UNIVAC **DATA FOR COPYRIGHT REGISTRATION**

TITLE

UNIVAC 9200 II/9300/9300 II Systems OS 500 Operating System Programmer Reference, UP-7869 Rev. 1

NAME AND HOME ADDRESS OF AUTHOR

Edward R. Kelly 732 W. Wingohocking St. Phila. Pa.,19140

CITIZENSHIP OF AUTHOR

USA

DATE OF PUBLICATION (RELEASE)

December 1972

NEW WORK

No

REVISION

Yes

TITLE, AUTHOR AND DATE OF ORIGINAL WORK AND/OR PRIOR REVISIONS

UNIVAC 9200 II/9300/9300 II Systems OS 500 Operating System Programmer Reference, UP7869 Edward R. Kelly September 3, 1971

CHANGES FROM ORIGINAL AND/OR PRIOR REVISIONS

See release memo.

NAME AND ADDRESS OF PHOTOENGRAVER

N/A

NAME AND ADDRESS OF TYPESETTER

UNIVAC P. O. Box 500 Blue Bell, Pa. 19422

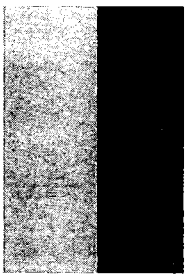
NAME AND ADDRESS OF PRINTER

Same as above.

NAME AND ADDRESS OF BINDER

N/A

UNIVAC
SYSTEMS
PUBLICATIONS



UNIVAC 9200 II/9300
9300 II Systems

REVISION

OS 500 Operating System Programmer Reference

UP-7869 Rev. 1

This UNIVAC 9200 II/9300/9300 II Systems Library Memo announces the release and availability of "UNIVAC 9200 II/9300/9300 II Systems OS 500 Operating System Programmer Reference," UP-7869 Rev. 1. This is a Restricted Distribution Item (RD). Order where required.

This revision contains additions and corrections to the "UNIVAC 9200 II/9300/9300 II Systems OS 500 Operating System Programmer Reference," UP-7869. This document now provides the programmer with the following additional information:

- In Section 3, description of OS 500 error displays; and
- In Section 4, description of the supervisor message display routine, symbiont programming, and job control in the OS 500 operating system environment.

Corrections to the manual have been made in the following major areas: the DTFUQ declarative macro instruction, the //ALR console control message, the online and remote request packet, and the OS5N/OS5C declarative macro instruction. Various other corrections have been made throughout the manual.

Destruction Notice: This revision supersedes and replaces "UNIVAC 9200 II/9300/9300 II Systems OS 500 Operating System," UP-7869 released on the library memo dated September 3, 1971. Please destroy all copies of UP-7869 and/or the associated library memo.

Request additional copies through your local Univac Manager via a Sales Help Requisition Form. These forms are submitted to:

Customer Information Distribution Center (CIDC)
Univac Division Sperry Rand Corporation
P.O. Box 448
King of Prussia, Pa. 19406

H. MASTERS
Group Manager
Systems Publications

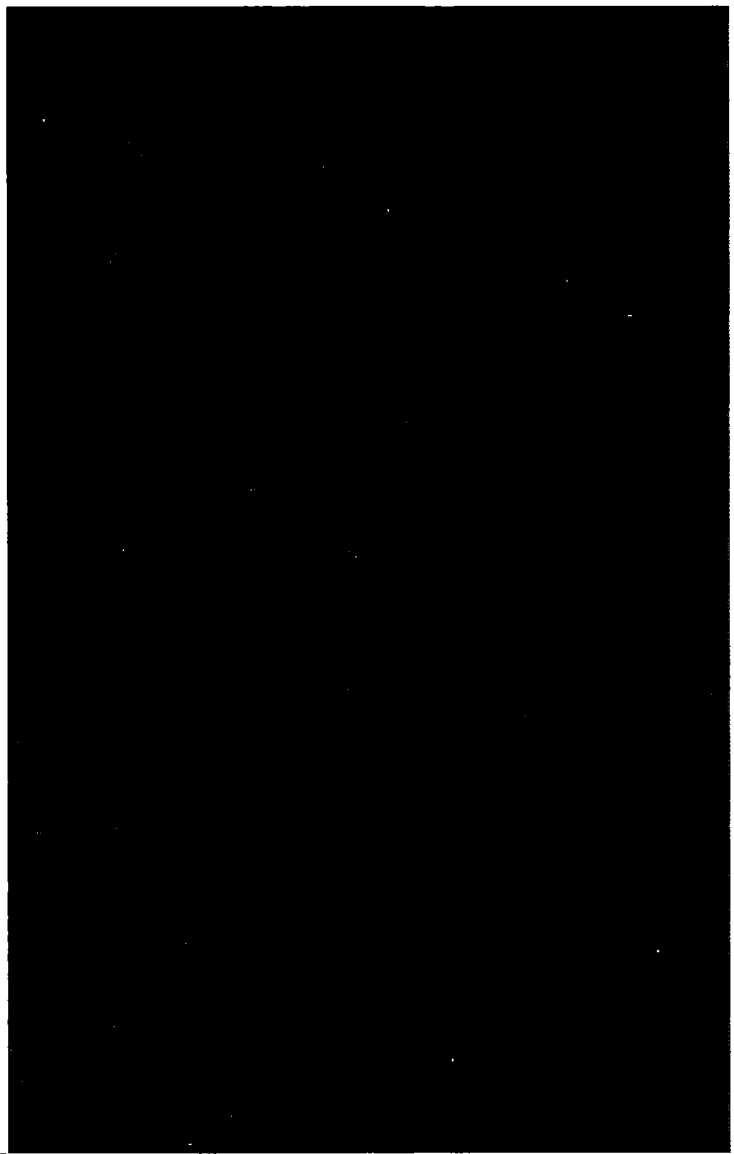
LISTS	ATTACHMENTS	THIS SHEET IS:
217, 630 (less 631E, 634, 635A), 692, 51, 51D, 52, 53, 53D, 54, 54D, 55, 55D, and 56 Library Memo only	UP-7869 Rev. 1 (covers and 66 pages) plus Library Memo to Lists 631E, 634, and 635A	Library Memo for UP-7869 Rev. 1

DATE:
December, 1972

SPERRY RAND

SPERRY RAND

UNIVAC
9200 II/9300
9300 II SYSTEMS
OS 500
OPERATING SYSTEM



PROGRAMMER
REFERENCE

UP-7869
Rev. 1

This document contains the latest information available at the time of publication. However, the Univac Division reserves the right to modify or revise its contents. To ensure that you have the most recent information, contact your local Univac Representative.

UNIVAC is a registered trademark of the Sperry Rand Corporation.

Other trademarks of the Sperry Rand Corporation in this publication are:

UNISERVO

PRINTED IN U.S.A.

PAGE STATUS SUMMARY

ISSUE: UP-7869 Rev. 1

The following table lists the status of each page in this document and indicates the update package (if applicable).

Section	Page Number	Page Status	Update Package
Cover/Disclaimer		Orig.	
PSS	1	Orig.	
Contents	1 thru 3	Orig.	
1	1 thru 3	Orig.	
2	1 thru 22	Orig.	
3	1 thru 6	Orig.	
4	1 thru 20	Orig.	
Appendix A	1	Orig.	
Appendix B	1	Orig.	
Index	1 thru 7	Orig.	

Section	Page Number	Page Status	Update Package

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

I. INTRODUCTION

1.1. GENERAL

This manual describes the UNIVAC OS 500 Operating System, the operating system provided for using UNIVAC console/inquiry units and DCT 500 Data Communications Terminal devices with UNIVAC 9200 II/9300/9300 II Systems. Included in this manual are descriptions concerning hardware/software configurations and components, generation of the OS 500 supervisor, and user and operator interfaces.

Use of this manual assumes prior knowledge of the following manuals: *UNIVAC 9200/9200 II/9300/9300 II Systems Tape/Disc Assembler Programmer Reference, UP-7508* (current version); *UNIVAC 9200/9200 II/9300/9300 II Systems Operating Systems IOCS Programmer Reference, UP-7526* (current version); and *UNIVAC 9200/9200 II/9300/9300 II Systems Operating System Programmer Reference, UP-7531* (current version).

This manual is divided into the following four sections:

- Section 1 contains introductory information required for understanding the format of the macro instructions and the statement conventions used within the manual.
- Section 2 contains information concerning the capabilities of OS 500, minimum hardware and software configurations, and the user inquiry program. A discussion of the environment includes hardware and software components of OS 500.
- Section 3 contains operating information necessary for using the console/inquiry unit both as a console device and as an inquiry device.
- Section 4 describes the components of the OS 500 supervisor and provides the procedures for generating the OS 500 supervisor.

1.2. MACRO INSTRUCTIONS

A macro instruction, like a basic assembler language instruction, has three parts: a label, an operation code, and one or more operands. The label is sometimes optional in a macro instruction, but an operand code and an operand are always present. The operands of a macro instruction are called parameters.

Two types of macro instructions are used. Declarative macro instructions are used for file definitions, with the parameters of the instruction describing all aspects of the file to be processed. Imperative macro instructions are used to communicate with input/output control systems; these instructions cause the processing of the files. The parameters of an imperative macro point to the file to be processed and sometimes add details about the processing.

1.2.1. Declarative Macro Instructions

A user's program must inform the system of the parameters, special conditions, current status, and options which pertain to a file. This is accomplished by using a declarative macro instruction for each file required by the problem program. When these macro instructions are executed, they generate nonexecutable code such as constants and storage for variables. For this reason, declarative macro instructions should be physically separated from the inline file processing coding.

The symbolic name of the file to be processed must appear as the label of the declarative macro instruction. The filename must start with an alphabetical character and must not exceed eight characters.

Declarative macro instructions have operation codes being with DTF, meaning "Define The File . . .". Two more characters are used in the operation code, indicating the type of device or the method of processing used.

Keyword parameters are used as operands for the declarative macro instruction. The keyword parameters used and their specifications define the file. The parameter consists of a keyword or code word followed by an equal symbol (=) and a specification.

Format:

LABEL	ƀOPERATIONƀ	OPERAND
filename	DTFaa	keyword-1 = x,keyword-2 = y, . . . , keyword-n = z

The keyword parameters may be written in any order in the operand field of the instruction. A comma separates one keyword and specification from the next. An alternate form of writing the instruction can be used; in this form, a continuation mark in column 72 is necessary on every line except the last.

Format:

LABEL	ƀOPERATIONƀ	OPERAND	COL. 72
filename	DTFaa	keyword-1 = x,	X
		keyword-2 = y,	X
		•	X
		•	X
		•	X
		keyword-n = z	

1.2.2. Imperative Macro Instructions

The user program must communicate with the input/output control system to accomplish the processing of files that have been defined by the declarative macro instructions. The necessary communication is accomplished by including imperative macro instructions in the user program. These macro instructions are expanded as inline executable code.

A symbol not exceeding eight characters and beginning with an alphabetical character can be used as a label for the imperative macro instruction.

The operation code or verb of the instruction defines how the file is to be processed. The code usually suggests the type of operation to be performed: OPEN, GET, PUT.

The parameters of the imperative macro instruction are used to identify the file to be processed and to further specify the operation to be performed. These parameters are positional parameters; their position in the operand field performs the same function as keywords in declarative macro instructions. Thus, when a certain parameter is used, it must always be specified in the position assigned. If a positional parameter is omitted, the comma which normally separates it from the next positional parameter must be retained. Only trailing commas can be omitted from the operand of an imperative macro instruction. For instance, if an imperative macro used eight positional parameters and only the first, second, and sixth were used, the instruction would appear:

LABEL	bOPERATIONb	OPERAND
[symbol]	xxxxx	positional-1,positional-2, . . . ,positional-n

Not all imperative macro instructions are used for every type of device or file; some imperative macros are specially designed and cannot be applied in all cases. For instance, a PUT imperative macro instruction causes an output operation and cannot be used to process a file from an input device, such as a card reader. The operations controlled and the positional parameters necessary are described for each imperative macro instruction in the appropriate section of the manual.

1.3. STATEMENT CONVENTIONS

The conventions used to illustrate statements in the manual are as follows:

- Capital letters and punctuation marks, except braces, brackets, and ellipses, are information that must be coded exactly as shown.
- Lower case letters and terms represent information that must be supplied by the programmer.
- Information contained within braces represents necessary entries, one of which must be chosen.
- Information contained within brackets represents optional entries that, depending on program requirements, are included or omitted. Braces within brackets signify that one of the entries must be chosen if that operand is included.
- An ellipsis indicates the presence of a variable number of entries.
- In the coding of macros, commas are required after each parameter except after the last parameter specified. When a positional parameter is omitted from within a series of parameters, the comma must be retained to indicate the omission.

2. OS 500

2.1. GENERAL

The UNIVAC OS 500 is an operating system designed to provide console control, inquiry performance, and continuous concurrent operation for users of tape-oriented or disc-oriented operating systems for the UNIVAC 9200 II/9300/9300 II systems. OS 500 provides users with control of system operations from a console/inquiry unit. Operations from the console/inquiry unit provide efficient control of the processing system and fault logging. Inquiry/response capability is available from the local (online) console/inquiry unit or from remote DCT 500 devices. Operations from an inquiry device (an online console/inquiry unit or a remote DCT 500 device) permit the user to interrogate his files for obtaining information when needed. Note that, throughout this manual, the term "online" pertains to the console/inquiry unit which is connected to the processing system by way of the multiplexer channel. The term "remote" means a DCT 500 device connected to the system by means of a data communications subsystem (DCS) and a communications network.

