condition evaluation takes place before sign correction.

The *multiply* instruction is programmed with j = 2 so that, if $(Q_{29}) = one$ before sign correction, a skip is performed. If $(Q_{29}) = zero$, the test is inconclusive, and the next instruction tests for (A) $\neq$ 0. The second test is conclusive: if (A) $\neq$ 0, the product is double length; if (A) = 0, it is not. The additional instruction that contains the second test is skipped unless the first test is inconclusive. If the probable length of the product is known, the programmer can select a variation of this method (such as reversing the order of the two tests) to reduce the probability of having to execute an additional instruction.

E. *PROGRAM BRANCHING*

A convenient method of providing a branch in the program is to include a skip condition followed by a jump instruction. The skip is performed unless the branch condition obtains, so that the jump is not normally performed. If the branch condition obtains, however, the skip is not performed. The jump then diverts the program to the branch. This method (which may be used wherever optional use of an out-of-sequence address is desired) is advantageous because the skip is ordinarily part of some other instruction. Since the skip requires no separate instruction word, no execution time is spent on the branch condition except when that condition is satisfied. This arrangement conserves not only execution time, but memory capacity as well. Since the jump (like the skip) may be conditional, the arrangement simplifies inclusion of compound branch conditions.

Another feature that facilitates preparation of branched programs is the presence of return-jump instructions in the repertoire. When performance of a subroutine initiated by a main-program return-jump is completed, main-program operation is automatically resumed at the point at which it was interrupted.

F. *SEARCH OPERATIONS*

The computer can search internally stored files at very high speeds. The search identifier is entered into the Accumulator, and a mask that identifies the key (a *one* at every key bit position) is entered in Q. The search consists of a masked comparison that is performed repeatedly at successive memory addresses until a *find* (or other terminal condition) is detected.

G. *HALF-WORD AND FULL-WORD LOGIC*

All instructions that procure their operands from memory may select the upper 15 bits, the

lower 15 bits, or the entire 30 bits as the operand. For some purposes, half-word accuracy is generally sufficient, and this feature can be used to reduce computation time and to double the effective memory capacity.

H. *EXTERNAL AND INTERNAL PROGRAM INTERRUPTS*

Provision is made for the interruption of running programs by events which occur asynchronously with the program.

External devices may, by placing a signal on one of 14 External Interrupt lines, interrupt the normal computer program in the event of a failure in data transmission or even in case of termination of some normal mode of operation (in case of a tape system, at end of rewind). Appropriate action is taken by the computer's interrupt program and the normal program is re-entered at the point at which it was left.

Moreover, interrupts are generated by the I/O section of the computer whenever a buffer, which has been initiated previously with a monitor imposed upon it, terminates at the end of the transfer of a given block of data. The interrupt program takes cognizance of the buffer termination, and the main program is resumed.

I. 16-*WORD WIRED AUXILIARY MEMORY*

In addition to the large main memory, a 16-word auxiliary memory is also provided. It is a wired memory and operates in the nondestructive read-out mode. The auxiliary memory is used to contain important instructions or constants. For example, a program-load routine may be stored there to facilitate rapid changes in the main program and automatic program recovery.

PART 2

REPERTOIRE OF INSTRUCTIONS AND PROGRAM TIMING

1. BASIC INFORMATION

This portion of this section presents the instruction repertoire for the AN/USQ-20 Unit Computer. Details presented are limited to the needs of the programmer and list only symbols, registers, terms, and instruction characteristics pertinent to programming the computer.

As mentioned previously, the AN/USQ-20 Unit Computer is a self-modifying, one-address computer. 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 (index) registers one through seven, which contain any previously stored constants. To modify the address, the content of a selected B register is added to the Operand Designator, **y**.

A programmed address is coded using octal notation with each octal digit denoting three binary digits. The instructions are read sequentially from Magnetic Core Storage except after Jump or Skip instructions.

A. *SYMBOL CONVENTIONS*

The following symbols are used throughout the descriptive material on instructions:

α	=	a register (A, Q, B^n), a memory location Y, or a constant.
(α)	=	content of α.
$(\alpha)_i$	=	initial content of α.
$(\alpha)_f$	=	final content of α.
α_n	=	the n^{th} bit of α.
$(\alpha)_n$	=	the n^{th} bit of the content of α.
f	=	Function Code Designator $(i_{29}, \ldots, i_{24})$*.
j	=	Branch Condition Designator $(i_{23}, \ldots, i_{21})$*.
$\hat{\mathbf{j}}$	=	Input/Output Channel Designator $(i_{23}, \ldots, i_{20})$*.
k	=	Operand Interpretation Designator $(i_{20}, \ldots, i_{18})$*.
$\hat{\mathbf{k}}$	=	Operand Interpretation Designator $(i_{19}, \ldots, i_{18})$*.
b	=	Index Designator (i_{17}, i_{16}, i_{15})*.
y	=	Operand Designator $(i_{14}, \ldots, i_0)$*.
Y	=	the Operand (regardless of source).
Y	=	**y** + (B^b).

1) The operand or address of the operand for the *Read* portion of an instruction or

2) The destination address for the *Store* portion of an instruction.

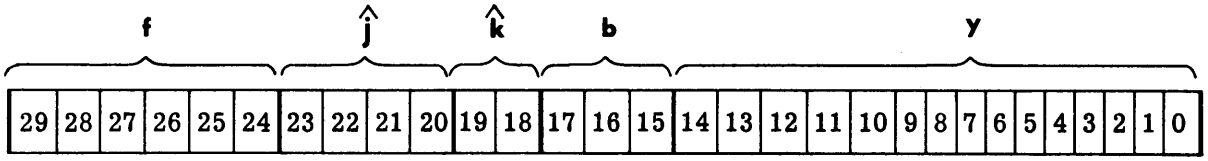
(Y)	=	content of memory address Y.
L(Y)(Q)	=	bit-by-bit multiplication, logical multiply of $\mathbf{Y}_n$, and $(Q)_n$.
A	=	A register or accumulator (30-bit arithmetic register).
B	=	seven B registers (15 bits each). B registers are address-modifying registers generally used for indexing loops in a program; in addition, B^7 serves as a *repeat* counter. (The address modification does not alter the instructions as stored in memory.) A **b** or **j** designator specifies the B register used.
Q	=	Q register (30-bit arithmetic register).
U	=	U register (30 bits). The U register holds the instruction word during execution of an operation. If address modification is required before execution, the appropriate B register content is added to the lower-order 15 bits of the U register before execution.

* i_n is the n^{th} bit position in an instruction.

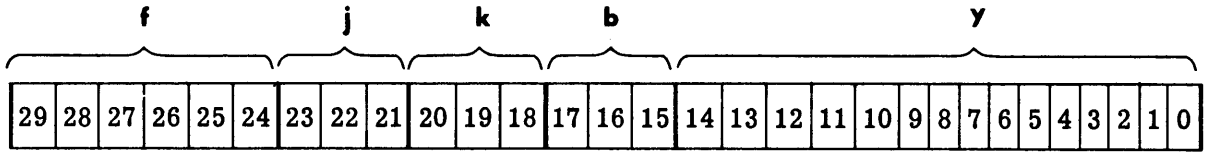
P = P register (15 bits). The P register is the Program Address Register. This register holds the address of the current instruction throughout the program except for Jump instructions where the P register is cleared and the new program address is entered.

C = the 14D input/output channels (30 lines each). Channels consist of transmission lines, therefore they *cannot* be considered registers. The designator $\hat{j}$ specifies (in octal) the channel used.

Figure C3-3 illustrates bit configuration of instruction designators in two forms. Form I pertains to input/output instructions; Form II pertains to all other instructions.



Form I - Input/Output Instructions



Form II - All Other Instructions

Note: $\hat{j} = C^n$ input/output channel

Figure C3-3. Bit Allocation of Instruction Designators

Table C3-1 is a list of the computer's entire repertoire of instructions; each instruction is listed by its function code number and name.

Table C3-2 indicates the time, in microseconds, required to execute each instruction.

TABLE C3-1. INSTRUCTION REPERTOIRE - AN/USQ-20 UNIT COMPUTER

CODE	FUNCTION NAME	CODE	FUNCTION NAME
00	(Fault Interrupt)	40	ENTER LOGICAL PRODUCT
01	RIGHT SHIFT Q	41	ADD LOGICAL PRODUCT
02	RIGHT SHIFT A	42	SUBTRACT LOGICAL PRODUCT
03	RIGHT SHIFT AQ	43	COMPARE MASKED
04	COMPARE	44	REPLACE LOGICAL PRODUCT
05	LEFT SHIFT Q	45	REPLACE A + LOGICAL PRODUCT
06	LEFT SHIFT A	46	REPLACE A - LOGICAL PRODUCT
07	LEFT SHIFT AQ	47	STORE LOGICAL PRODUCT
10	ENTER Q	50	SELECTIVE SET
11	ENTER A	51	SELECTIVE COMPLEMENT
12	ENTER B^n	52	SELECTIVE CLEAR
13	EXTERNAL FUNCTION ON C^n	53	SELECTIVE SUBSTITUTE
14	STORE Q	54	REPLACE SELECTIVE SET
15	STORE A	55	REPLACE SELECTIVE COMPLEMENT
16	STORE B^n	56	REPLACE SELECTIVE CLEAR
17	STORE C^n	57	REPLACE SELECTIVE SUBSTITUTE
20	ADD A	60	JUMP (Arithmetic)
21	SUBTRACT A	61	JUMP (Manual)
22	MULTIPLY	62	JUMP ON C^n ACTIVE INPUT BUFFER
23	DIVIDE	63	JUMP ON C^n ACTIVE OUTPUT BUFFER
24	REPLACE A + Y	64	RETURN JUMP (Arithmetic)
25	REPLACE A - Y	65	RETURN JUMP (Manual)
26	ADD Q	66	TERMINATE C^n INPUT BUFFER
27	SUBTRACT Q	67	TERMINATE C^n OUTPUT BUFFER
30	ENTER Y + Q	70	REPEAT
31	ENTER Y - Q	71	B SKIP ON B^n
32	STORE A + Q	72	B JUMP ON B^n
33	STORE A - Q	73	INPUT BUFFER ON C^n (without Monitor mode)
34	REPLACE Y + Q	74	OUTPUT BUFFER ON C^n (without Monitor mode)
35	REPLACE Y - Q	75	INPUT BUFFER ON C^n (with Monitor mode)
36	REPLACE Y + 1	76	OUTPUT BUFFER ON C^n (with Monitor mode)
37	REPLACE Y - 1	77	(Fault Interrupt)