OS 500 is an operating system designed to function in either a concurrent or nonconcurrent environment. A major difference between OS 500 and other concurrent and nonconcurrent operating system (COS and NCOS) environments of UNIVAC 9200/9300 systems is that, with OS 500, the system does not halt upon most contingencies; thus, continuous concurrent operation is provided to the user. Many other components, functions, and operations of COS and NCOS are applicable to OS 500.

2.2. OS 500 CAPABILITIES

OS 500 capabilities are described for the operator's console and for online or remote inquiry/response.

2.2.1. Console Capability

OS 500 affords control of system operations from the console/inquiry unit. Basically, the operator of the console/inquiry unit is provided with the following functions:

- hard copy printout of halt/display codes while permitting operator response without halting the processor;
- processing of inquiry messages entered by way of the console/inquiry keyboard;
- handling of solicited and unsolicited communication between the operator and running programs; and
- hard copy printout and alteration of the contents of a main storage location.

Note that many console/inquiry units can be attached to a system, but only one can be used as an operator's console and inquiry station. Any other directly connected inquiry units are usable as inquiry stations only. Console/inquiry units are not used as remote terminals.

2.2.2. Inquiry/Response Capability

An inquiry/response (logical inquiry) is generally defined as a series of conversational messages transmitted from the time the inquiry device requests and receives control of the system until the device terminates the inquiry by performing the end-of-inquiry (EOQ) function. The logical inquiry is performed by using a given inquiry station to access one or more files in the system one or more times. The number of times user files are physically accessed during a logical inquiry is based on the number of user typeins to his inquiry program. Therefore, when a logical inquiry is made, OS 500 can:

- activate a resident main inquiry program;
- locate, load, and execute a nonresident main inquiry program; or
- activate a resident inquiry symbiont (concurrent environment only).

A resident main inquiry program is one which is user-supplied, is currently loaded in the system, and is always ready to service inquiries and provide responses. Use of a resident inquiry program permits OS 500 to be used as a dedicated inquiry system.

A nonresident main inquiry program is essentially the same as a resident inquiry program except that it is not currently loaded in the system.

Nonresident main inquiry programs can be swapped with the main running program, provided the running program is not an inquiry program. When a logical inquiry is made and the inquiry program is nonresident (it may be on tape or disc, depending on the SYST parameter specified at system generation time), OS 500:

1. suspends the running program and stores its image on the applicable disc subsystem;
2. locates, loads, and executes the nonresident main inquiry program; and
3. upon completion of the inquiry program, restores and resumes the main program.

NOTE:

All peripherals used in the suspended main program are assumed to be conditioned for restart when the program is resumed.

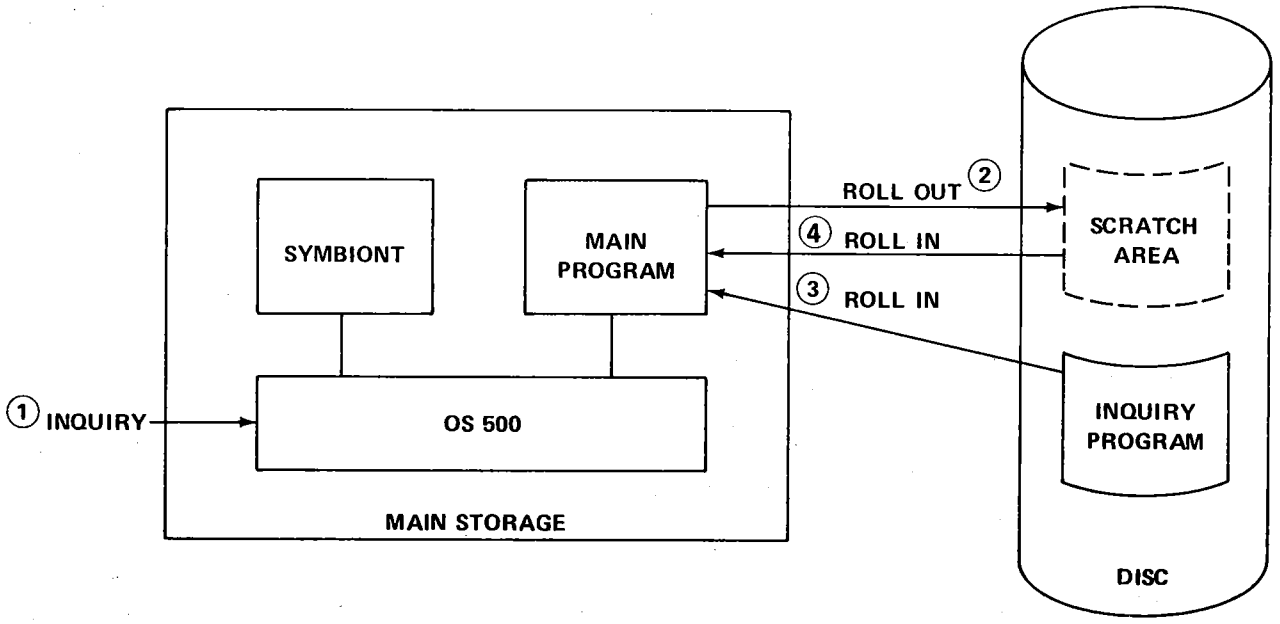
Dumping the currently running main program and preserving all pertinent information for subsequent restoration and resumption of the run is referred to as the rollout/rollin of the main program. For a rollout to a UNIVAC 8411/8414 Disc Subsystem, cylinder 199 is used exclusively by OS 500; for a rollout to the UNIVAC 8410 Disc Subsystem, tracks 98 and 99 are used exclusively. The nonresident inquiry program is located and loaded by means of the standard locator/loader in the tape or disc operating system.

Swapping the main program for the nonresident inquiry program has no effect upon symbionts which may be resident at the time. A tape-to-print symbiont, for example, can continue processing unaffected by the rollout/rollin of the main program.

Figure 2–1 illustrates main program swapping when a logical inquiry is made in a concurrent environment. the main program and a symbiont are running concurrently. When an inquiry occurs:

1. the main program is suspended and rolled out to the disc;
2. a nonresident inquiry program residing on disc is located and loaded;
3. the inquiry is processed by the inquiry program; and,
4. upon completion, the main program is rolled in again and is resumed at its point of suspension.

During this time, the symbiont has continued processing unaffected by the rollout/rollin of the main program. For logical inquiries in a nonconcurrent environment, program swapping is the same except that no symbionts are used.



NOTES:

1. Circled numbers indicate order of occurrence.

2. Inquiry programs may be on disc or tape.

Figure 2–1. Program Swapping in a Concurrent Environment

The user can also inquire by activating a resident inquiry symbiont. Here the main program (a payroll, for example) and an inquiry symbiont are running concurrently. Thus, when an inquiry occurs, it can be handled immediately by the inquiry symbiont already resident in main storage. The main program continues processing unaffected by the logical inquiry.

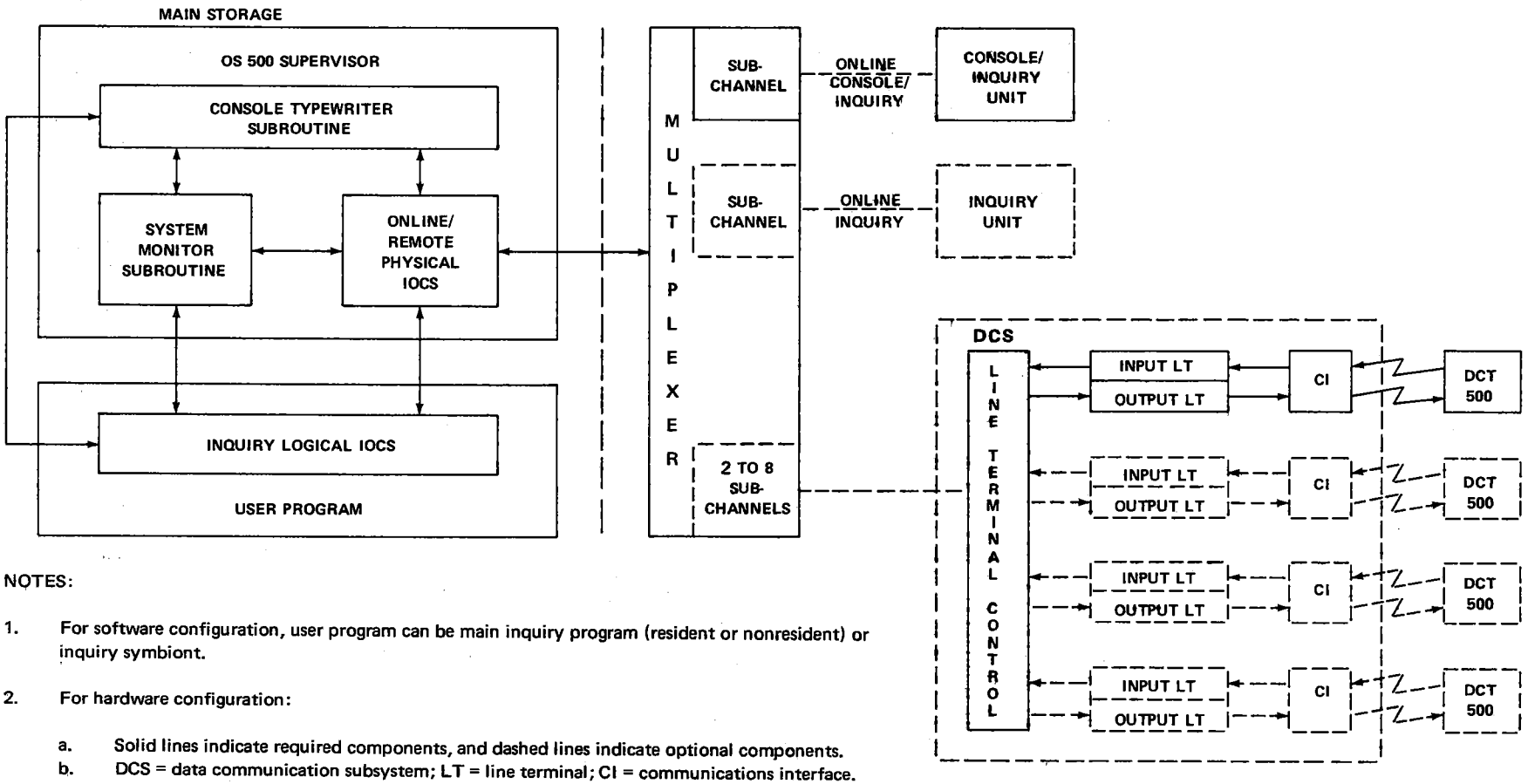
2.3. MINIMUM HARDWARE AND SOFTWARE CONFIGURATIONS

The minimum hardware configuration required for implementation of OS 500 is a central processor unit with a main storage capacity of at least 24K, a card reader, a UNIVAC Console/Inquiry Unit, either a UNIVAC 8410 or 8411/8414 Direct Access Subsystem or a UNISERVO Magnetic Tape Subsystem, and any other peripheral device required by the user problem program.

The minimum software configurations required for OS 500 is the nonconcurrent operating system.

2.4. ENVIRONMENT

The OS 500 is designed to operate in an interactive convcersational mode within an inquiry/response environment. Emphasis is on the speed of interaction. Figure 2–2 shows a functional diagram of OS 500 software components and their interface with DCT hardware components.



NOTES:

1. For software configuration, user program can be main inquiry program (resident or nonresident) or inquiry symbiont.

2. For hardware configuration:

   a. Solid lines indicate required components, and dashed lines indicate optional components.
   b. DCS = data communication subsystem; LT = line terminal; CI = communications interface.
   c. Maximum number of online devices, including one required console inquiry unit, is eight (one per multiplexer subchannel).
   d. Maximum number of DCS configurations is two (Table 2–1); each line requires two subchannels.

   For each DCS:

   – point-to-point communications – one remote device maximum
   – multipoint – 64 remote devices maximum
   – switched network – no device limitation (service limitation based on actual line terminal use, eight maximum at any given time).

Figure 2–2. OS 500 Interface with UNIVAC Online and Remote Devices, Functional Diagram

2.4.1. Hardware Components

The hardware components for operation of the OS 500 are:

- online console/inquiry units;
- DCT 500 remote inquiry units (optional); and
- communications interfaces (optional).

Table 2–1 gives the facility requirements for each hardware component.

Component	Facility Requirements
Online Console/Inquiry Unit	Eight units maximum, one per standard multiplexer channel (one unit for console/inquiry ability and seven units for inquiry capability only)
Remote Inquiry DCT 500	■ DCS-1 or DCS-4; two subsystems maximum (two DCS-1's, two DCS-4's, or one DCS-4 and one DCS-1) ■ Line Terminal Control (LTC); one LTC is required for each DCS: – LTC-1 is used with DCS-1 – LTC-4 is used with DCS-4 ■ Line Terminals – low speed LT provides ring interrupt – automatic answering required for switched network – automatic dialing is optional
Communications Interface	■ Private Line (point to point and multipoint) – Bell Type 103F modem or equivalent – automatic operation interface (F1352-00) ■ Switched Network (point to point) – Bell Type 103A modem or equivalent – automatic interface (F1352-00) – standard interface ■ Transmission Speed (asynchronous) – 110 baud (F1010-06) – 150 baud (F1010-08) – 300 baud (F1010-11)

Table 2–1. Hardware Components and Facility Requirements

2.4.2. Software Components

The necessary software components of OS 500 are:

- System monitor subroutine
- Console typewriter subroutine
- Online physical IOCS
- Remote physical IOCS

(Supervisor)

- Inquiry logical IOCS

Figure 2–2 shows a functional representation of these software components. Descriptions of the user interface with the inquiry logical IOCS are given in the paragraphs that follow. Descriptions of the software components of the OS 500 supervisor are provided in Section 4 to inform the user of their functional operation.