TABLE C3-2. INSTRUCTION EXECUTION TIMES

f	j = 0, 1 NORMAL k=0,4	j = 0, 1 NORMAL k=7	j = 0, 1 NORMAL k≠0,4,7	j = 0 REPEAT k=0,4	j = 0 REPEAT k=7	j = 0 REPEAT k≠0,4,7
01	9.6/12.8	11.2/14.4	16			
02	9.6/12.8	11.2/14.4	16			
03	11.2/16	11.2/16	16/20.8			
04	12.8	11.2	16	9.6	8.0	11.2
05	9.6/12.8	9.6/12.8	16			R
06	9.6/12.8	9.6/12.8	16			R
07	11.2/16	11.2/16	16/20.8			R
10	11.2	9.6	16	8.0	6.4	9.6 R
11	11.2	9.6	16	8.0	6.4	9.6
12	8.0	9.6	16	4.6	6.4	9.6
13	12.8	-	24			
S 14	12.8	-	16	6.4	-	9.6
S 15	12.8	-	16	6.4	-	9.6
S 16	12.8	-	16	6.4	-	9.6
S 17	-	-	16			
20	11.2	9.6	16	8.0	6.4	9.6
21	11.2	9.6	16	8.0	6.4	9.6
22			35.2 - 112			
23			112			
R 24	-	-	24	-	-	16
R 25	-	-	24	-	-	16
26	12.8	11.2	16	9.6	8.0	11.2
27	12.8	11.2	16	9.6	8.0	11.2

f	j = 0, 1 NORMAL k=0,4	j = 0, 1 NORMAL k=7	j = 0, 1 NORMAL k≠0,4,7	j = 0 REPEAT k = 0,4	j = 0 REPEAT k=7	j = 0 REPEAT k≠0,4,7
30	11.2	9.6	16	8	6.4	9.6
31	11.2	9.6	16	8	6.4	9.6
S 32	12.8	-	16	6.4	-	9.6
S 33	12.8	-	16	6.4	-	9.6
R 34	-	-	24	-	-	16
R 35	-	-	24	-	-	16
R 36	-	-	24	-	-	16
R 37	-	-	24	-	-	16
40	11.2	9.6	16	8	6.4	9.6
41	11.2	9.6	16	8	6.4	9.6
42	11.2	9.6	16	8	6.4	9.6
43	11.2	9.6	16	8	6.4	9.6
R 44	-	-	24	-	-	16
R 45	-	-	24	-	-	16
R 46	-	-	24	-	-	16
S 47	12.8	-	16	6.4	-	9.6
50	12.8	11.2	16	9.6	8.0	11.2
51	12.8	11.2	16	9.6	8.0	11.2
52	12.8	11.2	16	9.6	8.0	11.2
53	12.8	11.2	16	9.6	8.0	11.2
R 54	-	-	24	-	-	16
R 55	-	-	24	-	-	16
R 56	-	-	24	-	-	16
R 57	-	-	24	-	-	16

f	j = 0 NORMAL k=0,4	j = 0 NORMAL k=7	j = 0 NORMAL k≠0,4,7
60	8	9.6	16
61	8	9.6	16
62	8		16
63	8		16
64	12.8/19.2	11.2/17.6	16/24
65	12.8/19.2	11.2/17.6	16/24
66	8	-	16
67	8	-	16
70	8	9.6	16
71	9.6	11.2	16
72	8	9.6	16
73	16	-	24
74	16	-	24
75	16	-	24
76	16	-	24
77	-	-	-

S - STORE

R - REPLACE

Note:

All times are in microseconds

B. *FUNCTION CODE DESIGNATOR* - **f**

The **f** designator (6 bits) appears in bit-positions 29 through 24 of the U register, or an instruction, designating the function to be performed by that instruction. All values of **f** other than 00 and 77 are defined in the instruction list. The two codes 00 and 77 are *fault conditions* which, if executed, will cause a *fault interrupt.* This results in a jump to address 00014, the Fault Entrance Register or address 00014 of *wired* memory depending on the Automatic Recovery Switch setting (see page C3-23).

C. *BRANCH CONDITION DESIGNATOR* - **j**

The **j** designator (3 bits) appears in bit-positions 23, 22, and 21 of the U register, or an instruction; it is used in a majority of the instructions (see Figure C3-3, Form II). There are three primary categories of use: 1) for Jump and Skip determination, 2) for B register specification, and 3) for repeat status interpretation. Appropriate interpretations of the **j** designator are listed either below or under the descriptions of the individual instructions.

For those instructions in which the **j** designator has no special interpretation, it specifies the condition under which the next sequential instruction in the program will be skipped. This provides for branching from a sequence without executing a Jump instruction, as would normally occur if a Skip condition were not satisfied.

Skip of the next sequential instruction is determined by the following rules in all instructions except 04, 12, 13, 16, 17, 26, 27, 60 through 67, and 70 through 76.

j = 0: Do not skip the next instruction.
j = 1: Skip the next instruction.
j = 2: Skip the next instruction if (Q) is positive.
j = 3: Skip the next instruction if (Q) is negative.
j = 4: Skip the next instruction if (A) is zero.*
j = 5: Skip the next instruction if (A) is nonzero.
j = 6: Skip the next instruction if (A) is positive.
j = 7: Skip the next instruction if (A) is negative.

When the branch (Skip or Jump) condition involves the sign of the quantity in A or Q, the evaluation examines the sign bit of these quantities; hence, a positive zero (all *zeros*) is considered a positive quantity, and a negative zero (all *ones*) is considered a negative quantity.

---

* Positive zero

D. *INPUT/OUTPUT CHANNEL DESIGNATOR* - $\hat{j}$

The $\hat{j}$ designator (4 bits) appears in bit-positions 23, 22, 21, and 20 of the U register, or an input/output instruction, specifying the C-channel for the instruction (see Figure C3-3, Form 1). Bit 23 assumes a value of eight, bit 22 a value of four, bit 21 a value of two, and bit 20 a value of one; thus the $\hat{j}$ designator provides accessibility to the 14 (decimal) input/output channels numbered $0\text{-}15_8$.

Instructions 13, 17, 62, 63, 66, 67, 73, 74, 75, and 76 use the $\hat{j}$ designator configuration.

E. *OPERAND INTERPRETATION DESIGNATOR* - **k** or $\hat{k}$

The **k** designator (3 bits) [or $\hat{k}$ designator (2 bits)] appears in bit-positions 20, 19, and 18 of the U register, or an instruction; a $\hat{k}$ designator appears only in bit-positions 19 and 18, since bit 20 is a portion of the $\hat{j}$ designator. (See Figure C3-3, Forms I and II.) Instructions 13, 17, 62, and 73 through 76 use the $\hat{k}$ designator configuration since they perform input/output activities and require a $\hat{j}$ designator for channel specification.

The **k** and $\hat{k}$ designators control operand interpretation. Those instructions which *read* an operand but do not replace it after the arithmetic is performed are designated *Read* instructions. Those instructions which do not *read* an operand but *store* one are designated *Store* instructions. Instructions which both *read* and *store* operands are classified as *Replace* instructions.

The various values of **k** or $\hat{k}$ affect the operand in the following list except where otherwise noted under individual instruction descriptions.

1) *Read* instructions (01 through 13, 20 through 23, 26, 27, 30, 31, 40 through 43, 50 through 53, and 60 through 76):

k or $\hat{k}$ = 0: $\mathbf{Y}_u$ = 0's; $\mathbf{Y}_L = Y$.

k or $\hat{k}$ = 1: $\mathbf{Y}_u$ = 0's; $\mathbf{Y}_L = (Y)_L$.

k or $\hat{k}$ = 2: $\mathbf{Y}_u$ = 0's; $\mathbf{Y}_L = (Y)_u$.

k or $\hat{k}$ = 3: $\mathbf{Y} = Y$.

k = 4: $\mathbf{Y}_u$ = same bits as Y_{14}; $\mathbf{Y}_L = Y$.

k = 5: $\mathbf{Y}_u$ = same bits as Y_{14}; $\mathbf{Y}_L = (Y)_L$.

k = 6: $\mathbf{Y}_u$ = same bits as Y_{29}; $\mathbf{Y}_L = (Y)_u$.

k = 7: $\mathbf{Y} = (A)$.

For instructions 23, 52, and 53, **k** = 7 is not used.

For instruction 13, only $\hat{\mathbf{k}} = 3$ is permitted.

For instructions 73 through 76, $\hat{\mathbf{k}} = 2$ is not used.

2) *Store* instructions (14 through 16, 17, 32, 33, and 47):

k = 0: Store (A or B^j) in Q*.

k = 1: Store (A_L, Q_L, or B^j) in Y_L, leaving $(Y)_u$ undisturbed.

k = 2: Store (A_L, Q_L, or B^j) in Y_u, leaving $(Y)_L$ undisturbed.

k or $\hat{\mathbf{k}}$ = 3: Store (A, Q, C^j, or B^j) in Y.

k = 4: Store (Q or B^j) in A**.

k = 5: Store complement of (A_L, Q_L, or B^j) in Y_L, leaving $(Y)_u$ undisturbed.

k = 6: Store complement of (A_L, Q_L, or B^j) in Y_u, leaving $(Y)_L$ undisturbed.

k = 7: Store complement of (A, Q, or B^j) in Y. (Storing the complement of B^j is the same complement as for a 30-bit register.)

For instruction 17, only $\hat{\mathbf{k}} = 3$ is permitted.

3) *Replace* instructions (24, 25, 34 through 37, 44 through 46, and 54 through 57):

k = 0: Not used.

k = 1: *Read* portion - $\mathbf{Y}_u$ = 0's; $\mathbf{Y}_L = (Y)_L$.

Store portion - stores (A_L, Q_L, or B^j) in Y_L, leaving $(Y)_u$ undisturbed.

k = 2: *Read* portion - $\mathbf{Y}_u$ = 0's; $\mathbf{Y}_L = (Y)_u$.

Store portion - stores (A_L, Q_L, or B^j) in Y_u, leaving $(Y)_u$ undisturbed.

k = 3: *Read* portion - **Y** = Y.

Store portion - stores (A, Q, or B^j) in Y.

k = 4: Not used.

k = 5: *Read* portion - $\mathbf{Y}_u$ = same bits at Y_{14}; $Y_L = (Y)_L$.

Store portion - stores (A_L, Q_L, or B^j) in Y_L, leaving $(Y)_u$ undisturbed.

---

* A 14000 00000 instruction complements (Q).

** A 15040 00000 instruction complements (A).

k = 6: *Read* portion - $\mathbf{Y}_u$ = same bits as Y_{29}; $\mathbf{Y}_L = Y_u$.

Store portion - stores (A_L, Q_L, or B^j) in Y_u, leaving $(Y)_L$ undisturbed.

k = 7: Not used.

The *Repeat* instruction requires special interpretation when followed by a *Replace* instruction. See details on page C3-40, Instruction No. 70, REPEAT.

F. *INDEX DESIGNATOR* - **b**

The **b** designator (3 bits) appears in bit-positions 17, 16, and 15 of the U register, or an instruction (see Figure C3-3), specifying which of the B registers, if any, will be used to modify the Operand Designator, **y**, to form $Y = \mathbf{y} + (B^b)$. This operation employs an additive accumulator; hence, a quantity consisting of all *zeros* cannot result unless the bits of both the Operand Designator, **y**, and (B^b) are all *zeros*.

Effect of the various values of **b**, the Index Designator, is summarized:

b = 0: Do not modify **y**.

b = 1: Add (B^1) to **y** (modulo 2^{15}-1).

b = 2: Add (B^2) to **y** (modulo 2^{15}-1).

b = 3: Add (B^3) to **y** (modulo 2^{15}-1).

b = 4: Add (B^4) to **y** (modulo 2^{15}-1).

b = 5: Add (B^5) to **y** (modulo 2^{15}-1).

b = 6: Add (B^6) to **y** (modulo 2^{15}-1).

b = 7: Add (B^7) to **y** (modulo 2^{15}-1).

G. *OPERAND DESIGNATOR* - **y**

The **y** designator (15 bits) appears in bit-positions 14 through 0 of an instruction (see Figure C3-3). The operand or address of the operand, **Y**, is relative to **y** since $Y = \mathbf{y} + (B^b)$.

H. *MAGNETIC CORE MEMORY ASSIGNMENT*

The main Magnetic Core memory consists of 32,768 addressable storage locations. Seventy-three of these locations are special-purpose and provide eight distinct functions:

1) The starting address from MASTER CLEAR
2) The Fault Entrance Register
3) The Real-Time Clock Register
4) External Interrupt Entrance Register for each channel
5) Internal Interrupt Entrance Register for each *input* channel
6) Internal Interrupt Entrance Register for each *output* channel
7) Input Buffer Control Register for each *input* channel
8) Output Buffer Control Register for each *output* channel.

Each of the other memory locations are used for:

1) Instruction word storage
2) Data storage.

Table C3-3 specifies Magnetic Core Memory Address assignments and associated storage functions.

I. *WIRED MEMORY*

The AN/USQ-20 Unit Computer contains 16D words of semipermanent wired storage. Programming this memory area requires a process of ***wiring-in*** the desired instructions. The semipermanent feature of these storage locations prevents accidental destruction of program instructions contained therein since entries cannot be made via main memory.

An Input Bootstrap routine occupies this memory, and its execution is controlled by the Automatic Recovery Switch. (See Section E3, "AN/USQ-20 Wired Bootstrap.")

J. *AUTOMATIC RECOVERY*

In the event of a ***fault*** condition (encountering either a 00 or 77 function code), the Automatic Recovery Switch directs computer activity. This switch has three positions: 1) DOWN, 2) NEUTRAL, and 3) UP. Action resulting from these positions is:

1) DOWN position - This causes manual execution of the Bootstrap routine. Computer action begins at address 0 of Wired Memory and executes the Bootstrap routine when this switch is depressed. (A MASTER CLEAR should precede this operation.)
2) NEUTRAL position - This causes an Interrupt to address 00000 of Main Memory on a fault condition. Action continues as programmed.
3) UP position - This causes an Interrupt to address 00000 of Wired Memory on a fault condition. This results in automatic execution of the Bootstrap routine.

TABLE C3-3. CORE MEMORY ADDRESS ASSIGNMENTS

ADDRESS (Octal)	STORAGE FUNCTION
0 0 0 0 0	Fault Entrance Register
0 0 0 0 1	Memory Word
0 0 0 0 2	Memory Word
0 0 0 0 3	Memory Word
0 0 0 0 4	Memory Word
0 0 0 0 5	Memory Word
0 0 0 0 6	Memory Word
0 0 0 0 7	Memory Word
0 0 0 1 0	Memory Word
0 0 0 1 1	Memory Word
0 0 0 1 2	Memory Word
0 0 0 1 3	Memory Word
0 0 0 1 4	Memory Word
0 0 0 1 5	Memory Word
0 0 0 1 6	Memory Word
0 0 0 1 7	Memory Word
0 0 0 2 0	External Interrupt Entrance Register for Channel 0
0 0 0 2 1	External Interrupt Entrance Register for Channel 1
0 0 0 2 2	External Interrupt Entrance Register for Channel 2
0 0 0 2 3	External Interrupt Entrance Register for Channel 3
0 0 0 2 4	External Interrupt Entrance Register for Channel 4
0 0 0 2 5	External Interrupt Entrance Register for Channel 5
0 0 0 2 6	External Interrupt Entrance Register for Channel 6
0 0 0 2 7	External Interrupt Entrance Register for Channel 7
0 0 0 3 0	External Interrupt Entrance Register for Channel 8D
0 0 0 3 1	External Interrupt Entrance Register for Channel 9D
0 0 0 3 2	External Interrupt Entrance Register for Channel 10D
0 0 0 3 3	External Interrupt Entrance Register for Channel 11D
0 0 0 3 4	External Interrupt Entrance Register for Channel 12D
0 0 0 3 5	External Interrupt Entrance Register for Channel 13D

TABLE C3-3. CORE MEMORY ADDRESS ASSIGNMENTS (Continued)

ADDRESS (Octal)	STORAGE FUNCTION
0 0 0 3 6	Real-Time Clock Register
0 0 0 3 7	Memory Word
0 0 0 4 0	Internal Interrupt Entrance Register for Input Channel 0
0 0 0 4 1	Internal Interrupt Entrance Register for Input Channel 1
0 0 0 4 2	Internal Interrupt Entrance Register for Input Channel 2
0 0 0 4 3	Internal Interrupt Entrance Register for Input Channel 3
0 0 0 4 4	Internal Interrupt Entrance Register for Input Channel 4
0 0 0 4 5	Internal Interrupt Entrance Register for Input Channel 5
0 0 0 4 6	Internal Interrupt Entrance Register for Input Channel 6
0 0 0 4 7	Internal Interrupt Entrance Register for Input Channel 7
0 0 0 5 0	Internal Interrupt Entrance Register for Input Channel 8D
0 0 0 5 1	Internal Interrupt Entrance Register for Input Channel 9D
0 0 0 5 2	Internal Interrupt Entrance Register for Input Channel 10D
0 0 0 5 3	Internal Interrupt Entrance Register for Input Channel 11D
0 0 0 5 4	Internal Interrupt Entrance Register for Input Channel 12D
0 0 0 5 5	Internal Interrupt Entrance Register for Input Channel 13D
0 0 0 5 6	Memory Word
0 0 0 5 7	Memory Word
0 0 0 6 0	Internal Interrupt Entrance Register for Output Channel 0
0 0 0 6 1	Internal Interrupt Entrance Register for Output Channel 1
0 0 0 6 2	Internal Interrupt Entrance Register for Output Channel 2
0 0 0 6 3	Internal Interrupt Entrance Register for Output Channel 3
0 0 0 6 4	Internal Interrupt Entrance Register for Output Channel 4
0 0 0 6 5	Internal Interrupt Entrance Register for Output Channel 5
0 0 0 6 6	Internal Interrupt Entrance Register for Output Channel 6
0 0 0 6 7	Internal Interrupt Entrance Register for Output Channel 7
0 0 0 7 0	Internal Interrupt Entrance Register for Output Channel 8D
0 0 0 7 1	Internal Interrupt Entrance Register for Output Channel 9D
0 0 0 7 2	Internal Interrupt Entrance Register for Output Channel 10D
0 0 0 7 3	Internal Interrupt Entrance Register for Output Channel 11D

TABLE C3-3. CORE MEMORY ADDRESS ASSIGNMENTS (Continued)

ADDRESS (Octal)	STORAGE FUNCTION
0 0 0 7 4	Internal Interrupt Entrance Register for Output Channel 12D
0 0 0 7 5	Internal Interrupt Entrance Register for Output Channel 13D
0 0 0 7 6	Memory Word
0 0 0 7 7	Memory Word
0 0 1 0 0	Input Buffer Control Register for Input Channel 0
0 0 1 0 1	Input Buffer Control Register for Input Channel 1
0 0 1 0 2	Input Buffer Control Register for Input Channel 2
0 0 1 0 3	Input Buffer Control Register for Input Channel 3
0 0 1 0 4	Input Buffer Control Register for Input Channel 4
0 0 1 0 5	Input Buffer Control Register for Input Channel 5
0 0 1 0 6	Input Buffer Control Register for Input Channel 6
0 0 1 0 7	Input Buffer Control Register for Input Channel 7
0 0 1 1 0	Input Buffer Control Register for Input Channel 8D
0 0 1 1 1	Input Buffer Control Register for Input Channel 9D
0 0 1 1 2	Input Buffer Control Register for Input Channel 10D
0 0 1 1 3	Input Buffer Control Register for Input Channel 11D
0 0 1 1 4	Input Buffer Control Register for Input Channel 12D
0 0 1 1 5	Input Buffer Control Register for Input Channel 13D
0 0 1 1 6	Memory Word
0 0 1 1 7	Memory Word
0 0 1 2 0	Output Buffer Control Register for Output Channel 0
0 0 1 2 1	Output Buffer Control Register for Output Channel 1
0 0 1 2 2	Output Buffer Control Register for Output Channel 2
0 0 1 2 3	Output Buffer Control Register for Output Channel 3
0 0 1 2 4	Output Buffer Control Register for Output Channel 4
0 0 1 2 5	Output Buffer Control Register for Output Channel 5
0 0 1 2 6	Output Buffer Control Register for Output Channel 6
0 0 1 2 7	Output Buffer Control Register for Output Channel 7
0 0 1 3 0	Output Buffer Control Register for Output Channel 8D
0 0 1 3 1	Output Buffer Control Register for Output Channel 9D