2.5. USER INQUIRY PROGRAM

The user inquiry program interfaces with OS 500 and the inquiry device by way of the inquiry logical IOCS (Figure 2–2). The user-specified declarative macro instruction, DTFUQ, is used to generate the inquiry logical IOCS. Keyword parameters specified with the DTFUQ tailor the logical IOCS to the needs of the user.

The user inquiry program communicates with the logical IOCS package by use of imperative macro instructions, which perform the necessary functions to allow the user to communicate with the requesting device and other devices in the inquiry system. Note that the inquiry program can be as large as available main storage. In a concurrent environment, the maximum amount of main storage used by resident symbionts should be deducted from total main storage capacity since an inquiry program cannot be rolled into an area occupied by symbionts.

The design of the inquiry logical IOCS allows for two methods of inquiry program organization:

1. The inquiry logical IOCS may be assembled with the user inquiry program. In this case, the DTFUQ declarative macro instruction must be the first source statement of the user inquiry program.

2. The inquiry logical IOCS may be linked with the user inquiry program. In this case, the user's inquiry program must be the first element included at link time.

These rules are necessary to allow the system monitor subroutine (4.2.1) and the console typewriter subroutine (4.2.2) to establish the necessary interfaces with the user inquiry program.

2.5.1. Declarative Macro Instruction (DTFUQ)

In order to use OS 500, the programmer must define to the inquiry logical IOCS the file requiring the use of an online console/inquiry unit or remote DCT 500. This is accomplished by means of a DTFUQ declarative macro instruction. Through use of the DTFUQ macro instruction, the programmer is capable of generating an inquiry program which may function either as a main program or as a symbiont.

Format:

LABEL	♭ OPERATION ♭	OPERAND	72
filename	DTFUQ	ACT=label,	X
		BKSZ=n,	X
		EOQA=label,	X
		NAME=label,	X
		TIME=n,	X
		TOUT=label	[X]
		[,CNSL=n]	[X]
		[,CDEV=n]	[X]
		[,ERRO={label / IGNORE}]	[X]
		[,KEY=label]	[X]
		[,NICL=YES]	[X]
		[,REMO=YES]	[X]
		[,SYMB=n]	[X]

The filename entered into the label field of a DTFUQ macro instruction must correspond to the symbolic name assigned to the file by the programmer. Each symbolic name assigned is restricted to four characters in length and must conform to the assembler language rules for labels.

The format of the DTFUQ macro instruction shows each required or optional keyword parameter in alphabetical order. A description of the parameters and their specifications follows, and a summary of the keyword parameters is given in Table 2–2.

- Activity

  This required keyword parameter is used to specify the location where control is transferred when OS 500 alerts the inquiry logical IOCS that an inquiry device is requesting service. The operator of the requesting device types in //RUN,programname.

  Format:

  ACT=label

  where:

  label is the symbolic address where control is to be transferred.

- Block Size

  This required keyword parameter specifies the maximum length of the message transmitted or received.

  Format:

  BKSZ = n

  where:

  n is an unsigned decimal number indicating the number of characters to be transferred.

- End of Inquiry

  This required keyword parameter specifies the symbolic name of the problem program routine to which control passes when the operator initiates an end of inquiry (EOQ) function.

  Format:

  EOQA = label

  where:

  label identifies the EOQ routine address.

- Program Name

  This keyword parameter specifies the symbiont name which will be assigned to the linker command card SYMB or PRGM *(UNIVAC 9200/9200 II/9300/9300 II Systems Tape/Disc Assembler Programmer Reference Manual, UP-75088* (current version)).

  Format:

  NAME = label

  where:

  label is the symbolic address of the location which contains the EBCDIC name of the program. This field is defined as an 8-byte character field.

- Transmission Completion Time

  This required keyword parameter specifies the time permitted by the user for completion of the GET or PUT imperative macro instruction.

  Format:

  TIME = n

  where:

  n is a decimal number which indicates maximum number of seconds allowed before a GET or PUT function is timed out.

- Time Out Address

  This required keyword parameter specifies the symbolic name of the problem program address to which control passes if the time alloted by the TIME keyword parameter expires during execution of an imperative macro instruction.

  Format:

  TOUT = label

  where:

  label is the address of the user's timeout routine.

NOTE:

Control is not passed to the user's timeout routine when the poll function is specified in the CNTRL imperative macro instruction (2.5.2.4).

- Console Logical Unit Number

This keyword parameter specifies the logical unit number of the console device the user desires to use for purposes other than inquiry (input/output of user data).

Format:

CNSL = n

where:

n specifies the logical unit number of the console device expressed as a decimal number. This keyword parameter must be specified when NICL = YES is specified.

- Console Device Number

This keyword parameter specifies the hardware channel assignment of the console device the user desires to use for purposes other than inquiry.

Format:

CDEV = n

where:

n specifies the channel number of the console device; n is expressed as a decimal number.

- Error

This optional keyword parameter is used to specify action to be taken when an input message which exceeds the maximum length allowed is detected. If this keyword parameter is omitted from the declarative macro instruction, an appropriate console message is made, and the program terminates.

Format:

$$\text{ERRO=} \left\{ \begin{matrix} \text{label} \\ \text{IGNORE} \end{matrix} \right\}$$

where:

label is used when control is to be transferred to a user-specified error routine. The label must be the symbolic label of the routine.

IGNORE is used if the user program does not desire special handling of this condition.

- Keyin Table Address

This optional parameter specifies a location which is the base address of a keyin table contained within a user inquiry symbiont (2.6.7).

Format:

KEY = label

where:

label is the symbolic address of the first byte of the keyin table.

- Type of Usage

This keyword parameter specifies that the console is to be used for purposes other than inquiry; that is, using the console to input and output user data as one would use the printer or card reader. This capability can be used only by a main program.

Format:

NICL = YES

When NICL = YES is specified, CNSL and CDEV must be specified. The BKSZ, TIME, and TOUT parameters must also be specified. The ERRO keyword parameter may be specified. The SYMB, EOQA, NAME, KEY, REMO, and ACT parameters must not be specified.

If NICL = YES is omitted, it is assumed that an inquiry IOCS capability is to be generated (2.5).

- Remote Device Capability

This optional keyword parameter is used if remote capability is desired.

Format:

REMO = YES

- Symbiont Number

This optional keyword parameter is used to specify a symbiont number when the activity is to be generated as a symbiont.

Format:

SYMB = n

where:

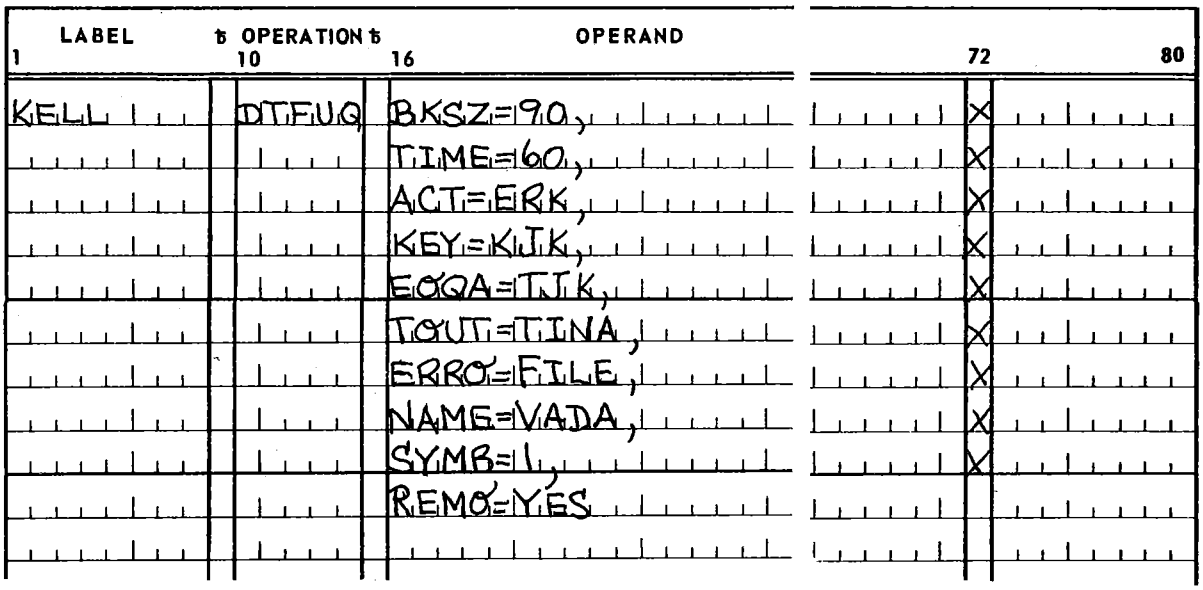
n is a decimal number from 1 through 5 representing the symbiont number. This number must be the same as the number assigned to the symbiont on the linker command card SYMB.

Keyword	Specification	Files		Remarks
		Input	Output	
ACT	label	R	R	Identifies activity address
BKSZ	n = maximum number of characters	R	R	Specifies maximum message length
CDEV	n = device number	X	X	Identifies address of console device used for noninquiry logging
CNSL	n = logical unit number	X	X	Specifies logical unit number of console device used for noninquiry logging
EOQA	label	R	R	Identifies end of inquiry routine address
ERRO	label IGNORE	X		Identifies error routine address or ignores error
KEY	label	X	X	Identifies keyin table address
NAME	label	R	R	Identifies symbolic address containing program name
NICL	YES	X	X	Indicates console logging of noninquiry data
REMO	YES	X	X	Indicates remote device capability
SYMB	n = 1 through 5	X	X	Specifies symbiont number
TIME	n = number of seconds	R	R	Specifies time allowed to complete imperative macro function
TOUT	label	R	R	Identifies timeout routine address

LEGEND: R = Required
X = Optional

Table 2–2. Summary of DTFUQ Macro Instruction Keyword Parameters

Example:



2.5.2. Imperative Macro Instructions

The imperative macro instructions provided to direct OS 500 are OPEN, GET, PUT, CNTRL, and CLOSE. By including the appropriate imperative macro instructions within the inquiry program or symbiont, the user directs OS 500 to open, close, and process files.

2.5.2.1. OPEN MACRO INSTRUCTION

The OPEN macro instruction functions to establish an initial communication link between the console/inquiry unit or DCT 500 device requesting the inquiry and the inquiry program. Performance of this macro enables the transfer of the inquiry device logical unit value from the monitor request packet (4.2.1.1) to the inquiry logical IOCS. The inquiry device is released from system allocation and assigned to the inquiry program.

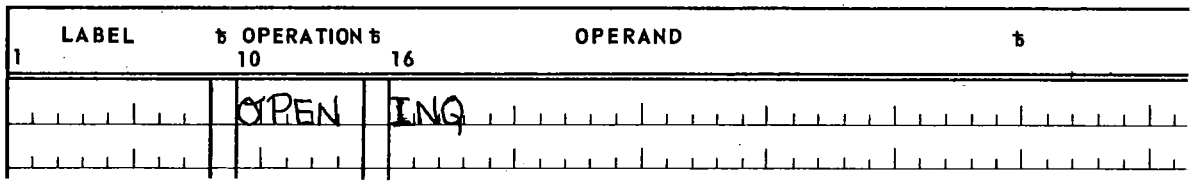
Format:

LABEL	bOPERATIONb	OPERAND
unused	OPEN	filename

Positional Parameter 1:

filename — The symbolic name of the inquiry file as defined in the label field of the DTFUQ macro. The filename may have a maximum of eight characters.

Example:



In the example, the file named INQ is initialized.

2.5.2.2. GET MACRO INSTRUCTION

The GET macro instruction permits the user to receive a forthcoming inquiry message from the specified device. Performance of this macro instruction results in the transfer of one logical record (2.6.4) from the input buffer within the inquiry logical IOCS to the user-defined work area.

If no inquiry message was received within a specified time period, the user is notified.

Format:

LABEL	bOPERATIONb	OPERAND
[name]	GET	filename, workarea

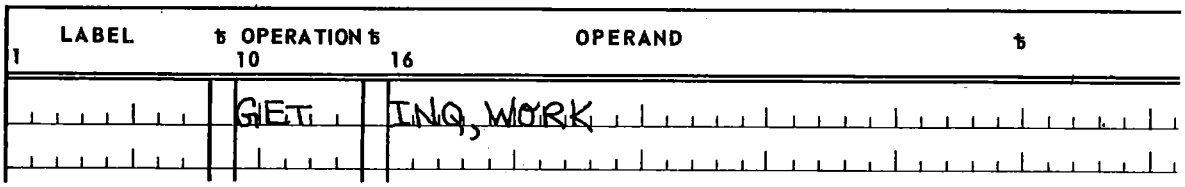
Positional Parameter 1

filename — is the symbolic name of the inquiry file to be accessed as defined in the label field of the DTFUQ declarative macro instruction.

Positional Parameter 2

workarea — The symbolic name of the user-defined storage area where the logical record is made available for processing.

Example:



This example places the record in the area labeled WORK as the next logical input record of the file labeled INQ and described in the DTFUQ instruction.

Operational Conditions:

It is the responsibility of the user inquiry program to determine if the input message is in error. A message error could be determined by record length or by turning the message around to the operator for validation. To abort the message sequence, see 2.6.8.