TABLE C3-3. CORE MEMORY ADDRESS ASSIGNMENTS (Continued)

ADDRESS (Octal)	STORAGE FUNCTION
0 0 1 3 2	Output Buffer Control Register for Output Channel 10D
0 0 1 3 3	Output Buffer Control Register for Output Channel 11D
0 0 1 3 4	Output Buffer Control Register for Output Channel 12D
0 0 1 3 5	Output Buffer Control Register for Output Channel 13D
(0 0 1 3 6 - 0 7 7 7 7)	= 4,002D words of memory
(1 0 0 0 0 - 1 7 7 7 7)	= 4,096D words of memory
(2 0 0 0 0 - 2 7 7 7 7)	= 4,096D words of memory
(3 0 0 0 0 - 3 7 7 7 7)	= 4,096D words of memory
(4 0 0 0 0 - 4 7 7 7 7)	= 4,096D words of memory
(5 0 0 0 0 - 5 7 7 7 7)	= 4,096D words of memory
(6 0 0 0 0 - 6 7 7 7 7)	= 4,096D words of memory
(7 0 0 0 0 - 7 7 7 7 7)	= 4,096D words of memory

All references to wired memory are controlled by means of a flip-flop. For ease of reference, this flip-flop will be called WMFF (Wired Memory Flip-Flop).

1) When the WMFF is set, a *read* portion of a memory reference to addresses 00000 through 00017 will reference wired memory.

2) When the WMFF is clear, a *read* portion of a memory reference to addresses 00000 through 00017 will reference main memory.

3) Addresses 00020 through 77777 only exist in main memory and all references to them are unaffected by WMFF.

4) The *write* portion of a memory reference (even the restore portion of a read cycle) to addresses 00000 through 00017 will always reference main memory regardless of the status of WMFF.

WMFF is set in *two ways only:* 1) manually by depressing the Automatic Recovery Switch (ARS) into the DOWN position or 2) by a fault condition (encountering either a 00 or 77 function code) when the ARS is in the UP position.

If the ARS is in the NEUTRAL position when a fault condition occurs, the interrupt lockout is set, the S register is cleared to 00000, and A sequence is selected with the transmission of P to S inhibited. WMFF is not affected (previous state of WMFF prevails).

If the ARS is in the UP position when a fault condition occurs, WMFF is set, the interrupt lockout is set, the S register is forced to 00000, and A sequence is selected with the transmission of P to S inhibited. This will execute the instruction at 00000 in wired memory since WMFF is set.

If the ARS is manually depressed into the DOWN position (the switch will snap back to NEUTRAL when released), the A sequence is selected, WMFF is set, the interrupt lockout is set, and computing begins. This depressing of ARS *must* be preceded by a MASTER CLEAR in order to clear the P register to 00000.

WMFF is cleared by *one program method only*. Execution of a 601XXXXXXX or 600XXXXXXX instruction will clear the WMFF and the interrupt lockout. *Only* the above two configurations will clear these flip-flops; thus, these flip-flops will not be affected by conditional jumps or a 61XXXXXXXX instruction. MASTER CLEAR always clears WMFF and the interrupt lockout.

K. *BUFFER MODES*

The AN/USQ-20 Unit Computer provides two modes of buffering: 1) with monitor and 2)

without monitor.

Buffering with monitor transfers words sequentially, starting at a given initial address through a given terminal address, on the specified input or output channel. The computer continues execution of program instructions during the buffer process. Completion of the buffering process causes an Internal Monitor Interrupt to the Internal Interrupt Entrance Register assigned to the input or output channel. (See subsection H, *MAGNETIC CORE MEMORY ASSIGNMENT.*) This register should contain a RETURN-JUMP instruction*. (See Instructions 75 and 76.)

Buffering without the monitor transfers words sequentially, starting at a given initial address through a given terminal address, on a specified input or output channel. The computer continues execution of program instructions during the buffer process. No monitor interrupt will occur. (See Instructions 73 and 74.)

2. LIST OF INSTRUCTIONS

This subsection lists the repertoire of instructions used with the AN/USQ-20 Unit Computer. Common usage of these instructions is also included; no attempt is made to indicate more sophisticated use.

01 *RIGHT SHIFT Q*

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 24 bits are ignored.

Example of right shift in Q: **Y** = 2

Content of Q		Content of Q	
$(Q)_i$ (positive) =	0 1 0 1	$(Q)_i$ (negative) =	1 0 1 0
First shift	0 0 1 0	First shift	1 1 0 1
Second shift	0 0 0 1	Second shift	1 1 1 0

02 *RIGHT SHIFT A*

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

*Suggested instruction for the Internal Interrupt Register is:
650nn nnnnn - Exit to an Interrupt subroutine for remedial action. This subroutine ends with a 601nn instruction which clears the Interrupt mode, then returns control to the main routine.

Y are recognized for this instruction. The higher-order 24 bits are ignored. The overall operation is analogous to the example given in the foregoing instruction.

03 *RIGHT SHIFT AQ*

This instruction shifts (A) and (Q) as one 60-bit register. The shift is to the right Y bit positions with the lower-order bits of A shifting into the higher-order 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 24 bits are ignored.

Example of right shift in AQ: Y = 2

Content of AQ		Content of AQ	
$(AQ)_i$ (positive) =	0 1 0 1 0 0 1 1	$(AQ)_i$ (negative) =	1 0 0 0 1 0 1 0
First shift	0 0 1 0 1 0 0 1	First shift	1 1 0 0 0 1 0 1
Second shift	0 0 0 1 0 1 0 0	Second shift	1 1 1 0 0 0 1 0

04 *COMPARE*

This instruction compares the signed value of Y with the signed value of (A) and/or (Q). It does not alter either (A) or (Q). The Branch Condition Designator, j, is interpreted in a special way for this instruction as listed below:

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).

j = 3: Skip the next instruction if Y is greater than (Q).

j = 4: Skip the next instruction if (Q) is greater than, or equal to Y, and Y is greater than (A).

j = 5: Skip the next instruction if Y is greater than (Q) or if Y is less than, or equal to, (A).

j = 6: Skip the next instruction if Y is less than, or equal to, (A).

j = 7: Skip the next instruction if Y is greater than (A).

05 *LEFT SHIFT Q*

This instruction shifts (Q) circularly to the left Y bit positions*. The lower-order bits

* Maximum shift count permitted is 59D places.

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 24 bits are ignored.

Example of left circular shift in Q: (Y) = 2

Content of Q		Content of Q	
$(Q)_i$ (positive) =	0 0 1 1	$(Q)_i$ (negative) =	1 1 0 0
First shift	0 1 1 0	First shift	1 0 0 1
Second shift	1 1 0 0	Second shift	0 0 1 1

06 *LEFT SHIFT A*

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 24 bits are ignored. The over-all operation is analogous to the example given in the foregoing instruction.

07 *LEFT SHIFT AQ*

This instruction shifts (A) and (Q) as one 60-bit register. The shift is circular to the left Y bit positions.* The lower-order bits of A are replaced with the higher-order bits of Q and the lower-order bits of Q are replaced with the higher-order bits of A. Only the lower-order six bits of Y are recognized by this instruction. The higher-order 24 bits are ignored.

Example of left circular shift in AQ: Y = 2

Content of AQ		Content of AQ	
$(AQ)_i$ (positive) =	0 1 0 1 0 0 1 1	$(AQ)_i$ (negative) =	1 0 0 0 1 0 1 1
First shift	1 0 1 0 0 1 1 0	First shift	0 0 0 1 0 1 1 1
Second shift	0 1 0 0 1 1 0 1	Second shift	0 0 1 0 1 1 1 0

10 *ENTER Q*

Clear the Q register. Then transmit Y to Q.

11 *ENTER A*

Clear A. Then transmit Y to A.

12 *ENTER* B^n

Clear B-register j. Then transmit the lower-order 15 bits of Y to B-register j. The higher-order 15 bits of Y are ignored in this instruction. The Branch Condition Desig-

* Maximum shift count permitted is 59D places.

nator, **j**, is used to specify the selected B register for this instruction and is not available for its normal function.

13 *EXTERNAL FUNCTION ON* C^n

$\hat{\mathbf{j}}$ = 0 or 1. Interrogate the two bits connected to the input-active designator (flip-flops) on an interconnected computer. If the interconnected computer's input buffer is active, skip the next instruction. If the interconnected computer's input buffer is not active, execute the next instruction. There are no External Function lines on C^0 or C^1. $\hat{\mathbf{k}}$ = 3 is required for timing. When $\hat{\mathbf{j}} \neq 0$ or 1, transmit **Y**, the External Function, over the channel specified by $\hat{\mathbf{j}}$. Only $\hat{\mathbf{k}}$ = 3 is permitted.

14 *STORE Q*

Store (Q) at storage address Y as directed by the Operand Interpretation Designator, **k**. If **k** = 0, complement (Q). If **k** = 4, store in A.

15 *STORE A*

Store (A) at storage address Y as directed by the Operand Interpretation Designator, **k**. If **k** = 4, complement (A). If **k** = 0, store in Q.

16 *STORE* B^n

Store a 30-bit quantity whose lower-order 15 bits correspond to the content of B-register **j** and whose higher-order 15 bits are *zero* at storage address Y as directed by the Operand Interpretation Designator, **k**. The Branch Condition Designator, **j**, is used to specify the selected B register for this instruction and is not available for its normal function.

17 *STORE* C^n

Store the content of the C-channel specified by $\hat{\mathbf{j}}$ at storage address Y. An Input Acknowledge signal is then sent on the C-channel. Only $\hat{\mathbf{k}}$ = 3 is permitted.

20 *ADD A*

Add **Y** to the previous content of the Accumulator.

21 *SUBTRACT A*

Subtract **Y** from the previous content of the Accumulator.

22 *MULTIPLY*

Multiply (Q) times **Y** leaving the double-length product in AQ. If the factors are considered as integers, the product is an integer in AQ.

The Branch Condition Designator, j, is interpreted prior to end correction permitting sensing of a product with $(A)_f = 0$. If j equal 4, a skip of the next instruction is made when $(A)_f = 0$. When $(A)_f \neq +0$, a double-length product has been formed with significant bit(s) in the Accumulator; however, if a Skip does occur for j = 4, the Multiply instruction can be re-executed with the same operand and with j = 2 or 3 to determine if Q_{29} contains a significant bit (a *one*) of the product.

In this instruction, k = 7 should not be used.

23 *DIVIDE*

Divide (AQ) by Y leaving the quotient in the Q register and the remainder in the A register. The remainder bears the same sign as the quotient. In this instruction, k = 7 should not be used.

NOTE:

If a DIVIDE FAULT condition exists, no Maintenance 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 a remedial means of noting or correcting the error. Therefore, the instruction which follows the Divide instruction should have its j = 1 in order to preclude the Jump instruction whenever the "Divide Sequence" culminates in a correct answer.
A DIVIDE FAULT can also be detected if the Divide instruction is executed with j = 2. In this case, a correct answer is indicated when a Skip occurs.

24 *REPLACE A + Y*

Add (Y) to the previous content of A. Store (A) at storage address Y.

25 *REPLACE A - Y*

Subtract (Y) from the previous content of A. Then store (A) at storage address Y.

26 *ADD Q*

Interchange (A) and (Q). Then add Y to (A). Interchange (A) and (Q). The content of A is undisturbed by this instruction. The Branch Condition Designator, j, has special meaning in this instruction as in instruction 27.

27 *SUBTRACT Q*

Interchange (A) and (Q). Then subtract Y from (A). Interchange (A) and (Q). The con-

tent of A is undisturbed by this instruction. The 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.*

30 *ENTER Y + Q*

Clear A. Then transmit (Q) to A. Then add **Y** to (A).

31 *ENTER Y - Q*

Clear A. Then transmit (Q) to A. Then subtract **Y** from (A). Finally, complement (A).

32 *STORE A + Q*

Add (Q) to the previous content of A. Then store (A) at storage address Y as directed by the Operand Interpretation Designator, **k**.

33 *STORE A - Q*

Subtract (Q) from the previous content of A. Then store (A) at storage address Y as directed by the Operand Interpretation Designator, **k**.

34 *REPLACE Y + Q*

Clear A. Then transmit (Q) to A. Then add (Y) to (A). Then store (A) at storage address Y.

35 *REPLACE Y - Q*

Clear A. Then transmit (Q) to A. Then subtract (Y) from (A). Then complement (A) and store at storage address Y.

36 *REPLACE Y + 1*

Clear A. Then set (A) = 1. Then add (Y) to (A). Then store (A) at storage address Y.

37 *REPLACE Y - 1*

Clear A. Then set (A) = 1. Then subtract (Y) from (A). Then complement (A) and store at storage address Y.

40 *ENTER LOGICAL PRODUCT*

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 value **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.

41 *ADD LOGICAL PRODUCT*

Add to (A) the bit-by-bit product of **Y** and (Q).

42 *SUBTRACT LOGICAL PRODUCT*

Subtract from (A) the bit-by-bit product of **Y** and (Q).

43 *COMPARE MASKED*

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 the Branch Condition Designator, **j**.

This instruction results in no net change in the content of any operational register. It provides, through the Branch Condition Designator, **j**, a comparison of a portion of **Y** with (A).

44 *REPLACE LOGICAL PRODUCT*

Enter in A the bit-by-bit product of (Y) and (Q). Then store (A) at storage address Y.

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.

45 *REPLACE A + LOGICAL PRODUCT*

Add to (A) the bit-by-bit product of (Y) and (Q). Then store (A) at storage address Y.

46 *REPLACE A - LOGICAL PRODUCT*

Subtract from (A) the bit-by-bit product of (Y) and (Q). Then store (A) at storage address Y.

47 *STORE LOGICAL PRODUCT*

Store in address Y the bit-by-bit product of (A) and (Q) as directed by the Operand Interpretation Designator, **k**.

50 *SELECTIVE SET*

Set the individual bits of A to *one* corresponding to *ones* in **Y** leaving the remaining bits of A unaltered.

51 *SELECTIVE COMPLEMENT*

Complement the individual bits of A corresponding to *ones* in **Y** leaving the remaining bits of A unaltered.

52 *SELECTIVE CLEAR*

Clear the individual bits of A corresponding to *ones* in **Y** leaving the remaining bits of A unaltered.

In this instruction, **k** = 7 should not be used.

53 *SELECTIVE SUBSTITUTE*

Set the individual bits of A with bits of **Y** corresponding to *ones* in Q leaving the remaining bits of A unaltered.

In this instruction, **k** = 7 should not be used.

54 *REPLACE SELECTIVE SET*

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.

55 *REPLACE SELECTIVE COMPLEMENT*

Complement the individual bits of A corresponding to *ones* in (Y) leaving the remaining bits of A unaltered. Then store (A) at storage address Y.

56 *REPLACE SELECTIVE CLEAR*

Clear individual bits of A corresponding to *ones* in (Y) leaving the remaining bits of A unaltered. Then store (A) at storage address Y.

57 *REPLACE SELECTIVE SUBSTITUTE*

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), and set *ones* of this product in corresponding bits of A leaving the remaining bits of A unaltered. Then store (A) at storage address Y.

60 *JUMP (Arithmetic)*

This instruction clears the Program Address Register, P, and enters a new program address in P for certain conditions of either the A- or Q-register content. The Branch Condition Designator, j, is interpreted in a special way for this instruction and thus 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, then Y becomes the address of the next instruction and the beginning of a new program sequence.

- j = 0: No jump. Set 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 is (A) is nonzero.
- j = 6: Execute jump if (A) is positive.
- j = 7: Execute jump if (A) is negative.

61 *JUMP (Manual)*

This instruction clears the Program Address Register, P, and enters a new program address in P for certain conditions of manual JUMP key selections. The Branch Condition Designator, j; is interpreted in a special way for this instruction and thus 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, then Y becomes the address of the next instruction and the beginning of a new program sequence.

Program execution may be stopped by certain STOP selections on execution of this instruction. The Branch Condition Designator, j, specifies which key selections are effective.

- j = 0: Execute jump regardless of key selections.
- 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.

62 *JUMP ON C^n ACTIVE INPUT BUFFER*

This instruction clears the 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; then Y becomes the address of the next instruction. If the buffer in inactive, the Jump condition is not satisfied. The next sequential instruction in the current sequence is executed in the normal manner. $\hat{k}$ = 0, 1, 2, or 3 is permitted.

63 *JUMP ON C^n ACTIVE OUTPUT BUFFER*

This instruction clears the 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; then Y becomes the address of the next instruction. If the buffer is inactive, the Jump condition is not satisfied. The next sequential instruction in the current sequence is executed in the normal manner. $\hat{k}$ = 0, 1, 2, or 3 is permitted.

64 *RETURN JUMP (Arithmetic)*

This instruction executes a Return-Jump sequence for certain conditions of either the A- or Q-register content. The 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) + 1 in the lower half of memory address **Y**. Then jump to **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.

65 *RETURN JUMP (Manual)*

This instruction executes a Return Jump sequence for certain conditions of manual key selections. The 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) + 1 in the lower half of memory address **Y**. Then jump to **Y** + 1.

- **j** = 0: Execute return jump regardless of key selections.
- **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.

66 *TERMINATE C^n INPUT BUFFER*

This instruction terminates the input buffer on channel $\hat{\mathbf{j}}$. No Input Buffer Monitor Interrupt will occur.

The Operand Interpretation Designator, $\hat{\mathbf{k}}$, the Index Designator, **b**, and the Operand Designator, **Y**, bits are not translated for this instruction.

67 *TERMINATE C^n OUTPUT BUFFER*

This instruction terminates the output buffer on channel $\hat{\mathbf{j}}$. No Output Buffer Monitor Interrupt will occur.

The Operand Interpretation Designator, $\hat{k}$, the Index Designator, **b**, and the Operand Designator, **Y**, bits are not translated for this instruction.

70 *REPEAT*

Clear B^7 and transmit the lower 15 bits of **Y** to B^7. If **Y** is nonzero, transmit (**j**) to **r** (designator register), thereby initiating the *repeat mode*. If **Y** is zero, skip the next instruction.

REPEAT MODE - The *repeat mode* executes the instruction immediately following the Repeat instruction **Y** times; B^7 contains the number of executions remaining throughout the repeat mode.

If no Skip condition is met for the repeated instruction, the *repeat mode* terminates. The instruction following the repeated instruction is then 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 B^7. In no way does the *repeat mode* alter a repeated instruction as stored in memory.