If the message received is a request from the device operator to terminate the inquiry, control is transferred to a user-specified end of inquiry (EOQ) subroutine. After any necessary close functions are performed, this routine should execute a CLOSE macro instruction.

If the GET macro was issued to an inquiry device which also functions as the console, the possibility exists that a message intended for OS 500 may be routed to the user inquiry program while the inquiry device is under user control. The inquiry IOCS has the ability to recognize these messages and forward them to the operating system.

NOTE:

All outstanding messages must have been answered by the console/inquiry unit operator prior to any rollout/rollin sequence. If not, the requesting station will receive the REJECT PRIORY message.

2.5.2.3. PUT MACRO INSTRUCTION

The PUT macro instruction permits the user to transmit a message to an inquiry device. Performance of this macro results in the transfer of one logical record from the user-designated work area to the output buffer within the logical IOCS.

Format:

LABEL	♭OPERATION♭	OPERAND
[name]	PUT	filename, workarea

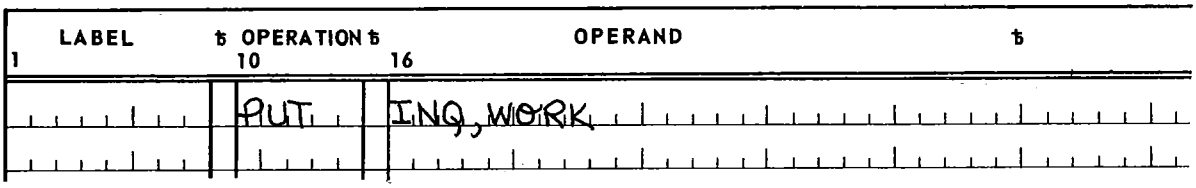
Positional Parameter 1:

filename — The symbolic name of the inquiry file to be accessed as defined in thy label field of the DTFUQ declarative macro instruction.

Positional Parameter 2:

workarea — The symbolic name of the user-defined storage where logical record is made available for output to an inquiry device.

Example:



The record is placed in the area labeled WORK as the next logical output record of the file described in the DTFUQ macro instruction labeled INQ.

Operational Considerations:

It is the responsibility of the user to put the record flag (2.6.5) in the work area. The first two bytes of the 4-byte record flag must contain the record length in binary. The second 2-byte field must be set to binary 0.

2.5.2.4. CNTRL MACRO INSTRUCTION

This macro permits polling of an inquiry transmitting or receiving device. It also permits automatic dialing of a telephone on the switched telephone network and disconnection of the telephone line hookup.

Format:

LABEL	♭OPERATION♭	OPERAND
[name]	CNTRL	filename, {TX / RX}, function-code,time
	DC	Y({workarea / device-id})

Positional Parameter 1:

filename — The symbolic name of the inquiry file to be accessed as defined in the label field of the DTFUQ declarative macro instruction.

Positional Parameter 2:

TX — Indicates that either of the following DCT 500 devices is to be polled:

— keyboard

— paper tape reader

RX — Indicates that either of the following DCT 500 devices is to be polled:

— printer

— Paper tape punch

NOTE:

This parameter must not be omitted when polling the inquiry device.

Positional Parameter 3:

function code — Specifies a decimal number that indicates a function to be performed on an inquiry device. Function codes are:

0 = poll

5 = dial

7 = change time-out value to value defined by positional parameter 4

19 = disconnect

31 = define new inquiry logical unit

Positional Parameter 4:

time — Specifies the time in seconds allowed for completion of the function specified by positional parameter 3.

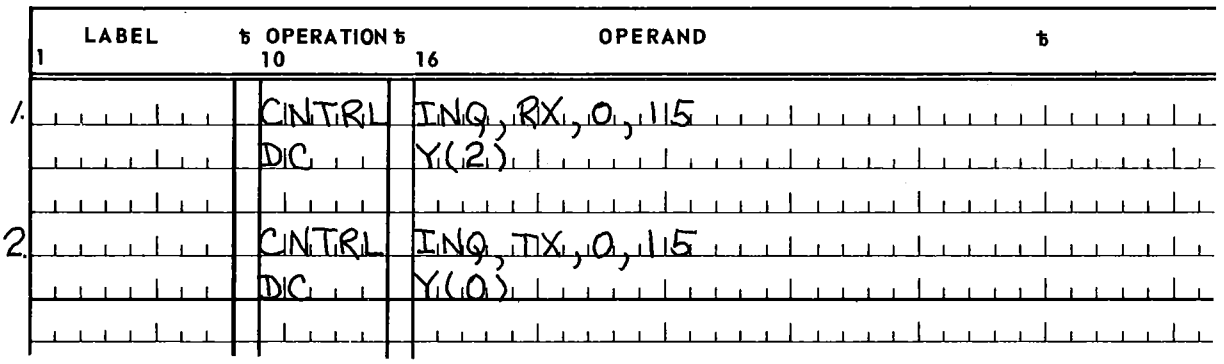
DC Statement

The DC statement must immediately follow the CNTRL macro instruction and must always be specified. Specifications for the operand are:

Y(workarea) — specified for the poll function and the dial function. For the poll function, this operand specifies the work area address which contains a user poll sequence; that is, broadcast operations. For the dial function, this operand is the half-word work area address which contains the direct distance dialing (DDD) network number. The standard record flag is present to identify the number of dial digits.

Y(device-id) — specified for the poll function only if the inquiry device is on a private line. When specified, this operand is the decimal equivalent of the ASCII code for the device identifier. Polling an online console/inquiry or a remote inquiry device on the switched telephone network causes the IOCS to ignore this value.

Examples:



1. The receive device identified by the DC statement is polled for a maximum of 15 seconds.

2. The transmit device is polled for a maximum of 15 seconds. The DC statement indicates that no device ID is specified; therefore, the transmit device is assumed to be the keyboard.

Operational Considerations:

A poll of a device that is online or on a DDD network is a logical poll. For a device on a DDD network, a physical connection is required prior to issuing the logical poll. A poll of a device that is on a private line is a physical poll. Device availability is signalled by the return of an acknowledgment (ACK) character. If the device is unavailable or no reply is forthcoming, a no-acknowledgment (NAK) character is returned. If no reply occurs, the poll times out.

2.5.2.5. CLOSE MACRO INSTRUCTION

This macro instruction causes the release of the inquiry device from the inquiry program to the system. The inquiry device is released when the end of inquiry (EOQ) function, which terminates active status of the inquiry program, is executed. Once released, the inquiry device may be polled by the OS 500 monitor for further inquiry program requests.

After execution of the CLOSE macro, the inquiry IOCS normally reverts to an inquiry request active testing state (an inquiry symbiont assumes a dormant state). However, if the user decides to use the program switch EX?U (2.6.2), control is passed to the user at the address of the source code statement following the CLOSE macro.

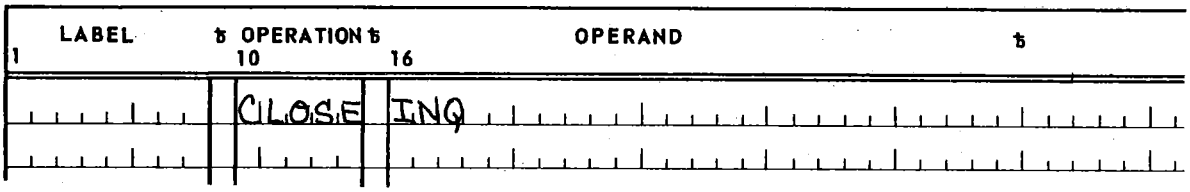
Format:

LABEL	OPERATION	OPERAND
unused	CLOSE	filename

Positional Parameter 1:

filename — The symbolic name of the inquiry file as defined in the label field of the DTFUQ macro. The filename may have a maximum of four characters.

Example:



The file named INQ is closed.

2.5.2.6. SUMMARY OF IMPERATIVE MACRO INSTRUCTIONS

Macro			Remarks
Label	**Operation**	**Operand**	
unused	OPEN	filename	Initializes files
[name]	GET	filename,workarea	Transfers input logical records
[name]	PUT	filename,workarea	Transfers output logical records
[name]	CNTRL	filename, {TX / RX} ,function-code,time	Specifies polling, dialing, or disconnection of a DCT 500 device
	DC	Y({workarea / device-id})	
unused	CLOSE	filename	Closes files

Table 2–3. Summary of Imperative Macro Instructions for Inquiry Devices

2.6. PROGRAMMING CONSIDERATIONS

The following paragraphs explain necessary procedures and required preparations to ensure the error-free completion of the logical inquiry.

2.6.1. Naming Inquiry Programs

When the user writes and names his inquiry program, whether it is a main program or a symbiont, he must assign seven significant characters for the program name. An eighth character must be assigned by the user for use exclusively by the monitor in determining the program type. The eighth character must be specified as either M or S, where M indicates a main program and S indicates a symbiont.

When the inquiry device operator types in //RUN,programname, the system receives the 7-character program name and appends the eighth character to determine the type of inquiry program. First, the character S is appended to determine if the inquiry program is named as a symbiont. The boundary table is scanned for symbiont entries, and each symbiont preamble is examined for this program name. If the name is found, the symbiont is activated. If a symbiont program name is not found, the appended character is altered to M, and it is determined if this main program name is currently the name of the one loaded in the system. If it is, the resident main inquiry program is activated. If the program name is nonresident and rollout/rollin parameters have been specified, the monitor locates and loads a nonresident main inquiry program of the same name with the M as the last character. If the program name cannot be located, the requesting device is so notified. In any case, the user may choose only seven unique significant characters for each program name. Also, names of inquiry programs may not be duplicated, even though they may be different types.

2.6.2. Main Inquiry Program Mode

A main inquiry program may or may not be resident in main storage when the inquiry request is made. A main program which is resident is normally waiting for inquiry activity; that is, the inquiry IOCS is continually testing the inquiry request active indicator of the OS 500 monitor for an on condition. The inquiry request active indicator is bit 1 of the monitor activity byte, which is located at main storage address 290_{10} (4.2.1.4 and Table 4–1). If an on condition occurs, the inquiry IOCS unconditionally transfers control to the user-specified activity entrance. To allow the inquiry IOCS to perform this test function, the user must specify the symbolic tag, IN?U, as the transfer address at program load time (2.7).

The user has the option of designing an inquiry program which functions as a secondary subroutine of a primary resident main program. A user operating under this environment is required to test the inquiry request active indicator of the OS 500 monitor. Prior to transferring control unconditionally to the inquiry IOCS by way of symbolic tag IN?U, the user program primary program element may ensure the return of control to the primary element by setting a program switch within the inquiry logical IOCS (MVI EX?U+1,0). This switch permits the user inquiry program to receive control at the program source code line immediately following the CLOSE macro instruction. Normally, the user program releases control at the performance of the CLOSE imperative macro instruction. Execution of the CLOSE macro allows the inquiry IOCS to revert to a dormant state and await notification of further inquiry program requests.

A main inquiry program which is nonresident at the time of the inquiry program request immediately receives control after the inquiry request active indicator is set. The nonresident main inquiry program is called into main storage following the rollout of the currently resident main program. The monitor (4.2.1) always attempts to locate nonresident main inquiry programs on the system file (SYSFILE) of the system disc pack.

2.6.3. Symbiont Inquiry Program Mode

A symbiont inquiry program must be resident in main storage when the inquiry program request is made. When the symbiont is loaded, it assumes a state enabling the OS 500 system monitor subroutine (4.2.1) to identify it as the recipient of inquiry program request. To accomplish this transfer of control, the symbiont load time address must be to the symbolic tag, IN?U. The inquiry symbiont assumes a dormant state following its initialization.

When an inquiry program request is made, the symbiont assumes an active state. It receives control from OS 500 at this time in I/O state. After the performance of standard housekeeping chores, the user inquiry program receives control in the processor state. The symbiont is not released at this time. The user inquiry program, in addition to its other initialization functions, must open the inquiry file for every logical inquiry (2.2.2) so that the inquiry IOCS can complete the interface with the inquiry device. The performance of a GET, CNTRL, or PUT imperative macro instruction enables the release of the symbiont while awaiting the completion of the function to the inquiry device. Any activity initiated for I/O devices other than the inquiry device delays the release of the symbiont.

2.6.4. Logical Records

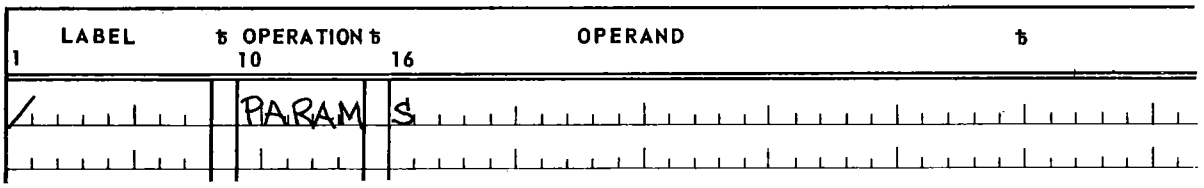
GET and PUT imperative macro instructions cause the transfer of logical records from and to inquiry devices, respectively. For online devices, one logical record is defined as all characters sent from the time the CLEAR TO SEND indicator lights to the time the inquiry device operator presses the CTL and EOT keys simultaneously. For remote devices, one logical record is defined as all characters sent from the start of transmission (STX) characters to the end of transmission (ETX) character. The 4-byte record flag is appended to all records. Half duplex is supported on remote devices; therefore, the line terminal is turned off following reception of the user logical record.