The three lower-order bits of the **r** designator (from **j** of instruction 70) affect operand indexing as follows:

r = 0: Do not modify the operand address of the repeated instruction after each individual execution.

r = 1: Increase the operand address of the repeated instruction by one after each execution of the repeated instruction.

r = 2: Decrease the operand address of the repeated instruction by one after each execution of the repeated instruction.

r = 3: Repeat the initial B-register modification of the repeated instruction before each execution.

r = 4: Do not modify the operand address of the repeated instruction after each individual execution. If the repeated instruction is a Replace instruction, the operand address is incremented by (B^6) for the store portion of the Replace Instruction.

r = 5: Increase the operand address of the repeated instruction by one after each execution of the repeated instruction. If the repeated instruction

is a Replace instruction, the operand address is incremented by (B^6) for the store portion of the Replace instruction.

r = 6: Decrease the operand address of the repeated instruction by one after each execution of the repeated instruction. If the repeated instruction is a Replace instruction, the operand address is incremented by (B^6) for the store portion of the Replace instruction.

r = 7: Repeat the initial B-register modification of the repeated instruction before each execution. If the repeated instruction is a Replace instruction, the operand address is incremented by (B^6) for the store portion of the Replace instruction.

NOTE:

Instruction 70 **j** *designator establishes the repeat mode* **r** *designator since* **j** *is transmitted to* **r**.

71 *B SKIP ON* B^n

If the content of B-register **j** is equal to **Y**, skip the next instruction in the current sequence and proceed to the instruction following. Clear B-register **j**.

If the content of B-register **j** is not equal to **Y**, proceed to the next instruction in the sequence in a normal manner. Increase the content of B-register **j** by one.

The 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 in the preceding paragraph.

72 *B JUMP ON* B^n

If the content of B-register **j** is *nonzero* execute a jump in program address to address **Y**. Reduce the content of B-register **j** by one.

If the content of B-register **j** is *zero*, proceed to the next instruction in a normal manner. Do not alter the content of B-register **j**.

The 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, then the lower-order 15 bits of **Y** become the address of the next instruction and the beginning of the new program sequence. The higher-order 15 bits of (Y) cannot be used in this instruction.

73 *INPUT BUFFER ON C^n (without MONITOR Mode)*

This instruction establishes an input buffer via input buffer channel $\hat{j}$ to Magnetic Core Storage with an initial storage address Y. Subsequent to this instruction, 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 during each individual transfer. The next current 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 superseded by a subsequent initiation or termination of an input buffer via the same input channel or until the higher-order half and the lower-order half of storage address 00100 plus $\hat{j}$ contain equal quantities, whichever occurs first.

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. Proceed to the next instruction. $\hat{k} = 2$ is not permitted.

74 *OUTPUT BUFFER ON C^n (without MONITOR Mode)*

This instruction establishes an output buffer via output buffer channel $\hat{j}$ from initial storage address Y in Magnetic Core Storage. Subsequent to this instruction, 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 during each individual transfer. The next current 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 superseded by a subsequent initiation or termination of an output buffer via the same output channel or until the higher-order half and the lower-order half of storage address 00120 plus $\hat{j}$ contain equal quantities, whichever occurs first.

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. Proceed to the next instruction. $\hat{k} = 2$ is not permitted.

75 *INPUT BUFFER ON C^n (with MONITOR Mode)*

This instruction establishes an input buffer via input buffer channel $\hat{j}$ to Magnetic Core Storage with an initial storage address Y. Subsequent to this instruction, 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 during each individual transfer. The next current 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 superseded by a subsequent initiation or termination of an input buffer via the same input channel or until the higher-order half and the lower-order half of storage address 00100 plus $\hat{j}$ contain equal quantities, whichever occurs first. Initiation of this input buffer selects the input channel $\hat{j}$ and establishes a buffer monitor on input channel $\hat{j}$. A Monitor Interrupt follows completion of the buffering operation: $(00100 + \hat{j})_u = (00100 + \hat{j})_L$.

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 $\hat{j}$. Proceed to the next instruction. $\hat{k} = 2$ is not permitted.

76 *OUTPUT BUFFER ON C^n (with MONITOR Mode)*

This instruction establishes an output buffer via output buffer channel $\hat{j}$ from initial storage address Y in Magnetic Core Storage. Subsequent to this instruction, the individual transfers will be executed at a rate determined by an external device. The storage initially established by this instruction will be advanced by one during each individual transfer. The next current 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 superseded by a subsequent initiation or termination of an output buffer via the same output channel or until the higher-order half and the lower-order half of storage address 00120 plus $\hat{j}$ contain equal quantities, whichever occurs first. Initiation of this output buffer selects the output channel $\hat{j}$ and establishes a buffer monitor on output channel $\hat{j}$. A Monitor Interrupt follows the completion of the buffering operation: $(00120 + \hat{j})_u = (00120 + \hat{j})_L$.

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. Proceed to the next instruction. $\hat{k} = 2$ is not permitted.

PART 3

INPUT/OUTPUT SPECIFICATION FOR THE AN/USQ-20 UNIT COMPUTER

1. BASIC INFORMATION

A. *GENERAL*

Communication with the AN/USQ-20 Unit Computer is carried on in a 30-bit parallel mode. The Unit Computer is provided with 14 input channels, which are divided into 12 normal and 2 special input channels, and 14 output channels, which are divided into 12 normal and 2 special output channels. *External Function Codes* are carried over the same 30 lines as are used for output data, but the control signals used with External Function Codes are carried on different lines to indicate the nature of the signals on the 30 lines.

The two *special* input channels and two *special* output channels differ from the normal channels only in timing and control of data transfer. All input/output channels maintain the same electrical specifications; minor modification makes the special input channels identical to the normal input channels. Peripheral equipment which incorporates the features necessary for inter-computer data transfer may also use the special output channels.

Note that all references, in this portion of this section, to input or output are made from the standpoint of the computer; that is, *input* is input *to* the computer and *output* is output *from* the computer.

B. *CONTROL COMMUNICATION*

The AN/USQ-20 Unit Computer is designed to use a d-c level input/output system. Signals are d-c levels which may be changed upon interchange of control information. Signals may exist for microseconds or days, depending on the nature of the particular task.

It should be noted that 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.

C. *DATA AND CONTROL SIGNALS*

Each input and each output channel has its own cable associated with it (28 cables in all). Each cable has 30 data lines plus 3 of a possible 4 control lines, as listed in Table C3-3.

TABLE C3-1. CONTROL SIGNALS IN INPUT AND OUTPUT CABLES

NORMAL INPUT CABLE	*SPECIAL* INPUT CABLE	*NORMAL* OUTPUT CABLE	*SPECIAL* OUTPUT CABLE
Input Data Request	Input Data Request	Output Data Request	Ready
Input Acknowledge	Input Acknowledge	Output Acknowledge	Resume
Interrupt	Interrupt (not used in inter-computer communication)	External Function	(Not Used)
(Not Used)	Input Buffer Active (used only in inter-computer communication)	(Not Used)	Input Buffer Active

Figure C3-4 shows the Unit Computer receiving input from Equipment I and sending output to Equipment II. Of course in most cases, both input and output cables will be used on the same peripheral equipment. Only normal output channels are used for output to peripheral equipment. Any input channel may be used for input from peripheral equipment. Note the direction of information flow. The Data Request signals are always sent from the peripheral equipment to the computer. The Acknowledge signals are always sent from the computer to the peripheral equipment. The third set of control signals, called Interrupt in the input cable and External Function in the output cable is always sent in the same direction as data flow.

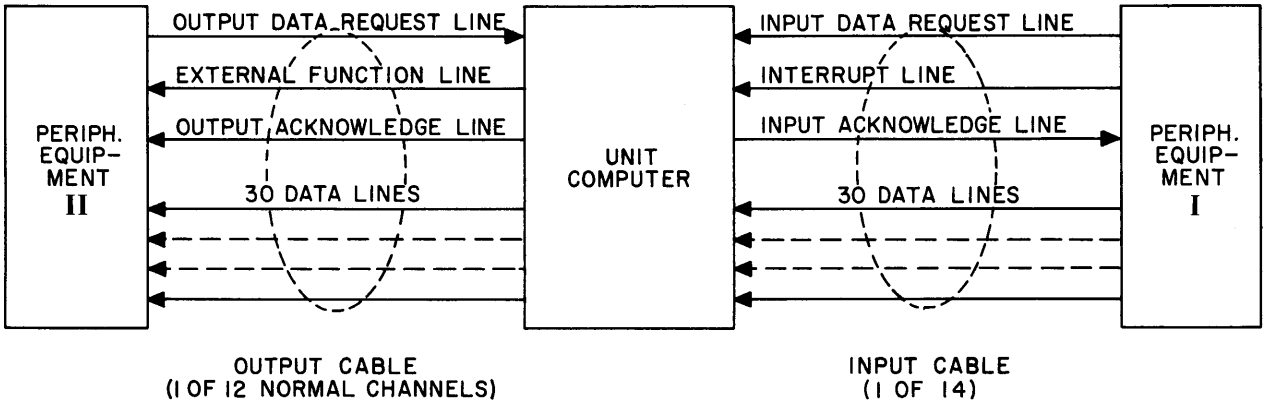


Figure C3-4. Cable Connections

D. *SEQUENCE OF EVENTS*

The sequence of events for each of four cases of communication between the AN/USQ-20 Unit Computer and peripheral equipment follows:

1) Normal Input sequence for data transfer to Unit Computer from Equipment I (Buffer mode):

   a) Computer initiates input buffer for given channel.
   b) Peripheral equipment places data word on 30 data lines.
   c) Peripheral equipment sets the Input Data Request line to indicate that it has data ready for transmission.
   d) Computer detects the Input Data Request.
   e) Computer samples the 30 data lines at its own convenience.
   f) Computer sets the Input Acknowledge line, indicating that it has sampled the data.
   g) Peripheral equipment senses the Input Acknowledge line.
   h) Peripheral equipment drops the data lines and the Input Data Request line.

   Steps b) through h) of this sequence are repeated for every data word until the number of words specified in the input buffer have been transferred.

2) Sequence for Peripheral Equipment I when transmitting an Interrupt code to computer:

   a) Peripheral equipment places the Interrupt code on the 30 data lines.
   b) Peripheral equipment sets the Interrupt line.
   c) Computer detects the Interrupt.
   d) Computer samples the 30 data lines.
   e) Computer sets the Input Acknowledge line, indicating that it has sampled the data.
   f) Peripheral equipment senses the Input Acknowledge line.
   g) Peripheral equipment drops the Interrupt code from the data lines and the Interrupt line.

   Note that the *Input Acknowledge* is the computer response to either an *Input Data Request* or to an *Interrupt*. Thus it is not permissible for a peripheral equipment to interrupt until its *Input Data Request* has been answered, since it would have no way of knowing which signal was being acknowledged.

3) Normal output sequence for data transfer from Unit Computer to Equipment II (Buffer mode):

   a) Computer initiates output buffer for given channel.
   b) Peripheral equipment sets the Output Data Request line indicating that it is in a condition to accept data.

c) Computer detects Output Data Request at its convenience.

d) Computer places information on the 30 data lines.

e) Computer sets the Output Acknowledge line, indicating that data are ready for sampling.

f) Peripheral equipment detects the Output Acknowledge.

g) Peripheral equipment may drop Output Data Request anytime after detecting Output Acknowledge.

h) Peripheral equipment samples the 30 data lines.

i) Computer drops Output Acknowledge and data lines.

Steps b) through i) of this sequence are repeated for every data word until the number of words specified in the output buffer have been transferred.

4) Sequence for Unit Computer when transmitting an External Function Code to Equipment II.

a) Computer places the External Function Code on the 30 data lines.

b) Computer sets the External Function line.

c) Peripheral equipment detects the External Function line.

d) Peripheral equipment samples the 30 data lines.

e) Computer drops External Function Code on the 30 data lines and the External Function line.

E. *USE OF SPECIAL OUTPUT CHANNELS*

Communications between two computers take place using the two special output channels reserved for this purpose. They are connected into a special input channel on the receiving computer. Note that while either a normal or a special input channel may be used for input from peripheral equipment, a *special* input channel is required for inter-computer communication.

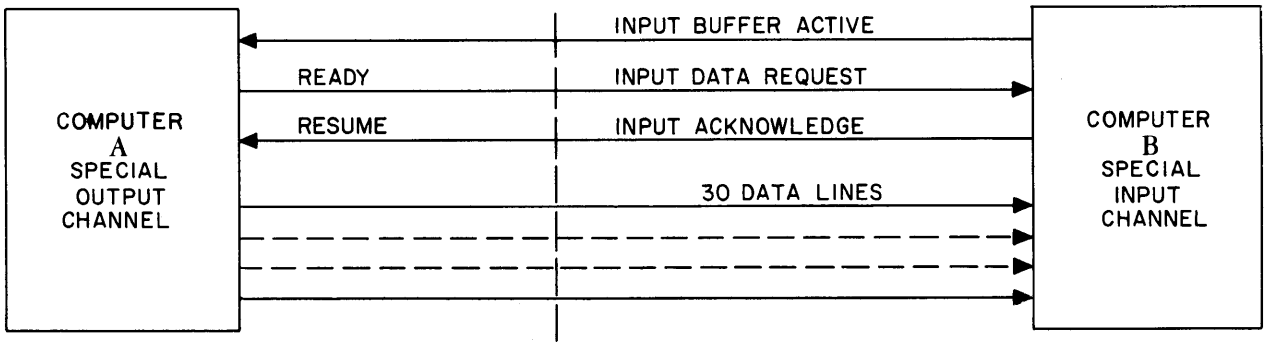


Figure C3-5. Connections from Computer A to Computer B

Figure C3-5 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 going to transmit data to Computer A.

Sequence of events for normal transfer of data from Computer A to Computer B (Buffer Mode):

a) Computer B sets Input Buffer Active signal.

b) Computer A detects Input Buffer Active signal.

c) Computer A places data on 30 data lines.

d) Computer A sets *Ready* which becomes Input Data Request in Computer B.

e) Computer B detects Input Data Request.

f) Computer B samples 30 data lines.

g) Computer B sets Input Acknowledge line (returned to Computer A as *Resume*).

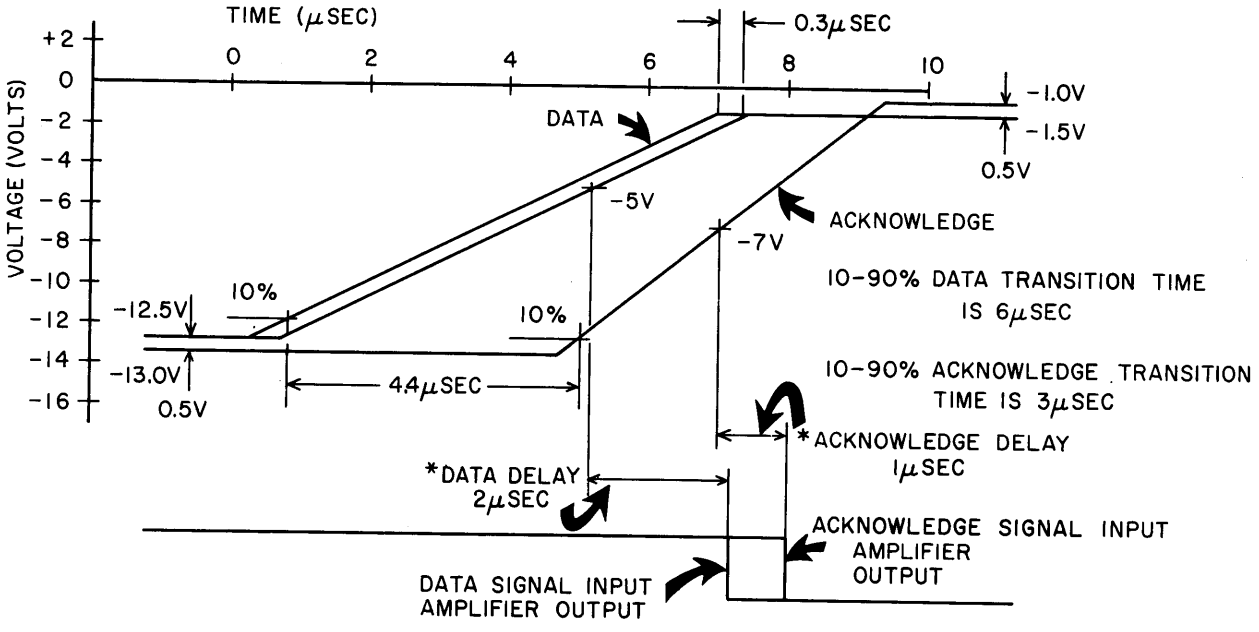
h) Computer A senses *Resume* line.

i) Computer A drops data lines and *Ready* line.

Steps c) through i) of this sequence are repeated for every data word. Input Buffer Active remains energized during entire transfer of block of words.

F. *TIMING*

Data lines, when transmitting data from computer to equipment, *must be stable* before being sampled. Hence, a 4.4-microsecond fixed time delay exists between the computer's loading of an output register and energizing of the Acknowledge signal. Adverse tolerance build-up, causing recognition of Acknowledge signal less than a microsecond after data have reached recognition state, is illustrated in Figure C3-6.



*This delay is due to integration of the input amplifier which is 1.5 μsec ± 0.5 μsec.

Figure C3-6. Effect of Tolerances on Timing

1) Input Timing Considerations

The Input Data Request signal indicates to the Unit Computer that data have been placed on the 30 data lines. The Input Data Request (or Interrupt) must be maintained on the lines until an Input Acknowledge is received. Note that there is no maximum limit on the time the Input Data Request may stay up until being acknowledged. The data lines must remain stable as long as the Input Data Request is up.

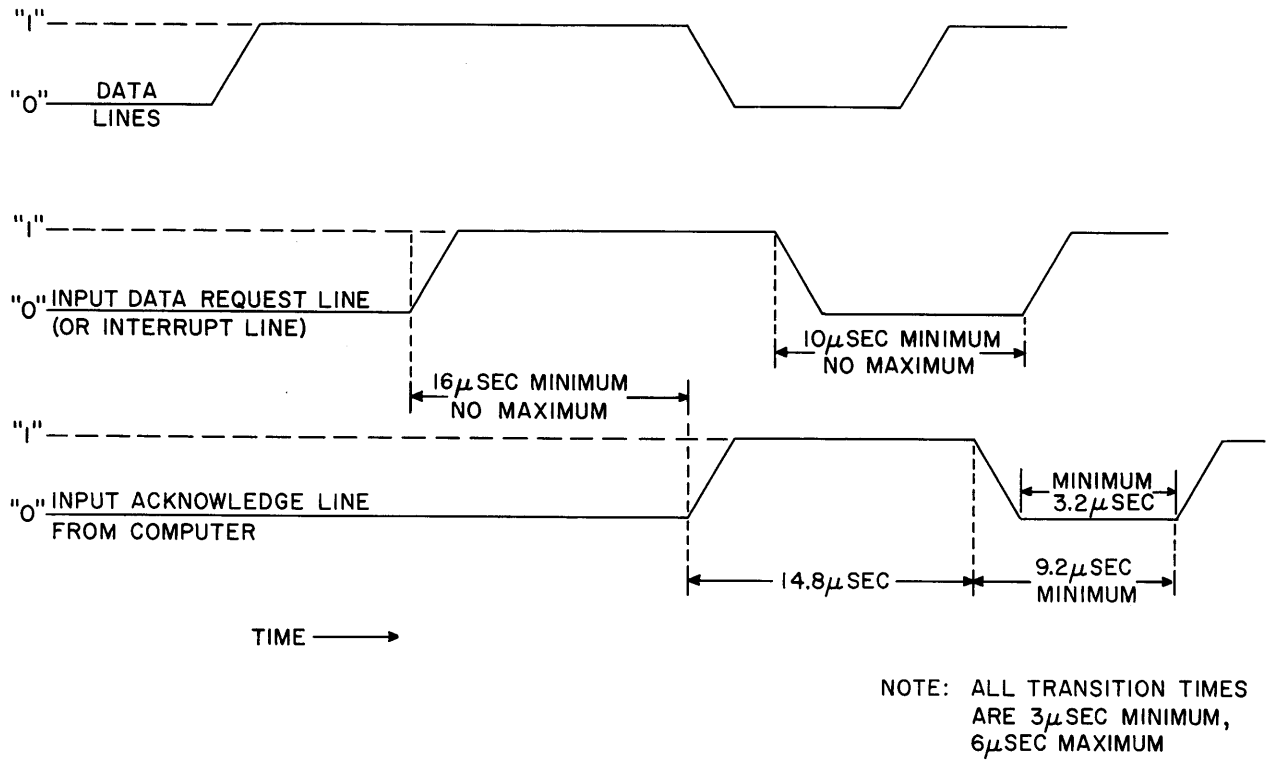


Figure C3-7. Timing of Input Signals

The Input Acknowledge indicates to peripheral equipment that its 30 data lines have been sampled. The Input Acknowledge signal will be set for a fixed period of 14.8 microseconds. Peripheral equipment must be capable of detecting, as an Input Acknowledge, a signal which may exist in a stable *one* state for as little as 8.8 microseconds, allowing for the maximum permissible rise time of 6 microseconds. On sensing the Input Acknowledge, the Input Data Request (or Interrupt) may be dropped to the *zero* state anytime, but it must be dropped to the *zero* state at least 10 microseconds before another Input Data Request can be initiated. Note that the time relationships are such that peripheral equipment wishing to transmit data at a maximum rate could legitimately reset the Input Data Request before the previous Input Acknowledge had dropped to the *zero* state; however, the Input Acknowledge

will always be returned to the *zero* state before being reset to the *one* state. Minimum time for which it will be dropped to the *zero* state is 9.2 microseconds. Allowing for the maximum permissible fall time of 6 microseconds, this leaves 3.2 microseconds minimum time in the *zero* state. These timing relationships are illustrated in Figure C3-7.