2.6.5. Work Area Size

The calling sequences of the imperative macro instructions GET, PUT, and CNTRL, poll and dial functions, require a work area address. The work area must be large enough to contain at least one logical record. The record format is the standard variable length record format. The first four bytes of the record are designated the record flag. The first two bytes must be a binary count of the record size, which includes the 4-byte flag. The second two bytes are reserved for system use. It is the responsibility of the user to set the record flag prior to the execution of an imperative macro which results in an output function (PUT or CNTRL). The inquiry IOCS sets the record flag prior to returning control to the user after execution of an imperative macro which results in an input operation (GET or CNTRL).

2.6.6. Suspendable Programs

The operating system must be notified when the main program to be executed is suspendable. This is accomplished by including the following control statement in the control stream:



This statement indicates that the program about to be executed is suspendable and that bit 6 of the activity byte (4.2.1.4) is to be set. The user program may set or reset bit 6 of the activity byte; otherwise, it is reset when the EOT supervisor macro instruction is executed.

2.6.7. User Keyin Table

The user keyin table contains addresses of routines for processing unsolicited keyins associated with the inquiry symbiont. The keyin table may contain a maximum of 16 entries, each 2 bytes long.

With OS 500, unsolicited keyins are made from the console/inquiry unit rather than from the DATA ENTRY switches on the processor console (without OS 500). An unsolicited keyin permits the user to transfer control to a user routine within the inquiry symbiont. The base address of this user routine is entered in the user's keyin table. For example, assume that the parameters SYSMB=3 and KEY=TAG are specified in the DTFUQ instruction, and the following user coding is given in the inquiry symbiont:

```
LABEL      OPERATION      OPERAND                    COMMENT
1          10             16
           ENTRY          TAG
TAG        DC             Y(KIN0)
           DC             Y(KIN1)
           DC             Y(KIN2)
KIN0       MVI            LABEL,0
           :
           BC             15,0(1,15)      RETURN TO DTFUQ IOCS
KIN1       MVI            LABEL,1
           :
           BC             15,0(1,15)
KIN2       MVI            LABEL,2
           :
           BC             15,0(1,15)
```

If an unsolicited keyin of X'30' is made, control is transferred to the user routine at KIN0; if the unsolicited keyin is X'31', control is transferred to the user routine at KIN1; if the unsolicited keyin is X'32', control is transferred to the user routine at KIN2. At the end of each user routine, control is returned to the inquiry logical IOCS by way of register 15.

2.6.8. Aborting a Message Sequence

To abort a remote device message, the operator, upon realization of an error, must immediately send the end-of-transmission sequence by simultaneously pressing the CTL and ETX keys. The user receives the remote message as is in his work area. It is the responsibility of the user to determine if the message is in error. Validation of an input message could be determined by record length or by turning the message around and allowing the operator to validate the record.

To abort an online device message, the operator presses the INTRPT key. When the INTRPT indicator lights, the user inquiry program receives control at the symbolic address specified by the TOUT parameter. The user program responds with a CNTRL imperative macro to poll the inquiry device. Meanwhile, the operator presses the PROCEED key, and the INTRPT indicator goes out. This action results in the user inquiry program receiving control at the symbolic address specified by the POLL parameter. The user program then reissues the GET macro, which causes the CLEAR TO SEND indicator on the inquiry device to light.

NOTE:

The operator may abort an inquiry message on an online device by a message validation procedure similar to that described for aborting a remote message.

2.7. INQUIRY PROGRAM EXAMPLE

The following example outlines the organization that the programmer may take to realize the function of a user inquiry program. This example shows an inquiry program for processing online inquiries.

NOTE:

The source coding in this example is intended only as an aid in understanding the general logic of the inquiry program and does not represent any part of a debugged and operational program.

```
LABEL     OPERATION     OPERAND                    COMMENTS
1         10            16
/         EXEC          KSMB
/         DATA          T
IOCS      START         0
          USING         *,0
INQ       DTFUQ         TIME=30,EOQA=END,BKSZ=84,ACT=INQUIRY,TOUT=CHCK
          END           INQU
/*
/         EXEC          KSMB
/         DATA          T
INQUIRY   START         0
          USING         *,0
          OPEN          INQ
*                       USER CODING TO OPEN OTHER FILES,ETC.
MORE      GET           INQ,WORK     GET MESSAGE FROM INQUIRY DEVICE.
*                       USER CODING TO EDIT MESSAGE.
*                       USER CODING TO SET UP REPLY.
          PUT           INQ,WORK     TRANSMIT REPLY.
          BC            15,MORE      GET ANOTHER MESSAGE.
END       EQU           *
*                       USER CODING TO CLOSE FILES.
          CLOSE         INQ
WORK      DS            CL84
CHCK      CNTRL         INQ,TX,0,15  MESSAGE WAS ABORTED OR NOT SENT; POLL.
          DC            Y(0)
          BC            15,END       NO ACKNOWLEDGE; TERMINATE.
POLL      BC            15,MORE      ACKNOWLEDGE ON POLL; TRY AGAIN.
          END
/*
          SWAP          01,02
/         EXEC          DLI          LINKING THE MAIN PROGRAM.
/         DATA          C
          PRGM          INQUIRYM,*   LAST EBCDIC CHARACTER M REQUIRED
*                       FOR MAIN PROGRAM.
          INCLUDE       IOCS,02
          INCLUDE       INQUIRY,02
/*
/         PAUSE         ////
```

In this example, the different elements are linked to form the user inquiry program. Use of the different elements allows rearrangement to accommodate the different modes of operation, that is, main inquiry program mode or inquiry symbiont mode. For symbiont operation, the following statements would replace corresponding statements in the example pertaining to the main inquiry program.

```
LABEL    OPERATION  OPERAND                       COMMENTS                   72
1        10    16
*                                 SYMBIONT OPERATION.
INQ      DTFUQ TIME=30,EOQA=END,BKSZ=84,ACT=INQUIRY,                         X
               SYMB=4,NAME=NAME,TOUT=CHCK
NAME     DC    C'INQUIRYS'        IDENTICAL TO SYMB CONTROL CARD.

*                                 LINKING PROGRAM AS A SYMBIONT.
//       EXEC  DL
//       DATA  C
         SYMB  INQUIRYS,H
         INCLUDE IOCS,02
         INCLUDE INQUIRY,02
/*
```

3. OPERATING INFORMATION

3.1. GENERAL

Operator communications with OS 500 are through the online console/inquiry unit or the remote DCT 500 device. Online and remote inquiry devices are used to submit inquiry requests to OS 500.

OS 500 permits the user of the console/inquiry unit to maintain a hard copy log of fault conditions and operator request/response functions. The OS 500 interface with the console/inquiry unit causes the hard copy printout to conform to current halt/display and request/response values. For example, the hexadecimal stop code normally displayed on the processor console is displayed on the hard copy printout produced by the console printer.

3.2. OPERATION OF THE ONLINE CONSOLE/INQUIRY UNIT

To condition the console/inquiry unit keyboard for a typein sequence, the station operator initially must press the PROCEED key. This action causes an attention interrupt to be generated. The system logical poll then generates a keyboard read function that causes the CLEAR TO SEND indicator on the keyboard control panel to light.

NOTE:

If the CLEAR TO SEND indicator is already lit, the PROCEED key need not be pressed.

Once the CLEAR TO SEND indicator is lit, the operator may begin typing his message. To terminate each message transmitted, the operator presses the CTL and EOT keys on the keyboard. This action causes the CLEAR TO SEND light to go out and the terminating interrupt to be sent.

If the operator makes a mistake, he may press the INTRPT key. On the keyboard control panel, the CLEAR TO SEND indicator goes out and the INTRPT indicator lights. The operator must again condition the keyboard as described above. This recovery procedure is mandatory for obtaining keyboard control.

All system messages require that uppercase EBCDIC characters be transmitted to the processor and that the operator employ the SHIFT key for all alphabetics. Numerics in EBCDIC are transmitted in lower case.

3.3. OPERATION OF THE REMOTE INQUIRY DEVICE

To condition the remote inquiry device for a typein sequence, refer to *UNIVAC DCT 500 Data Communications Terminal Operators Reference, UP-7832* (current version).

3.4. CONSOLE CONTROL MESSAGE SEQUENCES

The console/inquiry unit, regardless of whether it is an inquiry station, may communicate certain special messages to and from the system. These messages may alter and display main storage locations, perform unsolicited keyins, display system messages, and answer system messages. The console device, which must be an online device, is the only device in the system capable of communicating these functions. The console device is also informed of the system status of all other inquiry devices in the system. The console display subroutine routes all system messages to the console device; all answers from the console device are routed to the console display subroutine. Messages which the console operator may type in, as well as the messages and replies displayed to the console operator, are described next.

3.4.1. Altering a Main Storage Location

To alter a main storage location, the console operator types:

//ALR,nn,aaaa,dd...d_n

where:

nn is the number of bytes to be altered; nn may range from 01 through 06

aaaa is the absolute main storage address (hexadecimal) to be altered

dd...d_n is the hexadecimal data to be inserted

When the alter is complete, the console displays the contents of the altered locations.

3.4.2. Displaying Main Storage Locations

To display one or more main storage locations, the console operator types:

//DSP,nn,aaaa

where:

nn specifies the number of bytes to be displayed; nn may range from 01 through 06

aaaa specifies the initial main storage location

The console displays the main storage locations as follows:

! = ddd...d_n

3.4.3. Unsolicited Keyin

To perform an unsolicited keyin, the console operator types:

//KYN,dd

where:

dd is the hexadecimal data byte of the format expected by the standard keyin analysis subroutine. The system processes this keyin as through it were the result of the operator request/keyin method at the processor console.

3.4.4. System Displays

The console display subroutine causes the printout of all system displays on the console printer.

Format:

//MSG,mmmm

where:

mmmm is the 2-byte hexadecimal message value that appeared in the halt/display message on the processor console.

To answer a specific system display, the console operator types:

//ANS,dd,mmmm

where:

dd is the hexadecimal reply byte

mmmm is the 2-byte hexadecimal system display message value

NOTE:

dd = 00 is the reply to an informational message.

3.5. INQUIRY CONTROL MESSAGE SEQUENCES

To initiate an inquiry control message sequence, the operator must:

1. Condition the inquiry station to communicate an inquiry message to the system. Refer to *UNIVAC DCT 500 Data Communications Terminal Operator Reference, UP-7832* (current version).

2. Type in on the keyboard:

   //RUN,programname[,P]

   where:

programname	is a 7-character alphanumeric name assigned to the inquiry program by the user (2.6.1).
P	is an optional character which indicates to OS 500 that this is a priority request and is to be serviced before any nonpriority requests currently queued in the servicing list.

3. Take whatever steps are necessary to transmit the message to the system. Refer to *UNIVAC DCT 500 Data Communications Terminal Operator Reference, UP-7832* (current version).

When the system receives the message, it is validated. If valid, the message is queued in the servicing list. The inquiry device is then notified by the display:

INQRY–QUEUED

When the queued request is serviced, the inquiry device displays:

INQRY–ACTIVE

The device is now available to the user, and he may execute his inquiry operation as he has designed it.

NOTE:

If program swapping is to occur, all messages must be answered prior to the rollout/rollin function.

At termination of the user's inquiry, the inquiry device operator types:

//EOQ

In response to this message, the inquiry logical IOCS performs an end-of-inquiry (EOQ) function.

At the end of the EOQ function, the inquiry device is no longer active to the user. The inquiry device operator is so notified by the display:

INQRY–COMPLTE

To process further inquiries, the inquiry device operator must perform steps 1, 2, and 3.

3.6. SUMMARY OF CONTROL MESSAGE SEQUENCES

Table 3–1 summarizes user input and system response control messages.

	User Input Messages	System Response Messages	
INQUIRY	//RUN,programname[,P]	INQRY-QUEUED or INQRY-ACTIVE or	RJECT-INPUT RJECT-PRIORY RJECT-ABSENT RJECT-MEMORY INQRY-ABORTD
	//EOQ	INQRY-COMPLTE	
SYSTEM	//ALR,nn,aaaa,dd...d_n	dd...d_n	or RJECT-INPUT
	//DSP,nn,aaaa	ddd...dn	or RJECT=INPUT
	//KYN,dd	———	or RJECT-INPUT
	//ANS,dd,mmmm (user response)	//MSG,mmmm ———	RJECT-INPUT

Table 3–1. User Input and System Response Messages

3.7. ERROR MESSAGES AND DISPLAYS

Console message sequences and inquiry message sequences are retrieved and validated by the OS 500 monitor. If the initial message communicated to the system is rejected, a message is displayed on the printer of the requesting inquiry device. In any case, the message sequence must be reinitiated. Error messages displayed at console and inquiry devices are explained in Table 3–2.

Error Message	Reason
RJECT-INPUT	The message initiated by the console or inquiry station operator was either typed incorrectly or transmitted prematurely. In either case, the message cannot be deciphered and must be reinitiated.
RJECT-PRIORY	The requested inquiry program cannot be activated because: ■ it is a symbiont or resident main program which is not loaded; ■ it is a nonresident main program, and the currently running main program is not suspendable; ■ it is a nonresident main program, and program swapping (rollout/rollin) is not part of the operating system; ■ all messages have not been answered prior to rollout of the resident main program.
RJECT-ABSENT	A nonresident inquiry program cannot be located.
RJECT-MEMORY	A nonresident inquiry program cannot be loaded because of main storage limitations.
INQRY-ABORTD	An unrecoverable error occurred in the loading of a nonresident inquiry program. Reinitiate the request.

Table 3–2. UNIVAC Console/Inquiry Error Messages

Table 3–3 lists and describes the OS 500 error displays. In the description of displays 6Eu1 through 6Eu6, if u is the physical unit (PU) number of the system console/inquiry unit, the hexadecimal display is indicated on the HALT DISPLAY indicators on the processor console. If u is the PU number of any other online inquiry unit, the hexadecimal display is typed out on the system console/inquiry unit.

Hexadecimal Display	Meaning and Action
6Eu1	A nonoperational channel indication was received for a console/inquiry unit. Either a software status error or an incorrectly specified hardware address has been encountered. No recovery action is possible. To cancel, key in 0 on the processor console if u was the number of the system console/inquiry unit; if u was the number of any other online inquiry unit, type in 0 on the system console/inquiry unit.
6Eu2	An output bus parity check failure occurred for a console/inquiry unit u. To reissue, key in 0 on the processor console if u was the number of the system console/inquiry unit, or type in 0 on the system console/inquiry unit if u was the number of any other online inquiry unit.
6Eu3	Intervention is required. A hardware error other than a parity check failure occurred. For recovery procedures, see 6Eu2.
6Eu4	An invalid command sequence was attempted. This display indicates a software status error and is nonrecoverable. To cancel, use procedure given for 6Eu1.
6Eu5	The device identification does not match the contents of the PU table for a console/inquiry unit u. This display indicates a software status error and is nonrecoverable. To cancel, use procedure given for 6Eu1.
6Eu6	The device type (first byte of PU entry) is not 0E. This display indicates a software status error and is nonrecoverable. To cancel, use procedure given for 6Eu1.
6Fu1	A nonoperational channel indication was received for a remote inquiry unit u. Either a software status error or an incorrectly specified hardware address has been encountered. No recovery action is possible. To cancel, type in 0 on the system console/inquiry unit.
6Fu2	An output bus parity check failure occurred for a remote inquiry unit u. To reissue, type in 0 on the system console/inquiry unit.
6Fu3	Intervention is required. A hardware error other than a parity check failure occurred. For recovery procedures, see 6Fu2.
6Fu4	An invalid command sequence was attempted. This display indicates a software status error and is nonrecoverable. To cancel, type in 0 on the system console/inquiry unit.
6Fu5	The device identification does not match the contents of the PU table for a remote inquiry unit u. This display indicates a software status error and is nonrecoverable. To cancel, type in 0 on the system console/inquiry unit.
6Fu6	The device type (first byte of PU entry) is not 0F. This display indicates a software status error and is nonrecoverable. To cancel, type in 0 on the system console/inquiry unit.

Table 3–3. OS 500 Error Displays

4. OS 500 SUPERVISOR

4.1. GENERAL

The OS 500 supervisor is that part of the operating system which initiates and controls the execution of the programs necessary to accomplish the user's data processing. The supervisor is generated by issuing a macro instruction; the parameters and specifications used in this macro instruction determine what is included within OS 500.

4.2. SUPERVISOR COMPONENTS

The software components of the OS 500 supervisor are:

- System monitor subroutine
- Console typewriter subroutine
- Online physical IOCS
- Remote physical IOCS (includes local/remote dispatcher)

Functional descriptions of these software components are provided in the following paragraphs.

4.2.1. System Monitor Subroutine

The system monitor subroutine (monitor) is an integral part of the OS 500. It functions to initialize, maintain, and disconnect the interface between the physical IOCS routines of the operating system and the logical IOCS routines of the user program (Figure 2–2). The monitor performs polling, acknowledging, validating, accepting or rejecting, queueing, activating, aborting, terminating the program, and other associated functions of the inquiry system.

The monitor periodically polls the various devices within the inquiry system for an inquiry request. When an inquiry request is received, the monitor validates the request. Should the inquiry request fail the necessary validity checks, it is rejected and the requesting device is so notified. If the inquiry request is valid, it is accepted and queued. The queueing operation places the inquiry request on a servicing list, which allows for inquiry requests specified with priority to be so listed. The monitor notifies the requesting device that the inquiry request is queued.

The monitor services the inquiry request by initiating the action necessary to activate the associated inquiry program. When the inquiry program is activated, the inquiry station is so notified. In all cases, the state of the inquiry request is acknowledged by the monitor.

When the inquiry program terminates or is aborted, the monitor performs the operations necessary to effectively deactivate the inquiry program and return the system to its condition prior to the inquiry. The monitor then continues to service queued requests as the need requires. Where possible, polling and queueing of other devices can continue while a given request is serviced. The system monitor subroutine, therefore, serves as a logical control device within the inquiry system framework of OS 500.

4.2.1.1. MONITOR PACKETS

When the OS 500 supervisor is generated, packets are generated from parameters provided to the system by either the OS5N or the OS5C declarative macro instruction (4.3). These packets, one for each device described as being in the inquiry system, are listed by the monitor for use when logically polling each device; they are also used exclusively by the monitor for communicating with the local and remote physical IOCS. Thus, in addition to logical polling, the supervisor-generated packets are dynamically modified by the monitor to logical read and write functions.

4.2.1.2. MONITOR POLLING

Polling occurs each time the 1-second clock in the processor expires and the resulting interrupt gives control to the monitor. When this occurs, the monitor first attempts a logical poll on each inquiry device currently dormant in the system. The monitor always issues a logical poll, which is serviced by the particular physical inquiry IOCS using the poll packet.

Online and remote dispatchers set status indications in the monitor packets. Finding a packet with active status forces the monitor to retrieve the message request and subsequently to validate and accept or reject the message.

Note that at OS 500 generation time, the address of the monitor clocking subroutine is placed at the very bottom of the clock table. This is necessary because the monitor is not able to return to the clocking scan routine after receiving control; thus, it must be the last entry in the table. As a result, the clocking scan routine accesses each address in the clock list before the monitor gets control.

4.2.1.3. MONITOR QUEUEING

A single queue slot exists for each device specified as being in the system at system generation time. The monitor stores each inquiry request received in the first available slot of the queue table to await servicing. Each time the monitor services a request, it scans the queue table for priority entries, which are serviced first. Otherwise all requests queued are serviced on a first in/first out basis.

When a request is serviced, it is removed from the queueing list, and the remaining entries are floated to the top. Each device in the inquiry system can have but one queued request awaiting servicing in the list at any given time. The monitor accomplishes this by turning off the logical poll of any device which has communicated a request until the request is completely serviced, the EOQ function is performed, and the device is returned to the system. Console message sequences do not become part of the queue list; they are serviced immediately.

The console device continues to be polled even though it may be assigned as an inquiry station and is currently activated to a user. Polling is necessary since the console device is constantly in use for system control. The inquiry logical IOCS is capable of giving system messages to the monitor if the console device is also under user control. The monitor is given control at the message validation subroutine when an SRC 0,48 is issued. If the inquiry logical IOCS is a symbiont, an SRC instruction cannot be issued; instead, a direct (no interrupt) transfer should be employed.

4.2.1.4. MONITOR ACTIVITY BYTE

The monitor activity byte is used by the monitor to keep track of the state of operation of OS 500 and to relay pertinent information to main inquiry programs and inquiry symbionts. The monitor activity byte is at decimal location 290 in the address table of the supervisor. Bit configurations of this byte are described in Table 4–1.

4.2.1.5. ACTIVATING INQUIRY PROGRAMAS

A queued request is retrieved by the monitor from the servicing list, and an attempt is made to pass control to the program named, which specifies the type of program to the monitor (2.6.1). If the inquiry program is a symbiont, which must be resident, it is determined by scanning the highest main storage locations of the boundary table. Bit 5 in the activity byte is set, and the symbiont is activated at the next clock interrupt.

If the inquiry program is a resident main program, it is activated when the monitor sets bit 1 in the activity byte, which is periodically tested by main inquiry programs.

Bit Configuration 0123 4567	Description
1000 0000	Indicates that a main program has been suspended and rolled out and that, on the next clock interrupt, the suspended main program will be rolled back into main storage NOTE: The suspended main program is rolled back only after the nonresident inquiry program has accessed the monitor end of inquiry function (4.2.1.6).
0100 0000	Indicates that queued inquiry request has been serviced and is now available to the user. Resident or nonresident main inquiry programs test the setting of this bit on a user-determined basis to discover the state of unsolicited activity.
0010 0000	Indicates that the monitor is awaiting completion of a GET function; that is, the monitor is in the act of retrieving a message from a device in the inquiry system
0001 0000	Indicates that the monitor is awaiting completion of a PUT function; that is, the monitor is in the act of transmitting a message to a device in the inquiry system
0000 1000	Indicates that a device logically polled by the monitor has responded with active status, which indicates that the operator of the station wishes control of the keyboard
0000 0100	Indicates that an inquiry symbiont is resident and is about to receive control after an inquiry request is retrieved from the servicing list
0000 0010	Indicates that the currently running main program is suspendable if a nonresident main inquiry program is used to handle inquiry requests
0000 0001	Indicates that a console/inquiry device is currently busy completing a system control sequence of operations

NOTE:

More than one bit of the activity byte may be set at any one time.

Table 4–1. System Monitor Subroutine, Activity Byte

Bit 1 is always set when the monitor activates an inquiry program, whether it is a main program or a symbiont. If the inquiry program is nonresident, bit 6 of the activity byte is tested. If bit 6 is set, the main program is rolled out, the inquiry program rolled in, bit 0 in the activity byte is set, and control is transferred to the new start address. If bit 6 is not set, the request is rejected because the currently running main program is critical. The address of the packet containing the logical unit number of the inquiry or console device and other information needed by the physical IOCS is stored in the address at decimal location 286.

4.2.1.6. MONITOR INQUIRY SEQUENCE TERMINATION

When a user inquiry sequence is terminated, the system is notified of the end of inquiry (EOQ) by a typein of //EOQ, which is handled by the inquiry logical IOCS. The result is a transfer of control to the monitor's end-of-inquiry address. If the inquiry program is a main program, the following SRC instruction is executed:

```
SRC 0,50
```

If the inquiry program is a symbiont, the appropriate address in the SRC table is accessed directly.

When the monitor gets control at its EOQ address, the device from which the inquiry originated is no longer available to the user. The monitor resumes logical polling of that device and restores it to an initialized condition. At this time, the monitor also searches the servicing list for another request for the same program. If one exists, the request receives control; if none exists, the monitor continues logical polling. If a main program has been suspended and rolled out, rollin occurs at this time. Testing bit 0 of the activity byte (Table 4–1) determines this condition.

4.2.2. Console Typewriter Subroutine

The console typewriter subroutine prints out current halt/display values and allows the operator at the console device to key in response values to the operating system or to the user program. This subroutine also performs the operations necessary to:

- allow the console device operator to alter and display main storage locations, and
- permit the console device to act as an operator request station.

To provide these abilities, the console typewriter subroutine interfaces with the system monitor subroutine and the online physical IOCS. The console typewriter subroutine permits the OS 500 user to maintain a hard copy log of system conditions without halting the operation of the processor.

4.2.3. Online Physical IOCS

The online physical IOCS controls any and all input/output orders directed to or from any online console/inquiry unit within the OS 500 inquiry system. The online physical IOCS provides a logical interface between OS 500 and a user inquiry by means of the standard multiplexer channel (Figure 2–2). It performs such functions as logical polling, keyboard reading, and printing. The online physical IOCS attempts error recovery when and where feasible. It informs the issuing routines of the condition of orders listed and of any outstanding online error conditions as they may be encountered.

4.2.4. Remote Physical IOCS

The remote physical IOCS controls any and all input/output orders directed to or from any remote devices within the OS 500 inquiry system. The remote physical IOCS provides a logical interface between OS 500 and a user inquiry program and remote DCT 500 devices connected to the system. All remote DCT 500 devices are connected to the system by means of a data communications subsystem (DCS) and an appropriate communications interface (Figure 2–2). The remote physical IOCS performs such functions as logical/physical polling, reading, writing, and dialing. Error recovery is attempted when and where possible within the framework of a communications environment. Issuing routines are made aware of the condition of orders listed and of any outstanding remote error situations.

4.2.5. Local/Remote Dispatcher

The local/remote dispatcher portions of OS 500 are a function of the online/remote physical IOCS. The dispatcher controls all input/output functions to and from local/remote inquiry devices in the system. It performs the functions necessary to execute I/O requests, handle resulting interrupts, and perform error recovery.

The dispatcher maintains a chain list for each device assigned a specific channel number on the multiplexer. These chain tables contain slots which either point to the first device request packet (4.2.5.4) in the list (top of the chain) or which contain an end-of-chain flag (the value X'8000'). Each packet in the chain list points to the next packet in the chain until the end-of-chain flag is encountered. The length of each chain is theoretically limitless.

The local/remote dispatcher has an entry in the clocking table for the purpose of decrementing the timeout value used in the current packet of each chain list (Figure 4–1). If the timeout value expires, current status of the packet is altered to indicate that the clock timed out. Note that the timeout value is not used for keyboard read functions because once the CLEAR TO SEND indicator is lit on the online device, it can be turned off only by operator action.

An online device cannot be physically polled. The local dispatcher therefore simulates the poll by placing the poll packet on the chain without an associated issue instruction. If the operator presses the PROCEED key while the poll packet is on the top of the chain, the packet gets active status. If the PROCEED key is not pressed while the poll packet is on the top of the chain, a timeout eventually occurs. Remote devices are physically polled.