2) Output Timing Considerations (Normal Output)

Peripheral equipment must set the Output Data Request line, indicating that it is in a condition to accept data from the Unit Computer. Data will be available to the peripheral equipment for a time interval which may be as short as 23.4 microseconds if the computer is performing output operations at a maximum rate. Data lines need not be cleared to the *zero* state before being reset to the *one* state. The time which may elapse between the request and the data being placed on the line is not fixed, but may vary from 16 microseconds upward, depending upon the computer program, the priority of the particular channel, and the data rates of the other peripheral equipment.

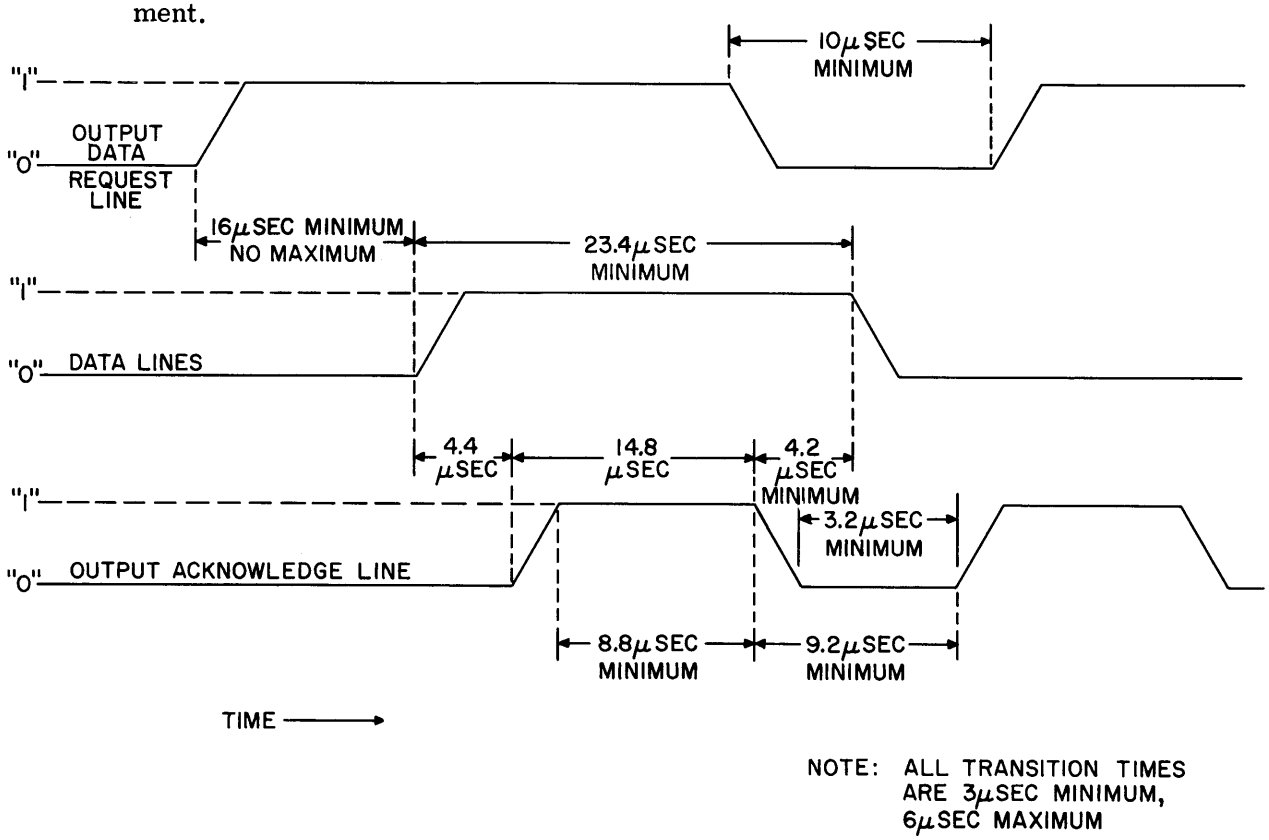


Figure C3-8. Timing for Normal Output Signals

The Unit Computer will put the Output Acknowledge on the line a nominal 4.4 microseconds after placing the data on the line. Timing relationships of normal output

signals are illustrated in Figure C3-8.

The peripheral equipment must sample the data lines within 14.8 microseconds after the Output Acknowledge has been sent. The peripheral equipment must be capable of recognizing, as an Output Acknowledge, a signal which may exist in the stable *one* state for as little as 8.8 microseconds, allowing for a maximum permissible rise time of 6 microseconds. In view of the future desirability of speeding up the output cycle, it is recommended that all new equipment be designed to operate with an Output Acknowledge of 10 microseconds duration which would exist in a stable *one* state for a minimum of 4 microseconds.

The Unit Computer will maintain data on the lines for a minimum of 4.2 microseconds after it starts to drop the Output Acknowledge.

Output Data Request may be dropped anytime after sensing rise of the Output Acknowledge; however, the Unit Computer will not recognize another Output Data Request unless the line is dropped to the *zero* state for at least 10 microseconds. Notice that again, as in the case of input, the timing relationships allow peripheral equipment wishing to transmit data at a maximum rate to reset the Output Data Request before the Output Acknowledge for the previous request has been dropped to the *zero* state. In the worst case, Output Acknowledge will be dropped to the *zero* state for a minimum of 9.2 microseconds, which, allowing for worst case fall time, gives 3.2 microseconds in the *zero* state before being reset.

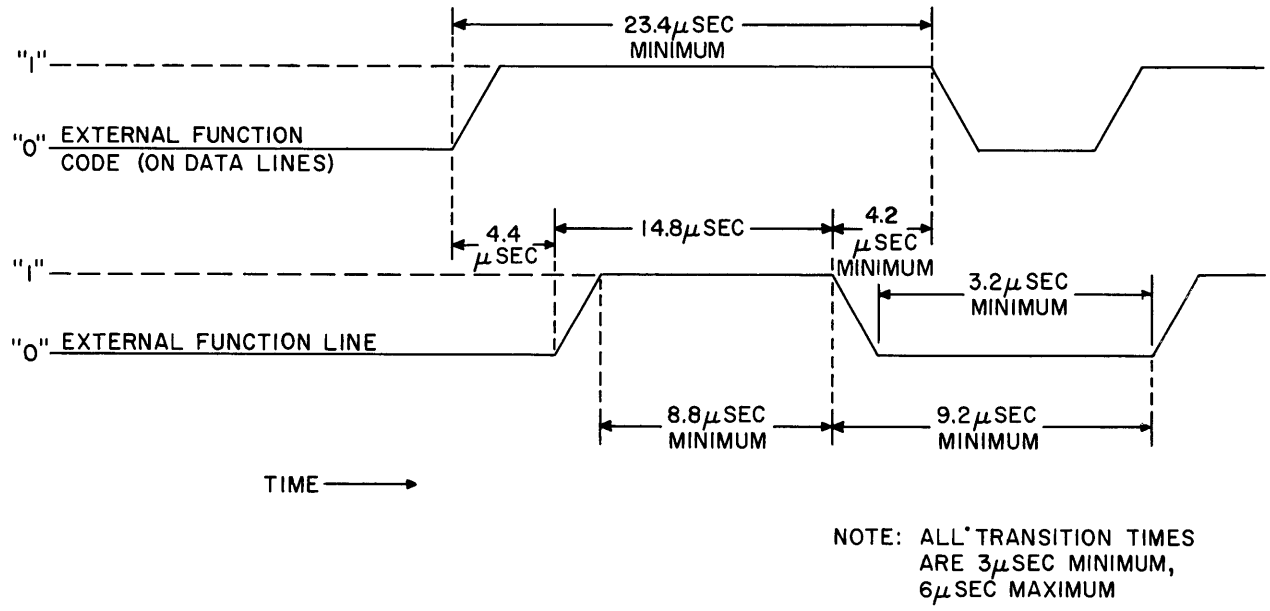


Figure C3-9. Timing for External Function Output

3) Output Timing Considerations (External Function Output)

External Function output timing is singular in that no response is sent by the peripheral equipment. The Unit Computer places the External Function Code on the 30 data lines and follows 4.4 microseconds later with the External Function signal. The External Function remains up for a period of 14.8 microseconds. The data lines remain energized a minimum of 4.2 microseconds beyond dropping of the External Function. The External Function line will be dropped to the *zero* state for at least 9.2 microseconds before being reset. Allowing for the maximum permissible fall time of 6 microseconds, this leaves a minimum of 3.2 microseconds in the *zero* state. Figure C3-9 illustrates these timing relationships.

NOT AVAILABLE AT PRESENT