4.2.5.1. REQUEST PROCEDURE

The local/remote dispatcher portion of OS 500 controls all input/output functions to and from local/remote devices in the system. An input/output request is forwarded to the dispatcher in the form of the address of a 22-byte request packet. This packet contains the information necessary to create the desired function and also provides the working storage area which contains status information and a chaining address. The local/remote dispatcher adds the request packet to a list of other outstanding packets it may be servicing; the status of the request packet at this time indicates that it is not serviced. The request packet may not be altered in any way from the time it is given to the dispatcher until some appropriate status is indicated. The request packet may be issued again only if the request has been completed and the dispatcher has released it. By testing the status indicator within the request packet, the problem program can determine when and if the request has been completed. The physical location of the request packet does not change.

4.2.5.2. PROBLEM PROGRAM PROCEDURE

The problem program submits an input/output request by storing the address of the packet into register 15 and issuing an SRC instruction as follows:

```
LH      15, packet-address

SRC     0,c
   or
SRC     0,52
```

where:

c has a value that is twice the number of the channel on which the online device is located.

Once submitted, a packet should not be resubmitted until it has been serviced by the local/remote dispatcher.

When the local/remote dispatcher, after receiving a request, finds the chain for that channel empty or, through an interrupt, determines that an XIOF may be executed, it examines the request at the top of the chain. The dispatcher then creates the input/output order by using the logical unit number supplied in the request packet to reference the LU table, which, in turn, points to the physical unit (PU) assigned to the file in the PU table.

4.2.5.3. ERROR RECOVERY

After servicing each request packet, the local/remote dispatcher sets any appropriate status bits in byte 6 (Figure 4–1). Errors which may be encountered on particular online peripheral devices are displayed on the console device by way of the message display subroutine. If a catastrophic error occurs on the console/ inquiry unit or a remote inquiry device, a halt/display occurs (3.7).

4.2.5.4. ONLINE/REMOTE DEVICE REQUEST PACKET

The request packets for the online and remote devices are generated from parameters provided to the logical IOCS by the DTFUQ macro instruction. Each packet contains 22 bytes of information and must be positioned on a half-word boundary. The request packet is forwarded to the dispatcher at issue time. The format of the request packet is shown in Figure 4–1. Detailed descriptions of the contents of the request packet follow the illustration.

NOTE:

In the descriptions of the request packet contents, the symbols (L), (R), and (C) are used to indicate that the information is for the local dispatcher only, for the remote dispatcher only, or common to both the local and the remote dispatchers, respectively.

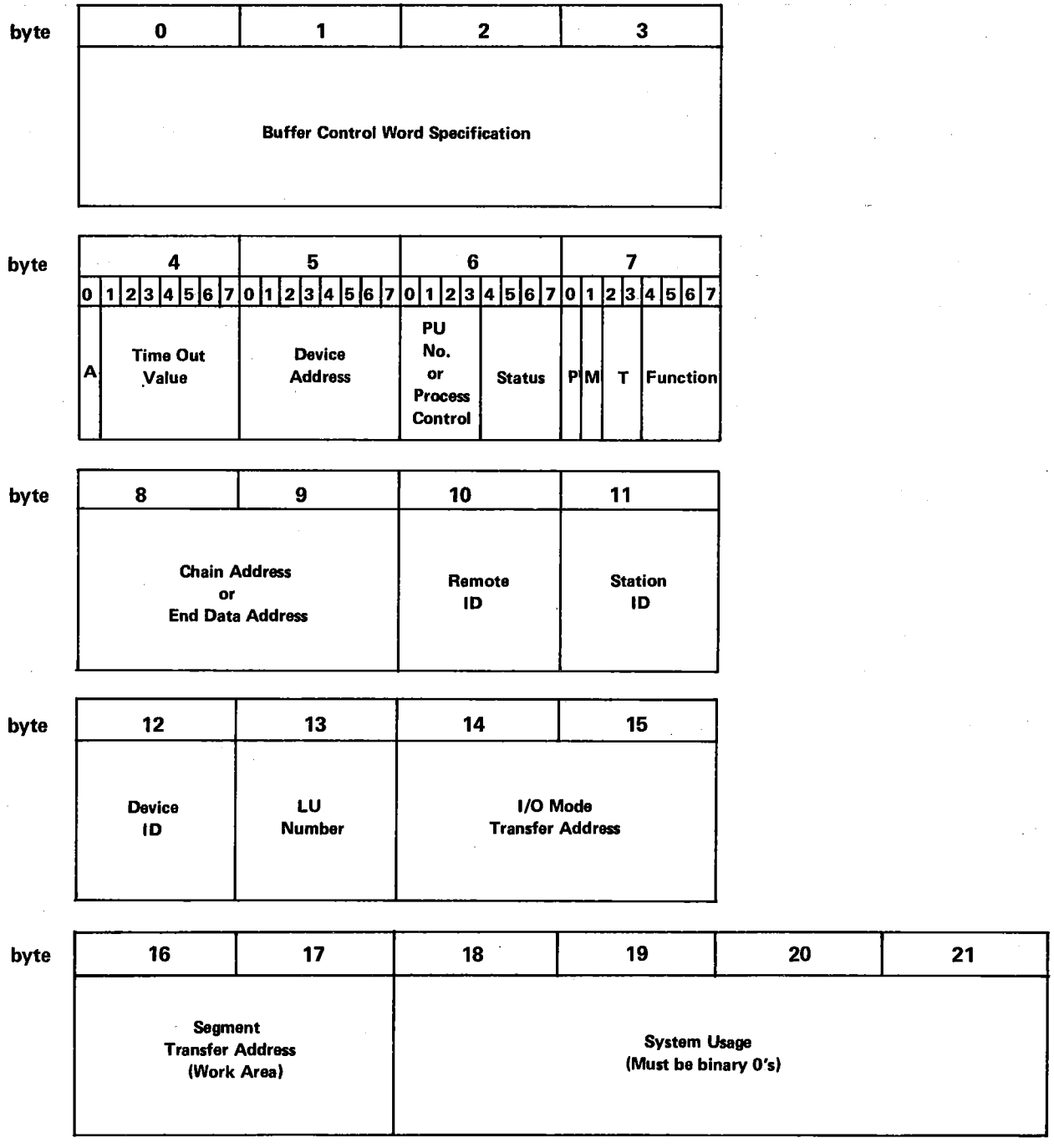
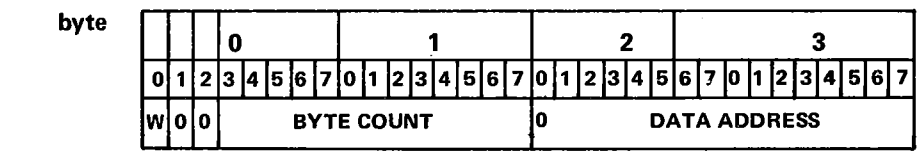


Figure 4–1. Online and Remote Device Request Packet

- Bytes 0–3 (C)

  These bytes specify the information required to load the buffer control word (BCW). The format of the BCW is:



– Byte 0

Bit 0 W: Data direction bit
0 = Read (input) operations
1 = Write (output) operations

Bits 1, 2 Always 0

– Bytes 0 and 1

Bits 3–7, Bits 0–7: Byte count: A binary number specifying the number of bytes to be transmitted or received.

– Byte 2

Bit 0 Always 0

– Bytes 2 and 3

Bits 1–7, Bits 0–7: Data address: The address of the data area. The length of the data area is specified in the byte count field. For remote dispatcher control, this field indicates the communications buffer address (CBUF).

NOTE:

For remote devices, CBUF must be a multiple of 128. This can be accomplished by using the ORG assembler directive and the MOD linker directive.

■ Byte 4

Bit 0 A: Activity bit (L) – used by the dispatcher for local control.

Bits 1–7 Timeout Value (C): Specifies the amount of time in seconds for completion of this packet. The number specified by the user may not exceed decimal 127.

■ Byte 5

Device address (C): specifies the device address for this logical unit. The dispatcher supplies this byte when the packet becomes chained in the dispatcher's servicing list.

■ Byte 6

Bits 0–3 PU number (L): half-byte specifying the physical unit number of the device associated with this packet. The local dispatcher supplies this byte when the packet becomes chained in the dispatcher's servicing list.

Process control (R): the remote dispatcher uses this field for function and process control as follows:

1000 – active packet (XIOF issued)
0100 – status poll answered (ACK character received)
0010 – to turn off CLEAR TO SEND indicator after input of keyboard data (T bit software control)
0001 – ignore B bit interrupt (in communications hardware BCW)

Bits 4–7 Status: half-byte indicating the current status of this packet. Initially, the status indicates that the packet is serviced by the dispatcher. When serviced, the appropriate status is reflected in this half byte. Status values are:

0000 – not serviced (C)
0001 – software error (C)
0010 – hardware error (C)
0011 – activity on poll (C)
0100 – no activity or timeout (L)
0101 – aborted (L)
1111 – completed (C)
0110 – dialing operation not accepted (R)
1000 – wrong length transfer error (C)

- Byte 7

Bit 0 P: Priority bit (C) – set by the user to indicate that the packet is to be serviced with priority. All system packets are issued with priority; IOCS packets are not.

Bit 1 M: Mode bit (R) – set by the user to indicate that the packet is for a remote device. This bit is never set for local device packets.

Bits 2 and 3 T: Type bits – set by the user to indicate the type of requesting routine. Possible settings are:

00 – routine that requests to regain control in I/O mode upon completion of servicing the packet (C)

nonzero – routine that does not need to take control in I/O mode (C)

Bits 4–7 Function: suppled by the user and indicating the operation to be performed on the device. Possible function codes in hexadecimal are:

0 – logical poll (L)
1 – logical write (C)
2 – logical read (C)
3 – turnoff (R)
5 – dial (R)
D – physical disconnect (R)

- Bytes 8 and 9

Chain address or end data address (C): set by the dispatcher. Prior to servicing the packet, these bytes contain the address of the next packet in the chain (or the end-of-chain flag). After servicing, these bytes contain the end-of-data address indicating the location of the last data byte transmitted or received.

- Byte 10

Remote ID (R): set by the user to indicate the remote identifier of multipoint devices not online to the system. This byte is used only when the mode bit (byte 7) is set.

- Byte 11

Station ID (R): set by the user to indicate the station identifier of the DCT 500 device operating remotely. This byte is used only when the mode bit (byte 7) is set.

- Byte 12

  Device ID (R): set by the user to indicate the device identifier (such as for the keyboard, printer, or paper tape) of the remote DCT 500. This byte is used only when the mode bit (byte 7) is set.

- Byte 13

  LU number (C): supplied by the user to indicate the logical unit number of the device associated with this packet.

- Bytes 14 and 15

  I/O mode transfer address (C): present only if the user wishes to take control in I/O mode, when servicing of the packet has been completed. Bits 2 and 3 of byte 7 must be OFF if these bytes are to be significant. For local devices, if the user does not need to take control in I/O mode, the packet may be limited to 14 bytes. For remote devices, these bytes must be present and must contain binary 0's if the user does not need to take control in I/O mode. Inquiry symbionts would normally place a transfer address in these bytes. Other symbionts whose use of the console is for purposes of occasional operator messages would probably not need to suspend themselves pending servicing of a console packet and would not place a transfer address in bytes 14 and 15.

- Bytes 16 and 17

  Segment transfer address (R): the communications buffer is loaded or unloaded in 64-byte (or less) segments; therefore, these two bytes are set by the user to indicate the origin or destination of the first segment. (The communications buffer size is 128 bytes.) The segment transfer address is updated by the remote dispatcher as each segment is transferred.

- Bytes 18, 19, 20, 21

  System usage bytes (R): must be binary 0's.

4.2.5.5. SUPERVISOR MESSAGE DISPLAY ROUTINE

Certain details of the supervisor message display routine (E?DS) within OS 500 differ from the usage under versions of nonconcurrent operating systems (NCOS) and concurrent operating systems (COS).

- Return of Control

  When a routine operating in I/O mode executes a branch-and-link (BAL) instruction to the supervisor display routine, control is not returned to the routine until the packet containing its display has been serviced by the OS 500 monitor. While this packet is being serviced, control is returned in processor mode to the address contained in the processor state control (PSC). This implies that user I/O mode routines should execute all processing which might affect related processor mode routines before branching to E?DS. Upon return from E?DS, they should reenter the processor mode via the supervisor reentry routine (E?RE) without attempting further processing.

- Reply

  When a routine operating in I/O mode requests that the system display a message, a value of X'01' is placed in location 4 before returning control to the requesting routine. This is for purposes of compatability with UNIVAC standard IOCS routines which may operate under either OS 500 or nonconsole operating systems; the reply of X'01' signifies a request for delayed error recovery. Therefore, user routines which need to obtain a meaningful reply to a display should switch to processor mode and execute an MSG macro instruction to obtain a reply.

4.2.5.6. SYMBIONT PROGRAMMING

Symbionts which are to operate under the control of OS 500 must observe the following conventions in allocating the first 46 bytes of their area:

Bytes	Usage
00–03	Upon taking control, the symbiont must save the PSC here.
04–19	Upon taking control, the symbiont must save the processor registers here.
20–27	If the symbiont is an inquiry symbiont, it must define its name here. See 2.6.1 for symbiont naming conventions.
28–29	If the symbiont is an inquiry symbiont, it must place its I/O mode activation routine address here.
30–45	This area is reserved for the system to save the symbiont's processor registers when suspending the symbiont.

If the symbiont is using DTFUQ, the areas described above will be generated automatically.

When a symbiont operating in processor mode issues an MSG macro, the OS 500 supervisor suspends that symbiont until a //ANS typein replies to the MSG.

4.3. OS 500 SUPERVISOR GENERATION

Two declarative macro instructions are provided for OS 500 supervisor generation. The call names for these macros are:

- OS5N – for OS 500 generation in the nonconcurrent environment.
- OS5C – for OS 500 generation in the concurrent environment

One or the other call name must be specified in the operation field of the macro instruction.

The format of the macro instruction for OS 500 supervisor generation is:

LABEL	♭ OPERATION ♭	OPERAND	72
unused	OS5N OS5C	CNSL=n,	X
		LCTC=n,	X
		LCTU=n,	[X]
		[,INQL=n]	[X]
		[,INQR=n]	[X]
		[,RCTC=n]	[X]
		[,RCTN=n]	[X]
		[,RCTU=n]	[X]
		[,RITB=label]	[X]
		[,ROLL={8410 / 8411 / 8414}]	[X]
		[,ROLU=n]	[X]
		[,ROTB=label]	[X]

In the operation field, either OS5N or OS5C must be specified. Required and optional keyword parameters appear in the operand of this instruction. A description of the individual parameters follows, and a summary of the keyword parameters is given in Table 4–2.

NOTE:

Keyword parameters used in generating the supervisor in a concurrent or nonconcurrent operating system are applicable for generation of the OS 500 supervisor. However, only those keyword parameters which directly apply to the generation of the OS 500 supervisor are described in this manual. Applicable keyword parameters are specified with the OS5C or OS5N declarative macro instruction. For a description of these keyword parameters, refer to *UNIVAC 9200/9200 II/9300/ 9300 II Systems Operating System Programmers Reference, UP-7531* (current version).

- Console Logical Unit Number

  This keyword parameter is required. It identifies the logical unit number of the online console/ inquiry unit.

  Format:

  CNSL = n

  where:

  n specifies the logical unit number of the console/inquiry unit as a decimal number

- Local Inquiry Stations

  This keyword parameter is used to specify the total number of online inquiry stations. This parameter may be omitted if there is only one local console and it is not to be used for inquiry programs.

  Format:

  INQL = n

  where:

  n specifies a decimal number

- Console/Inquiry Unit Channel

  This required keyword parameter is used to specify the channel number of the first (or only) local console/inquiry unit.

  Format:

  LCTC = n

  where:

  n specifies a decimal number between 5 and 12

  NOTE:

  All local console/inquiry units must be assigned consecutive channel numbers.

- Console/Inquiry Unit Number

  This required keyword parameter is used to specify the logical unit number of the first (or only) local console/inquiry unit.

  Format:

  LCTU = n

  where:

  n specifies a logical unit number expressed as a decimal number

  NOTE:

  All local console/inquiry units must be assigned consecutive logical unit numbers.

- Remote Inquiry Stations

  This keyword parameter is required only if remote inquiry stations are included in the configuration. It specifies the total number of remote inquiry stations.

  Format:

  INQR = n

  where:

  n specifies a decimal number

- Remote Communications Terminal Channel

  This keyword parameter is required when remote inquiry stations are included in the configuration. It is used to specify the first remote communications terminal channel number.

  Format:

  RCTC = n

  where:

  n specifies a decimal number

  NOTE:

  All remote inquiry stations must be assigned consecutive channel numbers.

- Line Terminal Pairs

  This keyword parameter is required only when remote inquiry stations are included in the system. It specifies the total number of line terminal pairs used for the remote inquiry stations.

  Format:

  RCTN = n

where:

n specifies a decimal

- Remote Communications Terminal Unit Number

This keyword parameter must be specified when remote inquiry stations are included in the configuration. This parameter is used to specify the first logical unit number of the remote communications terminal device.

Format:

RCTU = n

where:

n specifies a logical unit number expressed as a decimal number

NOTE:

All remote inquiry stations must be assigned consecutive logical unit numbers.

- Remote Device Input Translation Table

This optional keyword parameter specifies the name of the input translation table used only when remote inquiry stations are included in the system.

Format:

RITB = label

where:

label is the base address of the remote device input translation table, which must be linked to the supervisor.

NOTE:

A standard ASCII-to-EBCDIC translation table labeled ASEB is available in the Univac library.

- Rollin/Rollout Disc Type

This keyword parameter is specified only when program swapping is desired. It indicates the disc subsystem to be used for the rollin/rollout function.

Format:

$$\text{Roll} = \left\{ \begin{array}{c} 8410 \\ 8411 \\ 8414 \end{array} \right\}$$

where:

the specification used depends on the disc type used in the system

- Rollin/Rollout Unit

  This keyword parameter is specified only when program swapping is desired. It is used to specify the logical unit number of the disc subsystem used for the rollin/rollout function.

  Format:

  ROLU = n

  where:

  n is the logical unit number expressed as a decimal number

- Remote Device Output Translation Table

  This optional keyword parameter specifies the name of the output translation table used only when remote inquiry stations are included in the system.

  Format:

  ROTB = label

  where:

  label is the base address of the remote device output translation table, which must be linked to the supervisor

  NOTE:

  A standard EBCDIC-to-ASCII translation table labeled EBAS is available in the Univac library.

Keyword	Specification	Use	Remarks
CNSL	n = decimal number	R	Identifies logical unit number of console/ inquiry device
INQL	n = decimal number	X	Specifies total number of online inquiry stations
INQR	n = decimal number	X	Specifies total number of remote inquiry stations
LCTC	n = decimal number between 5 and 12	R	Specifies initial channel number of first online inquiry device, or channel number if console-only configuration
LCTU	n = decimal number	R	Specifies initial logical unit number of first online inquiry device, or console logical unit number if console-only configuration
RCTC	n = decimal number	X	Specifies initial channel number of first remote inquiry device
RCTN	n = decimal number	X	Specifies total number of line terminal pairs
RCTU	n = decimal number	X	Specifies initial logical unit number of first remote inquiry device
RITB	label	X	Specifies name of input translation table
ROLL	8410 8411 8414	X	Specifies disc subsystem used for rollin/roll-out of main program
ROLU	n = decimal number	X	Specifies logical unit number of disc subsystem used for rollin/rollout
ROTB	label	X	Specifies name of output translation table

LEGEND:

R = Required
X = Optional

Table 4–2. Summary of OS 500 Supervisor Generation Keyword Parameters

Example:

A typical generation of OS 500 in a concurrent environment is:

- for the supervisor

```
LABEL      OPERATION      OPERAND                                  72  80
           OSSC           ALTR=ALL,
                          ROLL=8,4,1,4,
                          ROLU=04,
                          DCU1=X'30',
                          UNIT1=1,
                          SIZE=32,
                          CSR=CRD,
                          ITBL=TBRD,
                          REP=YES,
                          SYST=TAPE,
                          DEVA=X'08',
                          INQL=2,
                          LCTC=1,1,
                          LCTU=5,
                          CNSL=6,
                          COMM=YES,
                          RCTC=2,2,
                          RCTN=1,
                          RCTU=7,
                          TPC1=8
           END
```

- for the LU/PU table

```
LABEL     OPERATION     OPERAND
1         10            16
          CFGOA         UNIT=1
          PUTBL         TAPE,8000000,9,,S
          PUTBL         TAPE,8111111,9
          PUTBL         TAPE,8222222,9
          PUTBL         TAPE,8333333,9
          PUTBL         DISC,16,X'70',04
          PUTBL         TYPW,12,0,06
          PUTBL         CRD,1,,,8,C
          PUTBL         CRP,2,,,9
          PUTBL         PRNT,3,,,10
          PUTBL         INQ,23,0,0,7,B,22,X'41',X'56'
          PUTBL         INQ,11,0,05
          END
```

NOTE:

UNIT = 1 indicates one remote inquiry unit connected to the system.

4.3.1. Supervisor Generation Considerations

For local inquiry stations, the first channel number, specified by the LCTC parameter, and the first logical unit number, specified by the LCTU parameter, will be incremented contiguously for the total number of local inquiry stations specified by INQL to determine subsequent channel and logical unit numbers. Therefore, these devices must occur contiguously within the hardware channel and logical unit/physical unit (LU/PU) device assignment.

The same applies to the remote inquiry station assignments by use of the RCTC, RCTU, and INQR keyword parameters. Continuity between local and remote assignments, however, need not be contiguous.

4.4. OS 500 JOB CONTROL

Two declarative macro instructions are provided for OS 500 job control generation. The call names for these instructions are:

JCNC— for OS 500 generation in the nonconcurrent environment

JCCN— for OS 500 generation in the concurrent environment

Format:

LABEL	b OPERATION b	OPERAND	72
unused	JCNC JCCN	[COMM=YES]	[X]
		[,SYST= {8410 / 8411 / 8414}]	[X]
		[,CLOG= {ALL / JECF}]	[X]
		[,CNSL=n]	[X]
		[,OS5H=YES]	

- Communications Handling

  This keyword parameter is required when communications handling is desired.

  Format:

  COMM=YES

  where:

  YES specifies communications handling

- System Orientation

  This keyword parameter identifies the orientation of the operating system.

  Format:

  SYST= {8410 / 8411 / 8414}

  where:

  the specification identifies the orientation of a disc-resident operating system. The parameter is omitted when the operating system is tape-resident.

- Console Logging

  This keyword parameter causes logging of all or some of the control stream cards.

  Format:

  CLOG= {ALL / JECF}

  where:

  ALL specifies the listing of all job control cards and JECF specifies listing only the 'JOB', 'EXEC', 'CALL', and 'FINIS' cards

- Console Logical Unit Number

This keyword parameter identifies the logical unit number of the online console/inquiry unit. It is required if CLOG is specified.

Format:

CNSL=n

where:

n specifies the logical unit number of the console/inquiry unit as a decimal number. This parameter must also be used in OS 500 supervisor generation.

- Local or Remote Inquiry Devices

This keyword parameter is required only for a local inquiry operation, and is omitted when remote communication devices are part of the system.

Format:

OS5H=YES

NOTE:

Parameter COMM=YES replaces OS5H when remote communication devices are part of the system.

Example:

A typical generation of OS 500 job control in a concurrent environment is:

LABEL 1	OPERATION 10	OPERAND 16	72	80
	JCCN	COMM=YES,	X	
		SYST=84,1,1,	X	
		CLOG=ALL,	X	
		CNSL=5		
	END			

APPENDIX A. CHARACTER CODES

The following table lists the characters on the keyboard of the console/inquiry unit and the DCT 500 device, along with the equivalent ASCII and EBCDIC codes. Note that the characters SP and DEL are not printable.

CHARACTER	$ASCII_8$	$EBCDIC_{16}$	CHARACTER	$ASCII_8$	$EBCDIC_{16}$	CHARACTER	$ASCII_8$	$EBCDIC_{16}$
LF	012	25	)	076	6E	←	137	6D
CR	015	0D	?	077	6F	\	140	79
FF	014	0C	@	100	7C	a	141	81
SP	040	40	A	101	C1	b	142	82
!	041	4F	B	102	C2	c	143	83
"	042	7F	C	103	C3	d	144	84
#	043	7B	D	104	C4	e	145	85
$	044	5B	E	105	C5	f	146	86
%	045	6C	F	106	C6	g	147	87
&	046	5D	G	107	C7	h	150	88
'	047	7D	H	110	C8	i	151	89
(	050	4D	I	111	C9	j	152	91
)	051	5D	J	112	D1	k	153	92
*	052	5C	K	113	D2	l	154	93
+	053	4E	L	114	D3	m	155	94
,	054	6B	M	115	D4	n	156	95
-	055	60	N	116	D5	o	157	96
.	056	4B	O	117	D6	p	160	97
/	057	61	P	120	D7	q	161	98
0	060	F0	Q	121	D8	r	162	99
1	061	F1	R	122	D9	s	163	A2
2	062	F2	S	123	E2	t	164	A3
3	063	F3	T	124	E3	u	165	A4
4	064	F4	U	125	E4	v	166	A5
5	065	F5	V	126	E5	w	167	A6
6	066	F6	W	127	E6	x	170	A7
7	067	F7	X	130	E7	y	171	A8
8	070	F8	Y	131	E8	z	172	A9
9	071	F9	Z	132	E9	{	173	C0
:	072	7A	[	133	4A	:	174	6A
;	073	5E	/	134	E0	}	175	D0
<	074	4C	]	135	5A	~	176	A1
=	075	7E	^	136	5F	DEL	177	07

Table A–1. Character Codes

APPENDIX B. OS 500 STORAGE REQUIREMENTS

Each of the functions of OS 500 requires storage. These storage requirements are given in Table B—1. Note that all values specified are approximate, and that storage requirements for buffers are not included.

Function	Requirement (Bytes)
Console control	700
Basic inquiry	400
Local dispatcher	800
Remote dispatcher	1500
Basic rollout/rollin	400
UNIVAC 8410 rollout/rollin	550
UNIVAC 8411/8414 rollout/rollin	750
New display subroutine	550
Symbiont inquiry	100
Inquiry logical IOCS integrated with main inquiry program	800
Inquiry logical IOCS integrated with symbiont inquiry program	1000
Additional storage required for remote capability	400

Table B—1. OS 500 Storage Requirements

INDEX

Comments concerning this manual may be made in the space provided below. Please fill in the requested information.

System: _____

Manual Title: _____

UP No: _____ Revision No: _____ Update: _____

Name of User: _____

Address of User: _____

Comments:

CUT

FOLD

FIRST CLASS
PERMIT NO. 21
BLUE BELL, PA.

BUSINESS REPLY MAIL NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

UNIVAC

P.O. BOX 500

BLUE BELL, PA. 19422

ATTN: SYSTEMS PUBLICATIONS DEPT.

FOLD

CUT